UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale SST

Laboratoire d'Informatique : EA2101

Équipe Ordonnancement et Conduite

THÈSE présentée par :

Gaël SAUVANET

soutenue le : 5 avril 2011

pour obtenir le grade de : Docteur de l'université François Rabelais - Tours

Discipline / Spécialité : Informatique

Recherche de chemins multiobjectifs pour la conception et la réalisation d'une centrale de mobilité destinée aux cyclistes

THÈSE DIRIGÉE PAR :

BAPTISTE Hervé	Maître de Conférences, Université François Rabelais Tours
NÉRON Emmanuel	Professeur des Universités, Université François Rabelais Tours

RAPPORTEURS :

ARTIGUES Christian	Chargé de recherche HDR, LAAS CNRS
WOLFLER CALVO Roberto	Professeur des Universités, Université Paris 13

JURY :

ARTIGUES Christian	Chargé de recherche HDR, LAAS CNRS
BAPTISTE Hervé	Maître de Conférences, Université François Rabelais Tours
CARLIER Jacques	Professeur des Universités, Université de Technologie de Compiègne
GALAND Lucie	Maître de Conférences, Université Paris Dauphine
LIBERTI Leo	Professeur des Universités, École Polytechnique
NÉRON Emmanuel	Professeur des Universités, Université François Rabelais Tours
WOLFLER CALVO Roberto	Professeur des Universités, Université Paris 13

MEMBRE INVITÉ :

GRUNBERG Benoit	Directeur, La Compagnie des Mobilités, Tours

Remerciements

Le travail présenté dans cette thèse a été réalisé dans le cadre d'une convention CIFRE au sein de l'association Autour du Train, puis de l'entreprise la Compagnie des Mobilités, et de l'équipe Ordonnancement et Conduite du Laboratoire d'Informatique de l'Université François Rabelais Tours situé à Polytech'Tours. Je tiens donc à remercier l'ensemble des personnes travaillant à la Compagnie des Mobilités et à Polytech'Tours pour leur accueil durant ces trois années de thèse.

Je tiens à remercier Benoît Grunberg qui a permis de débuter ce travail de thèse. Pouvoir partager avec lui le développement de Géovélo et la création de la Compagnie des Mobilités a été très enrichissant. Je le remercie également pour m'avoir fait partager sa passion du vélo et pour m'avoir laisser suffisamment de temps pour la recherche lorsque c'était nécessaire. Merci également à l'ensemble des personnes qui ont travaillé sur ce projet et en particulier à Jocelyn de Sinety pour son travail de terrain sur Paris et à Jiehao Yuan pour le développement de l'application sur téléphones mobiles. Enfin, je remercie Emmanuel Dewaele, nouveau doctorant et employé de la Compagnie des Mobilités, pour m'avoir permis d'aborder avec lui les problèmes de plus court chemin multimodaux.

Je remercie plus particulièrement Emmanuel Néron pour son encadrement dans la bonne humeur durant ces trois années. Il a su se rendre disponible malgré ses lourdes charges administratives. Merci également à Hervé Baptiste pour m'avoir fait découvrir le domaine de l'aménagement à travers plusieurs co-encadrements de projets de fin d'études. Je remercie aussi Jean Charles Billaut pour son accueil au sein du Laboratoire d'Informatique. Enfin, je voulais remercier l'ensemble des stagiaires et étudiants qui ont été amenés à travailler sur le projet et en particulier : Zhu Yu, Zhang Juan et Shang Ke.

Merci également à l'ensemble des membres du jury et plus particulièrement aux deux rapporteurs Christian Artigues et Roberto Wolfler Calvo.

Merci à tous les doctorants du bureau 202 et aux autres pour la bonne humeur et les discussions scientifiques ou non que nous avons pu partagées.

Enfin, merci à ma famille et à ma compagne qui m'ont encouragé à faire une thèse et qui ont toujours essayé de comprendre mon travail durant ces trois années.

Résumé

Les travaux présentés dans cette thèse visent à proposer des méthodes permettant de calculer des itinéraires adaptés aux cyclistes à l'échelle d'une agglomération. Contrairement au problème du plus court chemin classique, le contexte est ici multiobjectif puisque plusieurs critères, comme la distance, l'insécurité et l'effort, doivent être pris en compte dans le calcul des itinéraires. Dans un problème multiobjectif, il n'existe pas qu'une seule solution mais un ensemble de solutions de compromis. La difficulté est alors de calculer des chemins sous une contrainte de temps de quelques secondes pour pouvoir intégrer ce calculateur à un site web par exemple.

Deux approches ont été abordées pour résoudre ce problème. L'approche *a posteriori* consiste à calculer l'ensemble des solutions de compromis. Une méthode classique de la littérature est présentée et améliorée en utilisant des prétraitements. Les améliorations proposées reposent sur des recherches de plus court chemin mono-objectif qui permettent de calculer sur chacun des nœuds du graphe, des bornes inférieures et supérieures sur les coûts des chemins de ce nœud vers le nœud destination.

La deuxième approche *a priori* prend en compte les préférences de l'utilisateur pour se concentrer sur le calcul d'une solution de meilleur compromis. La méthode utilisée permet d'orienter la recherche des chemins, de façon à privilégier les sous-chemins les plus prometteurs du point de vue des préférences de l'utilisateur.

Enfin, nous proposons de modéliser le réseau routier sous la forme d'un graphe adjoint pour pouvoir prendre en compte de nouveaux critères comme le temps de parcours ou la linéarité de l'itinéraire et de nouvelles contraintes comme des manœuvres interdites. Ces critères et contraintes nécessitent de pouvoir résoudre le problème du plus court chemin multiobjectif dans lequel des coûts peuvent être associés sur les arcs mais également sur les nœuds ou sur des enchaînements d'arcs.

L'ensemble de ce travail a permis de développer le service Géovélo qui est un calculateur d'itinéraires adaptés au vélo et entièrement multiobjectif. Le service est disponible sous la forme d'un site web et d'applications mobiles.

Mots-clés : problème de plus court chemin, graphe, optimisation multiobjectif.

Abstract

The work presented in this thesis aims at proposing methods for computing bicycle paths across a metropolitan. Unlike the problem of the classical shortest path, here the context is multiobjective because several criteria such as distance, safety and effort must be considered in the path computation. In a multiobjective problem, there is no single solution, but a set of compromise solutions. Then, the difficulty is to compute paths under a time constraint of a few seconds, in order to integrate the computation in the respond-time of a web page for example.

Two approaches were discussed to solve this problem. The first one is an *a posteriori* approach where all compromise solutions are computed. A classical method of the literature is presented here and improved by using preprocessing. The proposed improvements are based on single objective shortest path searches in order to compute lower and upper bounds on the costs of paths from all nodes to the target node.

The second approach is an *a priori* method that takes user preferences into account to focus on the computation of the best compromise solution. The method allows to guide the search by the selection of the most promising sub-paths first, according to the user preferences.

Finally, we propose to model the road network as a line graph to take into account new criteria such as travel time or the linearity of the path and new constraints as prohibited turns. To take into account these new criteria and constraints, it's necessary to define costs on the nodes, on the arcs and on arc sequences.

All this work was necessary to develop the service Géovélo, which is a multiobjective route planner adapted to bicycle. The service is available on a website and as mobile applications.

Keywords : shortest path problem, graph, multiobjective optimization.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1

Introduction

Aujourd'hui, face notamment aux problèmes de pollution et de l'augmentation du prix du pétrole, les modes de transport que l'on appelle doux, tel que le vélo, sont en plein développement en milieu urbain. Il n'y a encore que quelques années, le vélo était principalement considéré comme un simple objet de loisir ou de pratique sportive. Il est aujourd'hui en train de devenir un moyen de transport particulièrement prisé par les usagers et les collectivités. Pour les usagers, le vélo est un mode de transport économique, qui maintient en bonne santé et qui est même devenu tendance. Pour les collectivités, il s'agit d'un moyen économique participant au désengorgement des réseaux et permettant la réduction de la pollution des centres-villes. A noter que plusieurs textes de loi, comme la loi Solidarité et Rénovation Urbaine de 2000, imposent aux collectivités de prendre en compte ces nouveaux modes de déplacements doux.

Dans le **chapitre 2**, nous détaillons ce contexte favorable au développement du vélo. De nombreuses politiques cyclables émergent dans les collectivités. Les vélos en libre service tels que Vélib' à Paris ou Vélo'V à Lyon, ou en location longue durée, comme à Tours, sont les plus médiatisés et connaissent un succès important. Les modifications de voiries sont également de plus en plus nombreuses ces dernières années : pistes et bandes cyclables, voies partagées avec les bus, zones limitées à une vitesse de 30 km/h, etc.

Cependant, il existe encore très peu de services d'information associés. Ces services sont nécessaires pour mettre en valeur un mode de transport et accompagner les usagers. Par exemple, il est aujourd'hui impensable de ne pas avoir accès à un site web pour choisir et réserver son billet de train. De même, il est devenu indispensable pour beaucoup d'automobilistes de repérer leur trajet sur un site de calcul d'itinéraires ou de se laisser guider par leur GPS. Ce genre de services est inexistant en France pour le mode de transport à vélo.

Pourtant, un service de calcul d'itinéraires adaptés au vélo peut permettre aux collectivités de communiquer sur les aménagements cyclables, de rassurer un usager en lui montrant visuellement qu'il existe un itinéraire sécurisé pour se rendre par exemple à son travail, etc. La principale difficulté dans la réalisation d'un tel service concerne les critères à prendre en compte. Contrairement à un service de calcul d'itinéraires pour les automobilistes, où les critères sont connus (distance, temps, coût, etc.), les critères entrant en jeu

dans le choix d'un itinéraire adapté aux cyclistes sont moins évidents à définir. En effet, d'après plusieurs recherches et enquêtes, les principaux éléments jouant un rôle dans le choix d'un itinéraire concernent la notion de chemin direct, l'aspect sécurisé et l'effort à fournir. La difficulté est alors de modéliser ces critères et d'avoir à disposition les données nécessaires.

Dans le **chapitre 3**, nous présentons le contexte de cette thèse et modélisons le problème. Nous nous intéressons aux problèmes de cheminement et notamment au problème du plus court chemin. En effet, un réseau routier peut être modélisé simplement sous la forme d'un graphe où les nœuds représentent les intersections des routes, et les arcs représentent les tronçons de route entre les intersections. Depuis Dijkstra [Dijkstra 59], la littérature est vaste concernant le problème du plus court chemin. De nombreuses méthodes, classiques et avancées, permettent aujourd'hui de résoudre très rapidement ce problème sur des graphes de plusieurs millions de nœuds. La plupart des méthodes avancées reposent sur des prétraitements importants qui permettent d'orienter la recherche pour réduire le nombre de nœuds traités.

La principale difficulté de ce travail de thèse est liée au contexte multiobjectif, étant donné que plusieurs critères entrent en compte dans le choix d'un itinéraire adapté aux cyclistes. En effet, dans un problème multiobjectif, il n'existe pas une unique solution, mais un ensemble de solutions de compromis pour lesquelles il n'existe aucune solution meilleure sur l'ensemble des objectifs. La dominance de Pareto permet de définir les relations entre les solutions d'un problème multiobjectif. Par exemple, entre un itinéraire de 1 km dangereux et un itinéraire de 10 km sécurisé, il peut exister de nombreux itinéraires correspondant à des compromis différents entre la distance et la sécurité.

Le nombre de solutions non-dominées d'un problème multiobjectif peut être très important et en plus de la complexité combinatoire qu'implique la résolution de ce problème, il n'est pas forcément évident de faire un choix parmi ces solutions. Plusieurs approches existent pour résoudre ce problème en fonction du moment où intervient le décideur pour préciser ses préférences. Dans l'approche *a posteriori*, l'ensemble des solutions non-dominées sont calculées puis présentées au décideur qui effectue son choix. L'approche *a priori* est souvent plus rapide en termes de temps de calcul puisque le décideur spécifie ses préférences avant le lancement de la méthode de résolution. Enfin, dans l'approche *interactive*, les préférences du décideur vont s'affiner pendant la méthode de résolution. Deux phases, l'une de calcul d'une ou plusieurs solutions de compromis, et l'autre de dialogue pour spécifier les préférences, sont alors répétées alternativement jusqu'à l'obtention d'une solution satisfaisante.

La problématique de cette thèse est donc de calculer des chemins non-dominés dans un contexte multiobjectif de manière à pouvoir proposer un service de calcul d'itinéraires adaptés aux cyclistes. Ce service prend la forme d'un site web mais également d'applications mobiles. Par manque de données, nous nous sommes initialement concentrés sur le problème bi-objectif en prenant en compte la distance et l'insécurité d'un itinéraire. Ces deux critères ont été modélisés sous la forme d'objectifs de type **Min-Somme**. L'insécurité ne peut pas être modélisée par un objectif de type **Min-Max**, étant donné que les réseaux cyclables en France ne sont pas continus, c'est-à-dire qu'il est bien souvent nécessaire de parcourir des tronçons de route peu sécurisés pour rejoindre, par exemple, des pistes cyclables. Le

problème du plus court chemin bi-objectif avec deux objectifs de type **Min-Somme** est NP-Difficile. La difficulté est alors de pouvoir calculer des chemins de compromis sur des graphes dont la taille correspond au réseau routier d'une grande agglomération sous une contrainte de temps d'environ trois secondes pour pouvoir être intégrés, par exemple, à un site web.

Dans cette thèse, deux approches ont été envisagées :

- une approche *a posteriori* où toutes les solutions de compromis sont calculées dans un contexte bi-objectif,
- une approche *a priori* où une unique solution de meilleur compromis est calculée dans un contexte multiobjectif.

Dans le **chapitre 4**, nous nous intéressons à la détermination de l'ensemble des solutions non-dominées (approche *a posteriori*) en se limitant à seulement deux objectifs : minimiser la distance et minimiser l'insécurité des itinéraires. Dans ce contexte, il n'est pas nécessaire de connaitre les préférences de l'utilisateur.

Dans la littérature, les méthodes dites de *labelling* [Tarapata 07, Climaco 10] sont très souvent utilisées pour résoudre ce problème. Elles sont une généralisation des méthodes mono-objectif comme l'algorithme de Dijkstra. Sur chaque nœud se trouve un ensemble d'étiquettes caractérisées par un vecteur de coûts des objectifs et représentant un sous-chemin jusqu'à ce nœud. À chaque itération, une étiquette ou un ensemble d'étiquettes, est sélectionné et est étendu à l'ensemble des arcs sortants du nœud concerné. Les méthodes de *label-setting* [Hansen 80, Martins 84] sont souvent utilisées car elles assurent la propriété qu'à chaque étiquette traitée, il n'existe pas d'étiquette qui la dominera plus tard dans la recherche.

Étant donné que le problème du plus court chemin mono-objectif est très rapide à résoudre, il est possible de calculer des bornes inférieures et supérieures sur les coûts de l'ensemble des sous-chemins menant au nœud de destination. Il suffit pour cela de lancer des recherches mono-objectif inverses partant du nœud de destination et avec les arcs inversés.

En utilisant ces bornes, il est possible d'intégrer des règles d'élimination des étiquettes dans la méthode de *label-setting*. Dans [Tung 92], lors du traitement de chacune des étiquettes, si le vecteur de coûts de l'étiquette, complété par les bornes inférieures de son nœud est Pareto-dominé par une des solutions déjà calculée, alors l'étiquette peut être éliminée car elle ne mènera qu'à des chemins dominés.

Comme l'efficacité de ces règles d'élimination est directement liée à l'ensemble des chemins complets déjà calculés, nous proposons d'utiliser les bornes inférieures et supérieures pour construire, à chaque étiquette traitée, deux chemins réalisables pour ensuite les ajouter à l'ensemble des chemins complets s'ils ne sont pas dominés. Cette méthode permet donc d'accélérer la construction du front de Pareto et ainsi d'améliorer l'efficacité des règles d'élimination.

Dans les problèmes bi-objectif, il est courant d'utiliser la méthode à deux phases [Ulungu 95]. Lors de la première phase, l'ensemble des solutions supportées, c'est-à-dire se trouvant sur l'enveloppe convexe du front de Pareto et pouvant être déterminées par

une méthode mono-objectif optimisant une combinaison linéaire des critères, sont calculées. Lors de la deuxième phase, les informations de la première phase permettent de réduire l'espace de recherche pour calculer l'ensemble des solutions manquantes. Le calcul des solutions supportées peut également permettre d'initialiser la méthode de *label-setting* et ainsi de rendre encore plus efficace les règles d'élimination. Cependant, ce calcul peut être long si le nombre de solutions supportées est important. Nous proposons alors une alternative qui permet d'approximer très rapidement le front de Pareto d'une instance donnée. Cette méthode est basée sur un prétraitement important, inspiré de la méthode mono-objectif ALT [Goldberg 05b].

Les résultats expérimentaux obtenus sur des instances réelles sont encourageants. Les méthodes proposées permettent effectivement de réduire significativement le temps de calcul et de résoudre les petites et moyennes instances sous la contrainte de temps. Cependant, pour les instances de grande taille, ces méthodes sont toujours trop longues et ne peuvent pas être intégrées sous la forme d'un service web. De plus, seulement deux objectifs ont été considérés dans ce chapitre alors qu'en réalité, d'autres critères peuvent être pris en compte dans le calcul d'un itinéraire adapté aux cyclistes.

Ce travail a donné lieu à plusieurs communications dans des conférences nationales [Sauvanet 09] et internationales [Sauvanet 10e, Sauvanet 10g], ainsi qu'à une soumission en revue internationale [Sauvanet 10a].

Le **chapitre 5** est dédié au problème de calcul d'une solution de meilleur compromis dans un contexte multiobjectif. Le calcul de plusieurs centaines de solutions n'est effectivement pas pertinent pour l'utilisateur. Nous nous plaçons dans le cas où les préférences de l'utilisateur sont connues (approche *a priori*).

Dans la littérature, il est courant de définir les préférences du décideur par un vecteur de poids définissant l'importance de chacun des objectifs. Des fonctions d'agrégation des objectifs sont utilisées pour évaluer les performances d'une solution selon le vecteur de poids des préférences du décideur. La norme de Tchebycheff est souvent utilisée car elle permet de focaliser sur le pire des objectifs et de trouver des solutions équilibrées.

Trouver une solution minimisant une fonction d'agrégation non-linéaire, comme la norme de Tchebycheff, dans le problème du plus court chemin multiobjectif est un problème NP-Difficile. La méthode BCA* [Futtersack 00] permet de déterminer une solution de meilleur compromis en ordonnant la recherche dans l'ordre des étiquettes les plus prometteuses. Pour évaluer une étiquette, cette méthode calcule une borne inférieure de la fonction d'agrégation de cette étiquette en ajoutant à son vecteur de coûts les bornes inférieures de chacun des objectifs.

Plusieurs améliorations sont proposées en utilisant un prétraitement basé sur une recherche mono-objectif optimisant une combinaison linéaire des objectifs. Ces améliorations permettent une meilleure évaluation de chacune des étiquettes pour orienter au mieux la recherche.

La méthode BCA* initiale et les deux améliorations proposées sont testées sur les mêmes instances que dans le chapitre 4, ainsi que sur des instances nouvelles à trois objectifs, en introduisant le critère d'effort. Les améliorations se sont révélées particulièrement efficaces sur les instances les plus difficiles. Ainsi, avec deux objectifs, la plupart des instances

peuvent être résolues sous la contrainte de temps de trois secondes. Avec trois objectifs, les méthodes proposées sont plus efficaces qu'avec la méthode BCA* classique, mais ne permettent de résoudre que les classes d'instances les plus simples.

Ce travail a été présenté dans une conférence internationale [Sauvanet 10b] et a donné lieu à une soumission en revue internationale [Sauvanet 10c].

Dans le **chapitre 6**, nous nous intéressons à la modélisation avancée d'un réseau routier. Jusqu'à présent, le graphe utilisé est modélisé de façon à ce que les intersections des routes correspondent aux nœuds du graphe, et les tronçons de route correspondent aux arcs du graphe. Cependant ce modèle ne permet pas de modéliser tous les types d'objectifs.

Pour commencer, les algorithmes de plus court chemin ne peuvent pas prendre en compte des coûts sur les nœuds. Pourtant, ce type de coûts est pertinent pour des critères comme le temps de parcours ou encore l'insécurité de l'itinéraire. Par exemple, le temps de parcours peut dépendre du temps d'arrêt à certaines intersections (feux tricolores, stop, etc.) et l'insécurité peut dépendre des intersections dangereuses traversées. De plus, lorsque l'on calcule des itinéraires sur un réseau routier, il est essentiel de respecter le code de la route et donc de tenir compte des manœuvres interdites. Il est, par exemple, interdit de tourner à gauche sur certaines intersections importantes. Ces manœuvres interdites se modélisent sous la forme d'enchaînements d'arcs interdits. Enfin, il peut être intéressant de définir des coûts sur des enchaînements d'arcs. L'objectif de linéarité est introduit dans ce chapitre. Il consiste à minimiser le nombre de changements de direction. Pour cela, il est nécessaire de définir des coûts sur les enchaînements d'arcs représentant un changement de direction.

Dans ce chapitre, nous présentons l'utilisation du graphe adjoint pour tenir compte de coûts sur les nœuds, sur les arcs et sur les enchaînements d'arcs. Dans le graphe adjoint, les nœuds correspondent aux arcs du graphe initial, et les arcs du graphe adjoint modélisent la possibilité d'enchaîner deux arcs du graphe initial. L'avantage principal de cette modélisation est qu'elle permet d'utiliser les algorithmes classiques de plus court chemin sans aucune modification. L'inconvénient est que la taille du graphe adjoint est plus importante que la taille du graphe initial dans le cas d'un réseau routier.

Des expérimentations sont réalisées avec trois objectifs en utilisant la méthode de calcul de solution de meilleur compromis améliorée. L'objectif de minimisation du temps de parcours prend en compte des coûts sur les arcs et sur les nœuds. L'objectif de minimisation de l'insécurité est conservé à l'identique par rapport aux chapitres précédents. Enfin, la linéarité minimise le nombre de changements de direction et prend donc en compte des coûts sur les enchaînements d'arcs. Les résultats obtenus montrent que cette modélisation est tout à fait envisageable pour résoudre des instances à l'échelle d'un graphe représentant la ville de Paris et sous la contrainte de temps de trois secondes.

Ce travail a donné lieu à une communication en conférence nationale [Sauvanet 11].

Dans le **chapitre 7**, nous présentons l'application Géovélo (http://www.geovelo.fr) qui a été développée durant cette thèse. Ce service s'articule autour d'une base de données routières spécifique au vélo, d'un calculateur d'itinéraires multiobjectif et de plusieurs applications clientes (un site web et deux applications mobiles).

Les données utilisées proviennent principalement du projet libre OpenStreetMap, qui est une base de données routières communautaire. Un travail de terrain a permis de compléter et de vérifier les données des aménagements cyclables mais également de donner une note de cyclabilité pour chacun des tronçons de route.

Ce travail a donné lieu à une communication [Sauvanet 10f] lors de la conférence annuelle State Of The Map 2010 (Girone) du projet OpenStreetMap.

Le calculateur d'itinéraires utilise la méthode de calcul de solution de meilleur compromis présentée dans le chapitre 5. Les améliorations proposées dans ce chapitre permettent au calculateur de déterminer des itinéraires dans un contexte multiobjectif en quelques secondes au maximum.

Le site web offre la possibilité de sélectionner un point de départ et de destination, soit textuellement, en entrant l'adresse, soit directement en positionnant ces points sur la carte. L'interface propose ensuite plusieurs itinéraires à l'internaute correspondant à plusieurs compromis entre les objectifs. Une autre interface a été développée pour permettre à l'internaute de spécifier directement ses préférences.

Deux applications mobiles, une pour les téléphones iPhone, l'autre pour les téléphones Android, ont été réalisées pour pouvoir avoir accès au service Géovélo en situation de mobilité. Des fonctionnalités de géolocalisation ont également été implémentées pour faciliter leur utilisation.

Enfin, en conclusion, nous résumons l'ensemble du travail réalisé durant cette thèse et nous présentons plusieurs perspectives envisageables autour du problème de plus court chemin multiobjectif.

Chapitre 2

L'usage du vélo et les services associés

Ce travail de thèse s'inscrit dans un contexte favorable aux modes de transport que l'on nomme doux tel que le vélo. En effet, poussé par plusieurs textes de loi [1], le développement du vélo est pris très au sérieux par les collectivités qui l'intègrent de plus en plus dans leur politique de transport. Alors qu'aujourd'hui le vélo est encore avant tout un objet de loisir, les conditions semblent désormais réunies pour voir le vélo devenir un mode de déplacement à part entière.

Parmi les politiques cyclables, plusieurs dispositifs, comme les aménagements cyclables, les vélos en libre service (ou en location longue durée) et les zones limitées à 30 km/h, laissent une place de moins en moins anecdotique au vélo face à la voiture. Cependant, les services d'information associés sont encore peu développés mais sont pourtant déterminants [Uster 08] dans l'accompagnement des usagers. Le calcul d'itinéraires, un service classique pour les autres modes de transport, est inexistant en France pour les cyclistes. Contrairement au calcul d'itinéraires pour les voitures où les critères sont connus de tous (distance, temps, coût, etc.), les critères entrant en jeu dans le choix d'un itinéraire adapté aux cyclistes sont moins évidents à définir.

La section 2.1 présente le contexte de cette thèse. Dans la section 2.2, une étude est présentée sur l'importance des différents critères jouant un rôle dans le choix d'un itinéraire adapté aux cyclistes.

1. notamment, la loi Solidarité et Rénovation Urbaine (2000) impose aux agglomérations de plus de 100 000 habitants d'élaborer un Plan de Déplacements Urbains (PDU), de rééquilibrer l'usage des différents modes de transport et de prendre en compte les besoins de mobilité dans les documents de planification.

2.1 Contexte

2.1.1 Le vélo en France

C'est en 1817 [Malgras-Serra 06] que l'histoire du vélo ou de la bicyclette débute lorsque le baron allemand Karl Drais a l'idée de relier deux roues par une poutre en bois. Assis à califourchon sur cet engin, il parcourt alors presque 15 kilomètres en une heure. Par la suite, plusieurs étapes déterminantes permettront à la bicyclette de s'imposer rapidement comme le moyen de transport le plus utilisé au monde après la marche : l'ajout de la chaîne dans les années 1880, les pneumatiques en 1888, le dérailleur en 1911, etc.

> *C'est le contraire du vélo, la bicyclette. Une silhouette profilée mauve fluo dévale à soixante-dix à l'heure : c'est du vélo. Deux lycéennes côte à côte traversent un pont à Bruges : c'est de la bicyclette.* [Philippe Delerm]

Contrairement à d'autres pays comme la Chine où le vélo n'est utilisé quasi exclusivement qu'à des fins utilitaires, la pratique du vélo est très diversifiée en France. En effet, alors que le vélo était initialement en France un moyen de transport pratique et économique, il est devenu un objet de loisir et sportif. C'est en 1936 avec le début des loisirs de masse que le vélo devient un objet de détente et de promenade. Concernant la pratique sportive, c'est le Tour de France qui a largement contribué à son développement.

Le vélo comme mode de transport en perte de vitesse en France

Si le vélo à vocation utilitaire a perdu du terrain en France [Wachter 05] depuis les années 1950, cela s'explique par le développement de la voiture et plus récemment, à partir des années 1970–1980, par l'étalement urbain et la périurbanisation qui a augmenté les distances de déplacement, rendant difficiles, voire impossibles, les déplacements à vélo ou à pied, par exemple, pour aller travailler. Une étude [Dep 07] explique que les déplacements à vélo ne représenteraient que 3 à 4% des déplacements domicile-travail alors qu'ils étaient, par exemple, de 39% en 1959 [Madre 97]. La pratique utilitaire du vélo en France reste marginale par rapport à des pays comme les Pays-Bas où une étude [hol 09] montre que les déplacements à vélo représenteraient environ 26% des déplacements. Cette différence entre la France et les Pays-Bas n'est pas un hasard : dès les années 1970, les Pays-Bas ont débuté des politiques volontaristes visant à promouvoir le vélo, alors que ces politiques sont très récentes en France.

Aujourd'hui, environ deux tiers des déplacements en France [Carré 03] se font en voiture. L'utilisation quasi exclusive de la voiture commence à montrer ses limites. Si la pollution, le réchauffement climatique et l'encombrement du trafic sont souvent pointés du doigt, par l'ADEME notamment [2], il ne faut pas non plus oublier la consommation d'espace, le coût des infrastructures, le manque d'activité physique et la dégradation de la qualité de vie en ville qu'implique un usage intensif de la voiture.

2. l'Ademe (Agence de l'Environnement et de la Maîtrise de l'Energie, http://www.ademe.fr) participe à la mise en oeuvre des politiques publiques dans les domaines de l'environnement, de l'énergie et du développement durable.

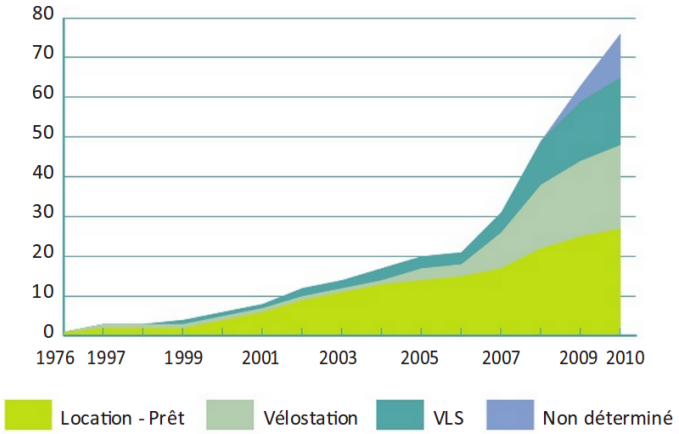


FIGURE 2.1 – Evolution du nombre de services vélos de 1997 à 2010 par type de service proposé (source : GART)

Les politiques cyclables

Les avantages de la pratique du vélo dit urbain sont tels que le développement du vélo fait aujourd'hui partie intégrante des politiques de déplacement des collectivités. Le vélo a en effet bien des avantages encore non exploités pour les collectivités. Pour commencer, le vélo participe à désengorger les centres-villes. Une part importante des déplacements pendulaires utilisant la voiture sont courts et pourraient aisément être remplacés par le vélo. Par rapport à la voiture, le vélo occupe moins d'espace, que ce soit pour le stationnement ou pour la circulation : aller au travail chaque jour à vélo parait plus adapté que de déplacer et de stationner un objet imposant comme une voiture [3], surtout que l'usager est souvent seul dans le véhicule. Par exemple à Tours [Madre 08], les déplacements dans l'agglomération sont en moyenne de 4,3 kilomètres pour un taux de remplissage moyen de la voiture de 1,1 passager.

Les autres avantages pour les collectivités et les usagers sont aussi variés que nombreux : le vélo est moins bruyant que la voiture, non polluant, économique pour l'usager et pour les collectivités (aménagements moins coûteux car leur emprise est plus faible), bon pour la santé, etc.

Ainsi, les collectivités ont tout intérêt à inciter les usagers à se mettre au vélo. Les politiques cyclables n'ont jamais été aussi actives en France et on assiste depuis quelques années au retour du vélo dit urbain.

Parmi ces politiques, la mise en place de vélos en libre service (VLS) dans les grandes agglomérations est en forte augmentation ces dernières années, comme le montre la figure 2.1. Les plus médiatisés sont Vélo'V à Lyon et Vélib' à Paris. Une alternative est la location longue durée, comme à Tours, où il est possible de louer un vélo à un prix attractif pour inciter à un usage plus important du vélo.

L'autre politique importante concerne les travaux pour partager la voirie au profit de

3. Une voiture occupe en moyenne 10 m^2, un bus 30 m^2, un vélo 1 m^2 [ade 06].

plusieurs modes de transport dont le vélo : pistes ou bandes cyclables, voies partagées avec les bus et les taxis, voies partagées avec les piétons, contre-sens cyclables, jalonnements, etc.

Il existe d'autres initiatives ou expérimentations qui sont pratiquées par les collectivités. Par exemple, la prévention sur la sécurité est importante car les français ne sont plus habitués à la circulation de vélos en ville. Il est donc nécessaire d'insister auprès des automobilistes pour qu'ils prennent conscience de la présence de ces nouveaux cyclistes, mais il faut également accompagner les cyclistes en leur expliquant les risques liés, par exemple, aux angles morts et leur apprendre la pratique du vélo en ville, qui n'est pas la même que celle du vélo loisir. Enfin, les collectivités mettent en avant les avantages du vélo et communiquent autour du réseau cyclable à travers des informations statiques : cartes des aménagements, rues conseillées, etc.

L'exemple de l'agglomération de Tours

Aujourd'hui, la part modale de l'utilisation du vélo varie beaucoup suivant les agglomérations, preuve de l'importance des politiques cyclables : de 1% environ pour des agglomérations comme Marseille et Aix-en-Provence à 10% pour Strasbourg.

L'agglomération de Tours l'a bien compris et a fait de nombreux efforts depuis une dizaine d'années. Tours a ainsi aménagé 135 kilomètres de voirie pour favoriser l'utilisation des vélos [tou 09]. Ces aménagements sont principalement des pistes et bandes cyclables mais également des contre-sens cyclables. Le nombre d'arceaux pour stationner son vélo a également augmenté. La part modale de l'utilisation du vélo dans l'agglomération de Tours se situe entre 3 et 4% [Madre 08].

Depuis 2006, l'agglomération de Tours propose un service de location de vélo longue durée. Ce service, nommé Vélociti, propose un vélo de ville à des tarifs attractifs. En effet, le prix des vélos de ville, qui est souvent de plusieurs centaines d'euros, est un des freins à l'utilisation d'un vélo en ville au même titre que la crainte du vol. Avec Vélociti, le tarif pour les étudiants est de 15 euros pour 3 mois ou seulement 6 euros s'ils sont abonnés au réseau de transport en commun de l'agglomération. Avec un tel tarif, le nombre de vélos en location est passé de 400 en 2006 à 1200 en 2010.

Les freins à l'utilisation du vélo

Entre les avantages du vélo comme mode de déplacement et les efforts consentis par les collectivités pour favoriser l'émergence de cette nouvelle pratique, quels sont les principaux freins à l'utilisation du vélo ?

D'après la FUBicy [fub 07], pour près de deux tiers des non-cyclistes, la crainte d'un accident est le facteur dissuasif d'une utilisation quotidienne du vélo. Pourtant le vélo n'est pas un mode de transport plus risqué que les autres. Pour comparaison, le vélo représente 4% des déplacements quotidiens pour 4% des blessés graves et 4% des tués, alors que les deux roues motorisées représentent 2% des déplacements quotidiens pour 30% des blessés

graves et 21% des tués en France [4]. En effet, la plupart des études montrent que le vélo est un moyen de transport sûr et que contrairement aux a priori de bon nombre de personnes, l'activité physique produite par la pratique du vélo permet au contraire d'allonger la durée de vie des cyclistes.

En fait, un des freins les plus importants pour la pratique du vélo n'est pas la sécurité réelle mais la **sécurité ressentie** [Gerber 09]. Il est important pour l'usager de se sentir en sécurité et le problème est surtout qu'il n'arrive pas à s'imaginer circuler à vélo quotidiennement.

2.1.2 L'association « Autour du train »

« Autour du train » est une association loi 1901. Elle a été créée en 2002 à Tours avec l'objectif de promouvoir les modes de déplacement doux et plus particulièrement le vélo. Pour cela, l'association s'est concentrée sur le développement de services et de produits destinés à un usage quotidien du vélo ou de loisir.

Ses principales réalisations sont :

- des cartes (Cyclocarte Touraine et Sologne) en coédition avec l'Institut Géographique National,
- le site Internet « Baladovélo » [5] qui propose de découvrir la ville de Tours à travers des balades thématiques à vélo,
- plusieurs études liées notamment à la complémentarité entre le vélo et le transport en commun.

Le manque d'outils efficaces pour calculer des itinéraires adaptés aux vélos et l'intérêt porté par l'association pour les nouvelles technologies sont les raisons principales du lancement de ce travail de thèse. Il est à noter que le travail réalisé durant cette thèse a poussé l'association à changer de statut pour devenir l'entreprise « la Compagnie des Mobilités » en mars 2010.

2.1.3 Un outil de calcul d'itinéraires adapté aux besoins des cyclistes

Une centrale de mobilité vélo est un ensemble d'outils à destination des cyclistes, pour les aider dans leurs déplacements, et à destination des collectivités, pour fournir des données utiles, telles que des statistiques et des mesures sur les pratiques des cyclistes (lieux de passages, retours des usagers, etc.). Un outil adapté aux besoins des cyclistes consisterait à proposer à l'usager des itinéraires **adaptés à la pratique du vélo** et à ses besoins, c'est-à-dire selon son origine, sa destination et ses préférences.

Ce type d'outil n'existe pas en France. Effectivement, il existe de nombreux outils pour les modes de déplacement tels que la voiture et le transport en commun, mais il n'est

4. ONISR, dossiers de presse DSCR mars 2005 et 2006.
5. http ://www.baladovelo.fr

pas proposé d'outil spécifique pour les cyclistes. Il est possible aujourd'hui d'estimer le temps nécessaire pour un déplacement en voiture, il est possible de préparer son voyage en train, de repérer son itinéraire et les changements de lignes en métro, mais il est pourtant impossible de faire de même pour un déplacement à vélo. Notons depuis quelques années l'apparition des SIM, systèmes d'information multimodale, qui permettent de calculer des itinéraires en utilisant plusieurs modes de transport. Cependant ces systèmes intègrent rarement le vélo et lorsqu'ils l'intègrent, ils ne prennent en compte aucune spécificité sur la sécurité ou la pénibilité.

Pourtant, face au contexte actuel, une centrale de mobilité vélo constitue un outil déterminant et complémentaire aux politiques cyclables d'une collectivité. Elle s'adresse évidemment aux non-cyclistes pour leur permettre de visualiser concrètement une ou plusieurs solutions alternatives pour leurs déplacements quotidiens. Cet outil implique l'usager en lui présentant des itinéraires personnels et le rassure en lui montrant notamment les aménagements cyclables qu'il pourrait emprunter. Une telle centrale de mobilité s'adresse également aux cyclistes qui peuvent préparer des itinéraires qu'ils ne connaissent pas, visualiser des itinéraires alternatifs ou repérer les aménagements cyclables présents dans un quartier particulier. Enfin une centrale de mobilité vélo peut également accompagner les collectivités pour détecter, par exemple, les manques en termes d'aménagements cyclables.

2.2 Critères pour élaborer un itinéraire adapté aux cyclistes

S'il existe peu d'outils dédiés aux cyclistes, cela s'explique en partie par le manque d'informations sur les caractéristiques des cyclistes. En effet, dans les enquêtes de déplacement, les cyclistes ont pendant longtemps été confondus avec les deux roues motorisées.

Il existe donc peu de recherches sur les critères jouant un rôle dans le choix d'un itinéraire par un cycliste. Malgré tout, dans [Noel 03], de nombreux critères sont cités par les cyclistes pour élaborer leur itinéraire : la recherche de rues locales (ou rues résidentielles), la recherche du chemin le plus court et le plus direct, la recherche des itinéraires offrant le plus d'aménagements cyclables, le chemin le plus sécurisé, celui évitant les grandes artères, la recherche des côtes les moins fortes, selon les conditions météorologiques et le paysage, etc. Bien sûr, il existe de nombreux liens entre tous ces critères. Par exemple la recherche d'un chemin le plus sécurisé dépend fortement de la recherche d'aménagements cyclables, de l'évitement des grandes artères, etc.

L'auteur insiste également sur l'importance du profil du cycliste qui influence largement l'importance de ces critères. Trois profils sont présentés :

- les cyclistes uniquement utilitaires,
- les cyclistes uniquement récréatifs,
- les cyclistes à la fois utilitaires et récréatifs.

Les critères qui reviennent le plus souvent pour les cyclistes utilitaires sont la recherche de rues locales, le chemin le plus court et le plus direct et la recherche d'aménagements

cyclables. Par contre, pour les cyclistes uniquement récréatifs, en plus de la recherche d'aménagements cyclables, l'auteur précise qu'il est important de tenir compte des conditions météorologiques et d'éviter les grandes artères. Les cyclistes à la fois utilitaires et récréatifs, eux, privilégient la recherche de rues locales et des aménagements cyclables.

Des études françaises [Carré 99, Carré 03] montrent que les habitudes des cyclistes utilitaires peuvent être caractérisées par deux stratégies bien différentes :

- la stratégie de l'écart qui consiste à éviter le trafic motorisé,
- la stratégie de l'intégration qui consiste à privilégier les routes directes et à intégrer la circulation automobile.

Bien sûr, ces deux stratégies correspondent à des profils de cyclistes bien particuliers. La stratégie de l'écart concerne souvent les cyclistes relativement lents et effectuant des trajets courts. La stratégie de l'intégration concerne des cyclistes plus à l'aise à vélo et qui sont également plus rapides.

De ces différentes études, plusieurs critères reviennent souvent et sont les plus importants. Dans la suite, chaque critère est détaillé mais également étudié d'un point de vue des données nécessaires à la modélisation d'un tel critère.

2.2.1 Recherche du chemin le plus direct

La recherche du chemin le plus court ou le plus direct est le critère le plus important. Plusieurs éléments peuvent entrer en considération :

- la distance totale de l'itinéraire, définie ici comme la distance kilométrique sur le réseau,
- le temps, ou durée, nécessaire pour parcourir l'itinéraire,
- l'évitement des stops et des feux impliquant des arrêts,
- le souhait de privilégier une vitesse constante et essayer de garder le plus possible une continuité du mouvement.

Si le critère de distance est le plus simple à mettre en œuvre étant donné qu'il ne nécessite que la longueur de chaque tronçon de route, il ne convient pas forcément à l'ensemble des utilisateurs qui peuvent chercher, non pas l'itinéraire le plus court en termes de distance, mais l'itinéraire qui nécessite le moins de temps de trajet.

Le critère de durée peut donc être plus approprié que la distance. Il permet en plus de prendre en compte d'autres éléments comme les stops ou les feux qui sont considérés comme des ralentissements pour les cyclistes. Enfin, le temps de parcours peut être calculé en fonction des pentes mais également en fonction du nombre d'intersections et du type de voies. Par exemple la circulation en centre ville peut prendre plus de temps qu'une circulation hors de la ville. Ou encore, sur les voies piétonnes, les cyclistes circulent moins vite que sur les autres voies car la circulation est partagée avec un mode de déplacement

plus lent.

2.2.2 Recherche du chemin le plus sécurisé

Un autre critère important est la sécurité de l'itinéraire, qu'elle soit réelle ou ressentie. Ce critère englobe la recherche de rues locales, la recherche d'aménagements cyclables, l'évitement des grandes artères et des intersections importantes.

Ce critère nécessite des informations tels que les aménagements cyclables, la vitesse de circulation des voitures, l'importance du trafic routier, etc. Dans [Ins 98], les auteurs définissent une notion d'attractivité pour les cyclistes qui est assez proche de la notion de sécurité. L'attractivité d'une route dépend de 20 indicateurs (aménagements, trafic automobile, etc.) pour déterminer 6 niveaux différents d'attractivité.

2.2.3 Recherche du chemin de moindre effort

Ce critère revient régulièrement parmi les études sur les cyclistes [Laporte 10]. Contrairement à la voiture ou aux transports en commun, la circulation à vélo nécessite pour le cycliste de fournir un effort physique. La minimisation de cet effort est donc un critère plus ou moins important selon les cyclistes : un cycliste utilitaire aura tendance à privilégier les itinéraires avec des pentes faibles alors que cela posera souvent moins de problèmes à un cycliste récréatif. Ce critère est directement lié aux pentes des voies et nécessite donc des informations altimétriques.

2.2.4 Recherche du chemin le plus agréable

Ce critère peut intervenir pour plusieurs profils de cyclistes. Les promeneurs ou touristes (cyclistes récréatifs) vont très souvent donner autant d'importance à la sécurité qu'au caractère agréable d'un itinéraire. De nombreux éléments peuvent alors entrer en jeu : voie en bordure de forêt, voie proche d'un cours d'eau, patrimoine rencontré, etc.

Concernant le profil utilitaire en milieu urbain, il est également important pour la plupart des cyclistes de privilégier des voies animées par la présence de commerçants, de promeneurs, etc.

2.3 Données routières disponibles

Nous avons pu voir que de nombreux éléments pouvaient entrer en compte dans le calcul d'un itinéraire adapté aux cyclistes. Le premier problème concerne les données routières nécessaires. En France, différents types de données existent : des données propriétaires et des données libres.

L'IGN (Institut Géographique National) est un établissement public national qui a pour objectif de produire, d'entretenir et de diffuser l'information géographique en France. Si l'IGN propose plusieurs types de bases de données routières, il ne commercialise plus de

bases de données dédiées à la navigation. Une base de données routières permettant de faire de la navigation doit posséder, en plus de la géométrie des routes, les sens de circulation précis, les manœuvres interdites, etc. Ainsi, même le GPS commercialisé par l'IGN n'utilise pas exclusivement les données de l'IGN et des données provenant de l'opérateur de bases de données routières Navteq sont intégrées. Au début de cette thèse, les données IGN ont été utilisées pour élaborer un prototype de calcul d'itinéraires vélo sur l'agglomération de Tours. Malgré la qualité et la précision des données, nous avons finalement dû changer de base de données, suite notamment à la décision de l'IGN d'arrêter de mettre à jour les sens de circulation des voies.

Les deux autres possibilités propriétaires sont les solutions Tele Atlas et Navteq. Ces deux solutions sont relativement similaires. Pour des raisons de coût, il a été décidé de choisir Tele Atlas pour tester les données et réaliser une version publique du calculateur sur l'agglomération de Tours. Contrairement aux données de l'IGN, les données Tele Atlas sont de moins bonne qualité : la position géographique des routes est plus approximative et de nombreuses erreurs sont visibles. Pour anecdote, Tele Atlas possède une interface web pour signaler les erreurs cartographiques : j'ai signalé de nombreux problèmes dans le quartier où se situe mon laboratoire car le quartier est nouveau et qu'il y a eu de nombreux changements, et un an après, aucune modification n'a été apportée sur la carte. La mise à jour des données est donc l'inconvénient majeur de ce type de solutions. De plus, sur les bases de données de l'IGN, il n'y a pratiquement pas de pistes ou bandes cyclables renseignées, mais la plupart des voies piétonnes et des chemins sont tout de même présents. Avec les données Tele Atlas ou Navteq, seules les routes dédiées à la circulation automobile sont présentes, à de rares exceptions près.

Une dernière solution existe, il s'agit de la base de données communautaire et libre OpenStreetMap [6]. Ce projet est parti d'un constat simple : dans la plupart des pays, les données géographiques ne sont pas libres. Assez souvent, ce sont les gouvernements qui financent des agences pour s'occuper de collecter ces informations. Cependant, si un citoyen veut avoir accès à ces informations géographiques, il devra quand même payer. Il ne pourra pas non plus corriger ou améliorer ces données s'il détecte des anomalies. Les cartes ne peuvent pas non plus être recopiées, partagées, etc. Au contraire, avec OpenStreetMap, ces différentes façons d'utiliser les données sont accessibles à tous. De plus les données étant libres, de nombreux services voient le jour. Voici quelques exemples de ce qu'il est possible de faire avec le projet OpenStreetMap :

- un utilisateur peut aller sur le site d'OpenStreetMap, récupérer un morceau de carte et l'intégrer dans un livre sans avoir de problème de droits de propriété,
- un service [7] a été créé autour d'OpenStreetMap permettant de générer des cartes grand format des villes, avec un index des rues, pouvant être imprimés par exemple et emmenés en voyage,
- comme les données sont accessibles à tous, de nombreux rendus cartographiques spécifiques ont vu le jour : pour les cyclistes, kayakistes, randonneurs, etc.

6. http://www.openstreetmap.org
7. http://www.maposmatic.org

Cependant, toutes ces données doivent être ajoutées par des contributeurs. Le nombre de contributeurs a dépassé la barre des 300 000 inscrits au cours de l'année 2010. Pour donner un ordre d'idée, la base de données OpenStreetMap représente quelque 800 millions de points géographiques et environ 60 millions de routes. En moyenne, un million de nouveaux points et 150 000 nouvelles routes sont ajoutés chaque jour.

De plus, OpenStreetMap est, à ce jour, la base de données routière la plus avancée en ce qui concerne les informations dédiées aux aménagements cyclables en France. Au cours de ce travail de thèse, la simplicité et la rapidité des ajouts et corrections dans OpenStreetMap ont fini de nous convaincre d'utiliser OpenStreetMap comme source de données principale pour la centrale de mobilité Géovélo.

2.4 Conclusion du chapitre

Dans ce chapitre nous avons pu voir que le mode de déplacement à vélo, notamment en milieu urbain, est en plein développement depuis quelques années. Ce développement est la conséquence des politiques cyclables mises en place par les collectivités : vélo en libre service, aménagements cyclables, etc. Cependant, à part des cartes des aménagements cyclables, très peu de services numériques existent pour accompagner le développement du vélo en ville. Un calculateur d'itinéraires adaptés au vélo serait un outil pratique. Il peut prendre la forme d'un site web pour préparer à l'avance ses trajets ou bien être sous la forme d'applications mobiles pour avoir l'information en temps réel, voire même directement sur son vélo.

Un calculateur d'itinéraires adaptés au vélo est un service inexistant en France car complexe à mettre en place. En effet, les critères à prendre en compte sont moins évidents que pour les autres modes de transport, telle que la voiture, et nécessitent des données comme les aménagements cyclables qui ne sont pas toujours disponibles dans les bases de données routières. Ainsi, ce travail de thèse s'est porté sur deux critères essentiels que sont la **distance** et la **sécurité** d'un itinéraire. Dans le chapitre 5, le critère d'effort simplifié est également intégré à nos méthodes de calcul pour réaliser des tests avec trois critères. Enfin, dans le chapitre 6, les critères de durée, de linéarité et d'autres éléments complexes, comme les manœuvres interdites sont abordés afin d'améliorer le calculateur d'itinéraires.

Chapitre 3

Présentation et modélisation du problème

Dans le chapitre précédent (cf. chapitre 2), le contexte général de la thèse a été présenté. Nous avons vu l'intérêt d'une centrale de mobilité vélo et l'importance des différents critères entrant en jeu dans le choix d'un itinéraire adapté aux cyclistes.

Dans ce chapitre, nous nous intéressons à la modélisation et au contexte scientifique de ce problème. Ce travail de thèse s'inscrit dans le domaine vaste des problèmes de cheminement. La section 3.1 introduit le problème de plus court chemin ainsi que les méthodes de résolution classiques et avancées de la littérature. L'une des principales difficultés de ce travail de thèse repose sur les différents critères à prendre en compte dans le calcul de plus court chemin. La section 3.2 présente ainsi les problèmes d'optimisation multiobjectif. Enfin, la section 3.3 explique la modélisation retenue du problème et la problématique de cette thèse.

Données relatives au problème multiobjectif :	
S :	l'ensemble des solutions représentées dans l'espace des objectifs
$\mathcal{PF}*$:	le front de Pareto représenté dans l'espace des objectifs
x :	une solution de l'ensemble S
x_I :	la solution idéale
x_N :	la solution nadir

Données concernant le graphe G :	
G :	le graphe
V :	l'ensemble des nœuds
A :	l'ensemble des arcs
K :	l'ensemble des objectifs
$n = \|V\|$:	le nombre de nœuds
$m = \|A\|$:	le nombre d'arcs
$q = \|K\|$:	le nombre d'objectifs
s :	le nœud de départ
t :	le nœud de destination
p :	un chemin
$c(p)$:	le vecteur de coûts associé au chemin p
$c^k(p)$:	le coût selon l'objectif k associé au chemin p
(i, j) :	l'arc reliant le nœud i au nœud j
c_{ij} :	le vecteur de coûts associé à l'arc (i, j)

TABLE 3.1 – Table des notations du chapitre 2

3.1 Problèmes de cheminement

Dans cette section, la modélisation d'un réseau routier sous la forme d'un graphe est présentée. Le problème de plus court chemin est ensuite introduit, ainsi que les méthodes classiques de résolution de ce problème.

3.1.1 Définition d'un graphe

Un graphe $G = (V, A)$ est défini par un ensemble V de $n = |V|$ nœuds et un ensemble A de $m = |A|$ arcs ou d'arêtes, défini par $A \subseteq V \times V$. L'ensemble A correspond intuitivement à des liens possibles pour passer d'un nœud à un autre. Par exemple, l'arc $(i, j) \in A$ modélise le fait de pouvoir passer du nœud i au nœud j.

Si A correspond à un ensemble d'arêtes, le graphe est dit non-orienté. Une arête (i, j) signifie alors qu'il est possible de passer de i vers j comme de j vers i.

Si A correspond à un ensemble d'arcs, le graphe est dit orienté. Un arc (i, j) signifie alors qu'il est possible de passer uniquement de i vers j.

En plus de la notion de lien entre les nœuds d'un graphe, il est courant d'affecter un poids, ou coût, sur les arcs. On parle alors de graphe valué. Ainsi, dans un graphe $G = (V, A, c)$, la fonction de coût c associe, pour chaque arc (i, j) reliant le nœud i au nœud j, un coût $c(i, j) = c_{ij}$.

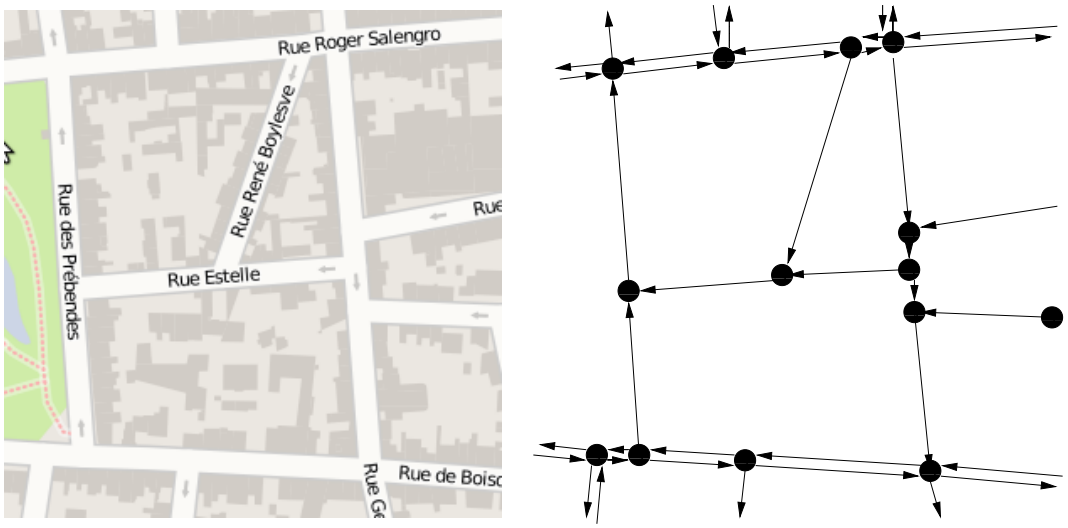


FIGURE 3.1 – Réseau routier modélisé par un graphe

Pour illustrer ces notions, un réseau routier peut être, comme sur la figure 3.1, modélisé par un graphe où les nœuds correspondent aux intersections des routes et les arcs correspondent aux tronçons de route entre les intersections. Il s'agit d'un graphe orienté puisque la circulation peut être en sens unique, et la valuation des arcs peut correspondre, par exemple, à la longueur en mètres des tronçons de route. Nous reviendrons sur cette modélisation dans la section 3.3.1.

3.1.2 Problème du plus court chemin

Dans un graphe $G = (V, A, c)$, un chemin p est défini par un ensemble ordonné de nœuds $p = \langle v_1, v_2, ..., v_l \rangle$ avec $p \subset V$ et $l = |p|$ la longueur du chemin p. Pour chaque couple de nœuds consécutifs i et j du chemin p, il existe un arc (i, j) reliant ces deux nœuds.

Le premier nœud v_1 du chemin p est appelé nœud *source* et le dernier nœud v_l est appelé nœud *puits*. Dans les problèmes de plus court chemin, le nœud *source* est souvent noté s, pour *start*, et le nœud *puits* est noté t, pour *target*.

Si le graphe est valué, le coût du chemin p correspond à la somme des coûts des arcs traversés : $c(p) = \sum_{i=1}^{l-1} c_{v_i, v_{i+1}}$

Le plus court chemin p entre deux nœuds s et t est le chemin qui minimise le coût $c(p)$ parmi l'ensemble des chemins possibles reliant s à t.

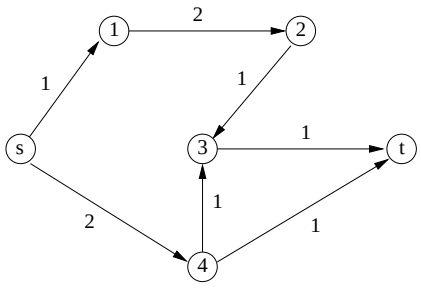


FIGURE 3.2 – Chemins dans un graphe

Sur le graphe de la figure 3.2, plusieurs chemins permettent de relier le nœud s jusqu'au nœud t :

- $p_1 = \langle s, 1, 2, 3, t \rangle$ de coût $c(p_1) = 5$
- $p_2 = \langle s, 4, 3, t \rangle$ de coût $c(p_2) = 4$
- $p_3 = \langle s, 4, t \rangle$ de coût $c(p_3) = 3$

Le plus court chemin de s à t minimisant la somme des coûts des arcs est donc le chemin p_3 de coût $c(p_3) = 3$.

3.1.3 Algorithmes du plus court chemin

L'algorithme de Dijkstra

L'algorithme de Dijkstra [Dijkstra 59] permet de calculer le plus court chemin d'un nœud vers tous les autres nœuds du graphe. Son principe est présenté dans l'algorithme 1. Une particularité de cet algorithme est qu'il ne fonctionne que sur les graphes valués avec des coûts positifs c'est-à-dire avec une fonction de coût de type $c : A \to \mathbb{R}^+$ qui associe à chaque arc un coût positif ou nul.

Algorithme 1 Algorithme de Dijkstra

1: $Q \leftarrow V$ *// la file Q contient tous les nœuds à traiter*
2: $distance_i = +\infty, i = \{1, ..., n\} \setminus \{s\}$
3: $arc_i = \phi, i = \{1, ..., n\}$
4: $distance_s = 0$
5: **tant que** $Q \neq \phi$ **faire** *// tant qu'il reste des nœuds à traiter*
6: $\quad i \leftarrow Q.Min()$ *// sélectionner le nœud de plus petite distance*
7: $\quad Q \leftarrow Q \setminus i$
8: $\quad$ **pour tout** $(i, j) \in A$ **faire** *// Pour chaque arc sortant du nœud i*
9: $\quad\quad$ **si** $distance_i + c_{ij} < distance_j$ **alors**
10: $\quad\quad\quad distance_j \leftarrow distance_i + c_{ij}$
11: $\quad\quad\quad arc_j \leftarrow (i, j)$
12: $\quad\quad$ **fin si**
13: $\quad$ **fin pour**
14: **fin tant que**

Il s'agit d'un algorithme dit d'étiquetage (*labelling*) : une étiquette $distance_i$ est associée à chaque nœud $i \in V$ et contient la valeur de coût minimal correspondant au plus court chemin entre le nœud source s et le nœud i. Au début de l'algorithme, toutes les étiquettes sont initialisées à $+\infty$ sauf l'étiquette du nœud s qui prend la valeur 0.

La file d'exploration Q contient les nœuds à traiter. Initialement, Q contient tous les nœuds de V. Le principe de l'algorithme de Dijkstra est ensuite de retirer les nœuds un à un de la file Q, en prenant soin de toujours prendre le nœud i de plus petite étiquette $distance_i$ (ligne 6), et d'étendre cette étiquette, c'est-à-dire de mettre à jour les étiquettes des nœuds successeurs si la valeur de leur étiquette est améliorée (ligne 8 à 13).

L'algorithme de Dijkstra est un algorithme dit de *label-setting* car il respecte la propriété suivante : à chaque fois qu'un nœud i est retiré de la file Q, la valeur de son étiquette est définitive et correspond au plus court chemin de s à i. Les algorithmes de *label-setting* sont opposés aux algorithmes de *label-correcting* qui sont également des algorithmes de *labelling* mais qui ne respectent pas cette condition.

L'algorithme de Dijkstra calcule les plus courts chemins d'un nœud *source* s vers tous les autres nœuds, ce qui correspond à l'arbre des plus courts chemins. Cependant, si l'objectif initial est de calculer uniquement le plus court chemin entre deux nœuds, comme de s vers t, il est possible d'arrêter l'algorithme dès que le nœud t est retiré de la file Q. En effet, comme il s'agit d'un algorithme de *label-setting*, lorsque t est retiré de la file Q, nous sommes alors certains que son étiquette associée correspond au coût du plus court chemin de s à t.

Dans l'algorithme 1, la variable arc_j d'un nœud j permet de garder en mémoire l'arc précédent (i, j) de manière à pouvoir reconstruire le chemin complet pour arriver de s à j.

La complexité de l'algorithme de Dijkstra est en $O(n^2)$. Pourtant, l'utilisation de structures de données spécifiques pour la gestion de la file Q, tel que le tas de Fibonnacci [Fredman 87] permet de passer à une complexité en $O(m + n \cdot \log n)$.

L'algorithme de Bellman-Ford

L'algorithme de Bellman-Ford [Bellman 58, Ford 56] permet également de calculer le plus court chemin d'un nœud vers tous les autres nœuds du graphe. Son principe est présenté dans l'algorithme 2. Cet algorithme se différencie de celui de Dijkstra par la possibilité d'avoir des arcs de coût négatif dans le graphe. De plus, il permet de détecter les circuits absorbants, c'est-à-dire des cycles de coût total négatif.

Algorithme 2 Algorithme de Bellman-Ford

1: $Q \leftarrow V$ *// la file Q contient tous les nœuds à traiter*
2: $distance_i = +\infty, i = \{1, ..., n\} \setminus \{s\}$
3: $arc_i = \phi, i = \{1, ..., n\}$
4: $distance_s = 0$
5: **pour** $k = 1$ à $|V|$ **faire**
6: **pour tout** $(i, j) \in A$ **faire** *// Pour chaque arc de A*
7: **si** $distance_i + c_{ij} < distance_j$ **alors**
8: $distance_j \leftarrow distance_i + c_{ij}$
9: $arc_j \leftarrow (i, j)$
10: **fin si**
11: **fin pour**
12: **fin pour**
13: **pour tout** $(i, j) \in A$ **faire** *// Pour chaque arc de A*
14: **si** $distance_j < distance_i + c_{ij}$ **alors**
15: Cycle absorbant détecté
16: **fin si**
17: **fin pour**

Contrairement à l'algorithme de Dijkstra qui sélectionne les nœuds un par un et étend une étiquette aux nœuds successeurs, l'algorithme de Bellman-Ford étend à chaque itération l'ensemble des étiquettes en utilisant tous les arcs. Il s'agit ici d'un algorithme de *label-correcting* puisqu'une étiquette peut être sélectionnée et améliorée à plusieurs reprises avant d'arriver à sa valeur optimale. L'algorithme itère n fois ce qui correspond à une borne maximale du nombre d'arcs composant le plus court chemin.

Dans l'algorithme 2, la dernière boucle ligne 13 permet de détecter si un circuit absorbant est présent ou non.

La complexité de cet algorithme est en $O(n \cdot m)$, donc supérieure à celle de l'algorithme de Dijkstra. De plus, comme nous le verrons ultérieurement, dans les problèmes abordés dans cette thèse, les coûts associés aux arcs sont positifs ou nuls. Cet algorithme n'est donc pas le plus adapté à nos problématiques.

Recherche bi-directionnelle

Une amélioration classique pour réduire le nombre d'étiquettes traitées dans une méthode de *labelling* est de lancer deux recherches simultanées :

– l'une partant du nœud *source* s,

– l'autre du nœud *puits* t avec les arcs inversés.

Le plus court chemin est trouvé lorsqu'un nœud a été traité par les deux recherches.

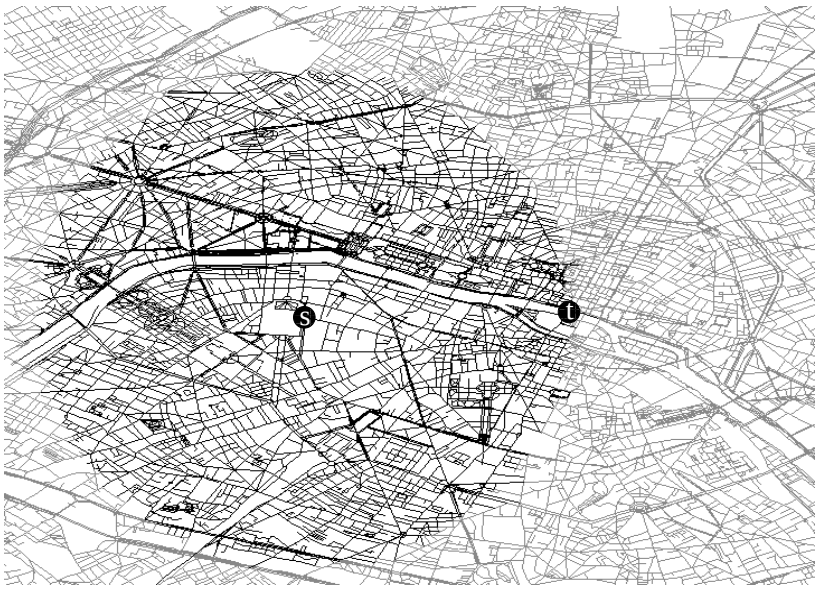


FIGURE 3.3 – Zone explorée par une recherche avec Dijkstra

La figure 3.4 montre schématiquement dans un graphe routier les nœuds traités par chacune des deux recherches. Au final, le nombre de nœuds traités par les deux recherches est inférieur à une recherche mono-directionnelle, de type algorithme de Dijkstra, illustrée sur la figure 3.3.

Notons que les deux recherches ne sont pas guidées et perdent du temps à traiter des nœuds qui ne sont pas pertinents.

Méthode A*

La méthode A* a été introduite par [Hart 68] et repose sur l'utilisation d'une heuristique pour guider la recherche jusqu'au nœud *puits*. Cette heuristique associe, pour chaque nœud i du graphe, une borne inférieure sur le coût du plus court chemin de i jusqu'au nœud *puits* t. A chaque itération, la méthode ne sélectionne plus le nœud i de plus petite étiquette, mais le nœud pour lequel la somme de l'étiquette $distance_i$ et de l'heuristique $heuristique_i$ est la plus petite. Cette valeur correspond à une borne inférieure du coût du plus court chemin de s à t passant par le nœud i.

Dans le cas des graphes routiers, une heuristique classique pour estimer la distance entre deux nœuds est d'utiliser la distance à vol d'oiseau.

La figure 3.5 montre le même exemple schématique que précédemment, mais avec une recherche utilisant la méthode A*. L'heuristique permet ici d'orienter la recherche vers

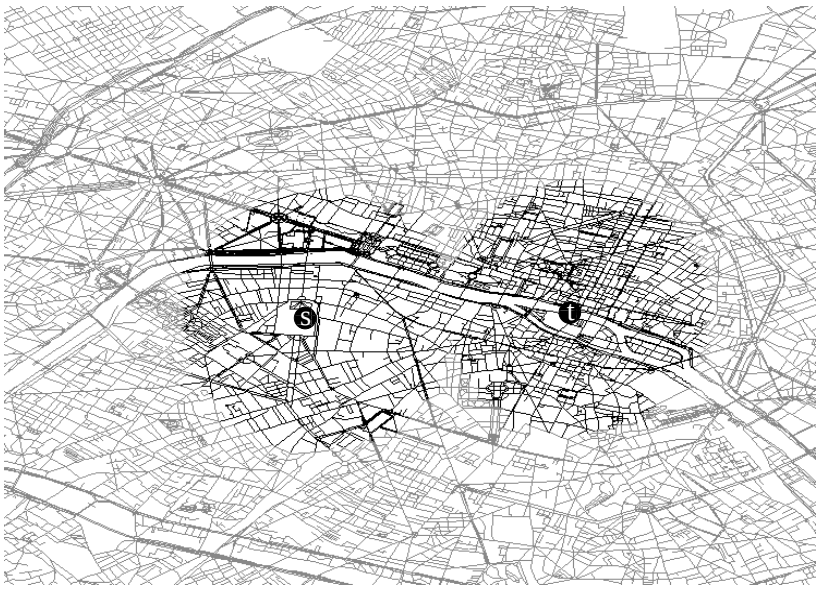


FIGURE 3.4 – Zone explorée par une recherche bidirectionnelle

FIGURE 3.5 – Zone explorée par une recherche avec A*

le nœud t et d'éviter de traiter des nœuds qui potentiellement s'éloignent du nœud t. Bien entendu, le résultat dépend fortement de la qualité de l'heuristique. Ici, la distance à vol d'oiseau est perfectible. En effet, le plus court chemin entre s et t peut nécessiter de s'éloigner du nœud t : par exemple dans le cas d'un détour pour accéder à un pont dans un graphe routier.

À partir de ces méthodes classiques (Dijkstra, Bi-directionnelle, A*, etc.), de nombreuses méthodes améliorant très significativement les performances de ces algorithmes ont été proposées dans la littérature pour résoudre le problème de plus court chemin. Nous présentons brièvement par la suite certaines de ces méthodes parmi les plus efficaces.

Méthodes hiérarchiques

Ces méthodes exploitent le concept de hiérarchie dans le graphe et tous les nœuds ou arcs ne se situent donc pas au même niveau. Ces techniques nécessitent souvent une phase de prétraitement importante.

La méthode Highway Hierarchies [Sanders 06] regroupe les nœuds et arcs dans plusieurs niveaux. Une étape de contraction permet de passer d'un graphe de niveau i à un graphe de niveau $i + 1$ de plus petite taille. Pour effectuer un calcul de plus court chemin de s à t, une recherche bidirectionnelle est lancée en partant du graphe du niveau le plus bas (graphe complet). L'idée est que lorsque la recherche lancée depuis s (respectivement t) est suffisamment éloignée de ce nœud, des arcs du graphe ne sont plus accessibles, obligeant à passer au niveau supérieur, jusqu'à ce que les deux recherches se rencontrent. Il s'agit de la première méthode permettant de calculer des plus courts chemins exacts sur des graphes de grande taille (États-Unis, Europe) avec des temps de calcul mesurables en millisecondes.

Dans la méthode Highway-Node Routing [Schultes 07], les nœuds sont regroupés arbitrairement par une heuristique dans différentes classes et des raccourcis sont créés entre les classes. La méthode Contraction Hierarchies [Geisberger 08] est un cas particulier de la méthode Highway-Node Routing dans laquelle chaque nœud est dans une classe différente. Cette méthode simplifie le calcul d'un plus court chemin, car la recherche bidirectionnelle s'effectue en autorisant uniquement de passer à des nœuds appartenant à des niveaux supérieurs.

La méthode Transit-Node Routing [Bast 07] repose sur le fait que dans les graphes représentant des réseaux routiers, il existe un petit ensemble de nœuds, appelés nœuds de *transit*, pour lesquels le plus court chemin de n'importe quelle paire de nœuds suffisamment éloignés passent par au moins un nœud de cet ensemble. De plus, pour chacun des nœuds, le nombre de nœuds de *transit* rencontrés en premier, appelés nœud d'*accès*, pour des plus courts chemins partant de ce nœud, est faible. La méthode calcule en prétraitement des distances entre tous les nœuds et leurs nœuds d'*accès* et entre tous les nœuds de *transit*. Ce prétraitement permet alors de calculer un plus court chemin très rapidement en utilisant uniquement les distances calculées en prétraitement.

Méthodes avancées pour orienter la recherche

La méthode ALT [Goldberg 05b] est basée sur la méthode A* mais propose un prétraitement sur le graphe pour calculer une heuristique de meilleure qualité. Ce prétraitement consiste à sélectionner un ensemble de nœuds $L \in V$ appelés *landmarks*. Ensuite, pour chacun des nœuds appartenant à l'ensemble des *landmarks*, tous les plus courts chemins sont calculés entre ce nœud et tous les autres nœuds du graphe. Ces informations sont gardées en mémoire et permettent, grâce à l'utilisation de l'inégalité triangulaire, d'améliorer l'heuristique dans la méthode de recherche A*.

La méthode Edge-flags [Hilger 09], ou marquage d'arcs, consiste à calculer en prétraitement des informations positionnées sur les arcs pour pouvoir guider la recherche et éviter de parcourir des arcs inutiles. Pour cela le graphe est découpé en régions, à la manière d'un simple quadrillage ou de manière plus élaborée pour obtenir des régions équilibrées en termes de répartition du nombre d'arcs par région. Ensuite, le prétraitement consiste à déterminer pour chaque arc et pour chaque région, s'il existe un plus court chemin passant par cet arc pour aller dans un nœud de cette région. Chacune de ces informations est un marqueur positionné sur les arcs, d'où le nom de marquage d'arcs.

La méthode SHARC [Bauer 10] est une extension de la méthode Edge-flags intégrant en plus une approche hiérarchique. Plus généralement, toutes ces méthodes peuvent se combiner entre elles pour tirer parti des avantages de chacune. Parmi ces méthodes et leurs combinaisons, il n'existe pas réellement de méthodes plus performantes les unes que les autres, étant donné qu'elles présentent toutes un compromis différent entre temps de calcul du prétraitement, occupation mémoire et temps de calcul des plus courts chemins.

Notons que ces méthodes s'accompagnent toujours de l'utilisation de structures de données spécifiques et non-triviales, permettant une implémentation efficace.

3.2 Optimisation multiobjectif

La modélisation d'un problème d'optimisation intégrant un seul critère n'est pas toujours suffisante. Par exemple, prendre en compte uniquement la distance ou bien la sécurité de l'itinéraire pour calculer un chemin adapté aux cyclistes ne sera pas toujours satisfaisant. En effet, entre un itinéraire d'un kilomètre très dangereux et un itinéraire de trente kilomètres très sécurisé, un cycliste peut préférer choisir des itinéraires intermédiaires. La prise en compte simultanée de plusieurs objectifs (ou critères) dans la résolution d'un problème d'optimisation permet de calculer des solutions de compromis et l'on parle alors d'optimisation multiobjectif (ou multicritère).

3.2.1 Définitions

Nous présentons ici les concepts fondamentaux de l'optimisation multiobjectif. Nous renvoyons à [Ehrgott 00] pour un état de l'art complet sur les problèmes d'optimisation multiobjectif.

Problème d'optimisation multiobjectif

Soit p une solution, comme un chemin par exemple, d'un problème d'optimisation multiobjectif. Notons $x = c(p)$ la représentation de cette solution dans l'espace des objectifs avec $x \in \mathbb{R}^q$ un vecteur objectif correspondant aux coûts de la solution p avec $q = |K|$ le nombre d'objectifs du problème.

Dans les problèmes multiobjectif, il est souvent plus pratique de travailler dans l'espace des objectifs et même de confondre par abus de langage une solution p avec sa représentation x dans l'espace des objectifs. Par la suite, une solution x désignera donc une solution dont le vecteur de coûts des objectifs est x.

Si nous considérons que l'on souhaite minimiser chacune des valeurs des objectifs, on parle alors de coûts. La maximisation des valeurs des objectifs, que l'on appelle gains, peut se transformer sous la forme d'un problème de minimisation. Par la suite, nous nous placerons donc toujours dans un problème de minimisation des coûts. Notons S l'ensemble des solutions réalisables dans l'espace des objectifs. Lorsque le nombre d'objectifs q est égal à deux, il est possible de représenter l'ensemble S sur un repère où les axes des abscisses et des ordonnées correspondent à chacun des deux objectifs, comme sur la figure 3.6.

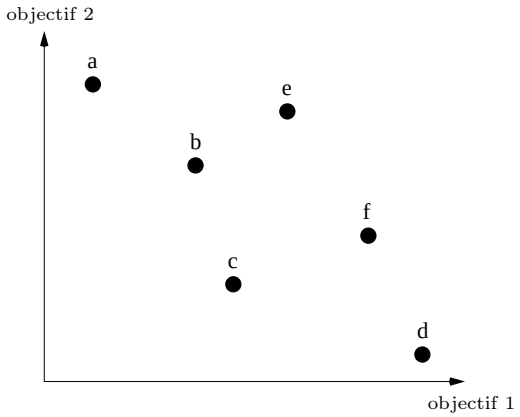


FIGURE 3.6 – Représentation des solutions S dans l'espace des objectifs

Parmi les solutions S, certaines solutions sont plus intéressantes que d'autres. Par exemple sur la figure 3.6, la solution e est moins bonne que la solution b pour chacun des deux objectifs. Mais, en considérant les deux objectifs simultanément, il n'est pas possible de déterminer qui de a ou de b est meilleure. En effet, a est meilleure que b sur l'objectif 1, mais b est meilleure que a sur l'objectif 2.

La dominance de Pareto permet de représenter ces relations entre les vecteurs de coûts des solutions :

Définition 1. *Une solution $x_1 \in S$ est dite faiblement Pareto-dominée par une solution $x_2 \in S$ si et seulement si $c^k(x_2) \leq c^k(x_1)$, $\forall k \in K$*

Définition 2. *Une solution $x_1 \in S$ est dite Pareto-dominée par une solution $x_2 \in S$ si et seulement si $c^k(x_2) \leq c^k(x_1)$, $\forall k \in K$ et $\exists k \in K$ tel que $c^k(x_2) < c^k(x_1)$*

Définition 3. *Une solution $x_1 \in S$ est dite strictement Pareto-dominée par une solution $x_2 \in S$ si et seulement si $c^k(x_2) < c^k(x_1)$, $\forall k \in K$*

Par exemple, sur la figure 3.6, les solutions e et f sont strictement Pareto-dominées par la solution c.

Dans la suite de cette thèse, nous utiliserons l'écriture $x_2 \prec_p x_1$ pour signifier que la solution x_1 est Pareto-dominée par la solution x_2.

Solutions d'un problème d'optimisation multiobjectif

L'ensemble des solutions optimales d'un problème d'optimisation multiobjectif est représenté par $\mathcal{PF}* \subset \mathcal{S}$ appelé ensemble Pareto-optimal. Cet ensemble est défini de façon qu'il n'existe aucune solution Pareto-dominée par une autre solution de S. Cet ensemble est également appelé front de Pareto et est représenté sur la figure 3.7. Ces solutions sont dites non-Pareto-dominées (ou efficaces). Par la suite, nous parlerons de solutions non-dominées pour non-Pareto-dominées et de solutions dominées pour Pareto-dominées.

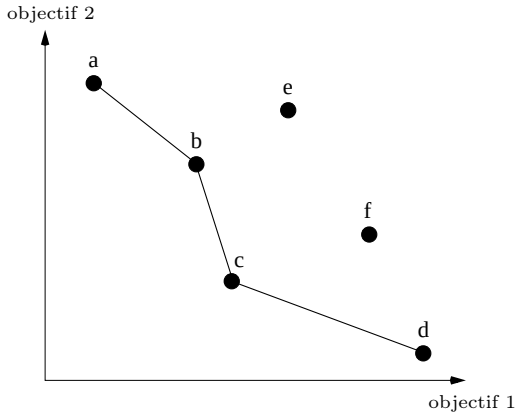


FIGURE 3.7 – Représentation du front de Pareto

Parmi les solutions non-dominées, une distinction est souvent faite entre les solutions appartenant ou non à l'enveloppe convexe du front de Pareto dans l'espace des objectifs :

- les solutions dites supportées qui sont sur l'enveloppe convexe : il s'agit des solutions a, c et d sur la figure 3.7,
- les solutions dites non-supportées qui ne sont pas sur l'enveloppe convexe : il s'agit de la solution b sur la figure 3.7.

Cette distinction existe car il s'agit de deux catégories différentes de solutions. Une solution supportée x peut être obtenue en optimisant une combinaison linéaire des objectifs $CL = \sum_{k=1}^{q} \lambda^k c^k$, tq $\sum_{k=1}^{q} \lambda^k = 1$ en choisissant la bonne valeur de λ. Cela revient à transformer le problème multiobjectif en problème mono-objectif.

Parmi toutes les solutions non-dominées $\mathcal{PF}*$ il peut être intéressant de connaitre les valeurs minimales et maximales de chacun des objectifs. Pour cela les points *idéal* x_I et

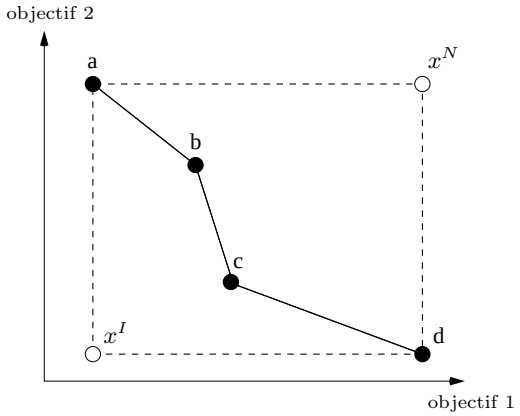


FIGURE 3.8 – Représentation du point idéal et du point nadir

nadir x_N sont définis comme sur la figure 3.8 pour borner les valeurs des objectifs. Ces points servent uniquement de références et ne correspondent pas à des solutions réalisables.

Définition 4. *Parmi l'ensemble Pareto-optimal* $\mathcal{PF}*$*, le point idéal* x_I *est la solution vérifiant :* $x_I^k = \min_{x \in \mathcal{PF}*} x^k$, $\forall k \in K$

Définition 5. *Parmi l'ensemble Pareto-optimal* $\mathcal{PF}*$*, le point nadir* x_N *est la solution vérifiant :* $x_N^k = \max_{x \in \mathcal{PF}*} x^k$, $\forall k \in K$

3.2.2 Différentes problématiques

Nous avons vu que dans un problème d'optimisation multiobjectif, l'ensemble des solutions non-dominées $\mathcal{PF}*$ sont potentiellement intéressantes et susceptibles d'intéresser le décideur. Cependant, le décideur possède ses propres préférences et il doit intervenir dans la sélection des solutions. Il existe différentes approches de résolution en fonction du moment où intervient le décideur :

- avant la méthode de résolution : approches *a priori*,
- pendant la méthode de résolution : approches *interactive*,
- après la méthode de résolution : approches *a posteriori*.

Approches a posteriori

Les approches *a posteriori* consistent à énumérer l'ensemble du front de Pareto $\mathcal{PF}*$, sans intégrer les préférences du décideur. Toutes les solutions de compromis sont ensuite présentées au décideur qui peut faire sa sélection selon ses préférences (implicites ou explicites). L'avantage de ces approches est que toute la phase de calcul ne nécessite pas l'intervention du décideur.

Ces approches peuvent également être utilisées en prétraitement des approches *a priori*. Il peut être alors aisé de déterminer la ou les solutions de compromis satisfaisant les préférences du décideur parmi la totalité des solutions de compromis $\mathcal{PF}*$.

Le principal inconvénient de ce type d'approches vient du nombre potentiellement important de solutions de compromis sur le front de Pareto $\mathcal{PF}*$. En effet, pour le problème de plus court chemin multiobjectif, le graphe proposé par [Hansen 80] montre que le nombre de solutions non-dominées peut croitre de façon exponentielle suivant le nombre de nœuds. Il a également été montré dans [Rosinger 91] que le nombre de solutions non-dominées était très fortement corrélé avec le nombre d'objectifs. Ces approches ne peuvent donc pas être utilisées sur des problèmes de taille trop importante ou bien lorsqu'il existe une contrainte forte sur le temps de calcul.

Les approches *a posteriori* sont traitées dans le chapitre 4 où un état de l'art et plusieurs voies d'améliorations y sont présentés.

Approches a priori

Dans les approches *a priori*, les préférences du décideur sont connues à l'avance. Ces préférences peuvent être définies, par exemple, sous la forme d'un vecteur de poids associant une importance à chacun des objectifs.

Ces approches consistent à optimiser une fonction qui agrège l'ensemble des objectifs suivant les préférences du décideur. Le résultat de ces approches n'est plus l'ensemble du front de Pareto $\mathcal{PF}*$ mais une solution de meilleur compromis qui appartient à cet ensemble. Le fait de se concentrer sur la détermination de cette unique solution permet d'éviter l'énumération de toutes les solutions de compromis et donc, bien souvent, de réduire le temps de calcul.

Une partie importante de la thèse [Galand 08] est dédiée à ces approches où l'auteur les applique notamment au problème de plus court chemin multiobjectif. Dans [Futtersack 00], la méthode BCA*, pour Best Compromise A*, permet de calculer une solution de meilleur compromis pour le problème de plus court chemin multiobjectif, en orientant la recherche de manière à explorer en premier les étiquettes potentiellement meilleures d'un point de vue des préférences du décideur.

Cette approche est utilisée dans le chapitre 5 où un état de l'art plus complet y est présenté.

Approches interactives

Les approches interactives consistent à déterminer une solution de compromis en fonction des préférences du décideur qui vont s'affiner pendant la méthode de résolution. Pour cela, deux phases sont répétées alternativement jusqu'à l'obtention d'une solution de compromis satisfaisant le décideur :

– la phase de calcul d'une ou plusieurs solutions de compromis,

– la phase de dialogue avec le décideur où ce dernier fait évoluer ses préférences en fonction des solutions qui lui sont proposées.

Il existe principalement deux façons de réaliser la phase de calcul :

– calculer une seule fois en prétraitement l'ensemble des solutions de compromis $\mathcal{PF}*$ et limiter la phase de calcul à la sélection de la ou les solutions de meilleur compromis parmi l'ensemble $\mathcal{PF}*$,

– calculer directement à chaque itération la ou les solutions de meilleur compromis.

Cette dernière approche est mise en œuvre dans [Galand 08] où la méthode prend en compte, à une itération courante, des solutions trouvées aux itérations précédentes pour accélérer le calcul de la nouvelle solution de compromis.

Cette approche n'est pas directement abordée dans cette thèse, bien qu'une interface a été développée (cf. chapitre 7) pour permettre de tester une approche interactive.

3.3 Modélisation et problématique

Dans cette section, la modélisation du réseau routier ainsi que les critères retenus dans cette thèse (cf. chapitres 4 et 5) sont introduits. Ensuite, un état de l'art général sur le problème du plus court chemin multiobjectif est présenté. Enfin, la problématique et les objectifs de la thèse sont définis.

3.3.1 Modélisation du réseau routier

Un graphe $G = (V, A)$ est particulièrement adapté à la représentation d'un réseau routier. Dans la représentation la plus classique, les intersections entre les routes peuvent être vues comme les nœuds V du graphe et les tronçons de route d'intersection à intersection sont alors les arcs A du graphe. Un graphe orienté permet de modéliser facilement les sens de circulation du réseau routier. Par exemple, entre deux nœuds i et j, un seul arc (i, j) modélise un sens unique de i vers j alors que deux arcs (i, j) et (j, i) modélisent le fait qu'il est possible de circuler dans les deux sens.

Des coûts sont très souvent associés aux arcs. Par exemple, il est courant de considérer les coûts de temps et de distance. Notons c^1 et c^2 les fonctions associant à chaque arc du graphe un coût positif ou nul. La distance peut facilement se calculer en connaissant la longueur de chaque tronçon de route, et le temps se calcule par exemple à partir de la vitesse moyenne [1] et de la longueur des tronçons de route. Cette modélisation simple, où les nœuds correspondent aux intersections et les arcs correspondent aux tronçons de route, est la modélisation retenue pour la présentation des différentes méthodes proposées, et leur test de performance, abordés dans les chapitres 4 et 5.

1. Le choix des vitesses moyennes est tout de même complexe car elles peuvent varier en fonction du profil du cycliste, de ses envies, des voies empruntées, etc.

Cependant, cette modélisation du réseau routier n'est pas toujours suffisante. Pour commencer, il peut être nécessaire d'ajouter des coûts sur les nœuds. Dans l'exemple avec les coûts de distance et de temps, le temps dépend fortement des intersections, c'est-à-dire des nœuds du graphe : le temps d'attente moyen à un feu tricolore sera bien supérieur au temps nécessaire pour traverser une intersection classique. De plus, dans le calcul de chemins dans un réseau routier, il faut être en mesure de pouvoir interdire des enchaînements d'arcs modélisant des manœuvres interdites à certaines intersections. Pour terminer, il peut être intéressant d'associer des coûts sur des enchaînements d'arcs : par exemple un cycliste préfère réduire le nombre de changements de direction et il est utile d'affecter un coût à ces derniers. Toutes ces problématiques sont abordées dans le chapitre 6 qui traite de la modélisation avancée d'un réseau routier sous la forme d'un graphe adjoint [Winter 02].

3.3.2 Définition des objectifs

Comme nous l'avons vu dans le chapitre 2, les critères les plus importants dans le choix d'une itinéraire adapté aux cyclistes sont la distance et la sécurité. Par la suite, il s'agit des deux critères principaux considérés dans cette thèse. Cependant, un critère d'effort sera ajouté au chapitre 5 pour évaluer nos méthodes avec trois critères. De même, le critère de linéarité est abordé dans le chapitre 6.

Définissons deux fonctions de coût non négatives c^1 pour la distance et c^2 pour l'insécurité (prendre en compte la sécurité revient à définir des coûts d'insécurité) associant à chaque arc deux coûts $c^1 : A \to \mathbb{R}^+$ et $c^2 : A \to \mathbb{R}^+$. Nous noterons c^1_{ij} et c^2_{ij} les coûts selon l'objectif 1 et 2 de l'arc (i, j).

Le premier objectif est relativement simple puisqu'il consiste à minimiser la longueur totale, en mètres, d'un chemin, c'est-à-dire trouver le chemin p minimisant la somme des coûts c^1 de ses arcs : $\min c^1(p) = \min \sum_{i=1}^{l-1} c^1_{v_i,v_{i+1}}$.

Le deuxième objectif est plus complexe puisqu'il fait intervenir la notion de sécurité et de cyclabilité. Comme nous l'avons vu dans la section 2.2, ces notions prennent en compte les aménagements cyclables, le trafic automobile, la largeur des voies, etc. De façon similaire à [Ins 98], nous définissons la sécurité en plusieurs niveaux, en fonction de l'aménagement cyclable et de la cyclabilité. Les aménagements cyclables peuvent être : une piste cyclable, une bande cyclable, une voie mixte bus/vélo, une voie mixte piéton/vélo, un piste ou bande en contre sens cyclable. La cyclabilité est une note de 0 à 3 évaluée par un collecteur de terrain qui prend en compte des éléments différents en fonction de l'aménagement : trafic automobile/piéton estimé, largeur de la voie, stationnement gênant régulier, manque de visibilité, etc.

La combinaison de l'aménagement cyclable et de la cyclabilité permet de définir un coefficient d'insécurité sur chaque voie. Pour obtenir la note finale d'insécurité d'un arc, la longueur, en mètres, de l'arc est multipliée par ce coefficient d'insécurité. Cela signifie que l'insécurité d'un arc est proportionnel à la fois au temps que passe le cycliste sur cet arc et au coefficient d'insécurité de cet arc. Ce modèle est inspiré du modèle de transport des matériaux dangereux où le risque est minimisé.

Le deuxième objectif consiste donc à minimiser l'insécurité totale d'un chemin, c'est-à-dire trouver le chemin p minimisant la somme des coûts c^2 de ses arcs $\min c^2(p) = \min \sum_{i=1}^{l-1} c^2_{v_i,v_{i+1}}$. Une autre possibilité serait de minimiser le coefficient maximal d'insécurité d'un itinéraire, c'est-à-dire un objectif du type $\min c^2(p) = \min \max_{i=1}^{l-1} c^2_{v_i,v_{i+1}}$. Cette solution n'a pas été retenue car elle ne permet pas de maximiser l'utilisation des aménagements cyclables. Par exemple, cet objectif privilégierait un itinéraire de 10 km de bandes cyclables face à un itinéraire avec 10 km de pistes cyclables et 50 mètres de voies sans aménagement.

3.3.3 Problème de plus court chemin multiobjectif

Nous avons défini le graphe G et les deux objectifs que nous souhaitons prendre en compte dans le calcul des chemins. La principale difficulté de ce travail repose sur le caractère multiobjectif du problème. Le résultat de ce problème n'est pas un seul et unique chemin, mais un ensemble de chemins non-dominés, au sens de la relation de dominance de Pareto, qui correspondent à des compromis différents entre le critère de distance et de sécurité.

Le problème du plus court chemin multiobjectif (problèmes MOSP : MultiObjective Shortest Path) fait parti des problèmes d'optimisation combinatoire multicritère (problèmes MOCO : MultiObjective Combinatorial Optimization). Un état de l'art général sur ce type de problème peut être trouvé dans [Ehrgott 00, Ehrgott 02]. Dans ces papiers, les auteurs expliquent la nécessité de classifier la littérature autour des problèmes MOCO. En effet, d'une part il convient de spécifier le problème (plus court chemin, sac à dos, etc.), le nombre et le type d'objectifs (minimisation d'une somme, minimisation d'un maximum, etc.) ainsi que le type de problème (calculer l'ensemble des solutions, calculer un sous ensemble, calculer un meilleur compromis, etc.). D'autre part, il est également intéressant de préciser la méthodologie utilisée : il peut s'agir de méthodes exactes ou approchées. Dans ce dernier cas, les solutions trouvées peuvent ne pas être optimales. Les auteurs expliquent également la différence très importante entre les problèmes bi-objectif et les problèmes multiobjectif. En effet, en plus de la complexité de calcul qu'implique un nombre plus important de critères, la plupart des méthodes proposées pour le problème bi-objectif ont été développées spécifiquement pour ce dernier et ne peuvent pas, ou difficilement, être adaptées à plus de deux critères.

Un état de l'art récent sur les problèmes de plus court chemin multiobjectif est présenté dans [Tarapata 07, Climaco 10]. La littérature traitant de ce problème peut être classifiée suivant la même notation X/Y/Z que dans [Ehrgott 00] où X est le nombre et le type des objectifs, Y correspond au type de problème et Z est le type de méthode utilisée. Comme nous l'avons expliqué dans la section 3.3.2, nous nous intéressons uniquement aux objectifs de type « somme ». Nos deux objectifs correspondent donc à la distance et à l'insécurité que nous souhaitons minimiser. Dans ce cas précis, la détermination de l'ensemble des solutions non-dominées est NP-Difficile [Serafini 87]. Cependant, tous les problèmes de plus court chemin bi-objectif n'ont pas pour autant la même complexité combinatoire. En effet, il est montré dans [Pascoal 06] que le problème bi-objectif, où une fonction objectif est de

type Min-Somme et l'autre est de type Max-Min, peut être résolu en temps polynomial. Ce type de problème se retrouve souvent dans les problèmes de routage dans les réseaux où le deuxième objectif représente la maximisation de la bande passante.

3.3.4 Problématique de la thèse

La problématique de cette thèse est de répondre au besoin d'un cycliste qui désire connaître un chemin ou des chemins adaptés à ses besoins de cycliste entre deux nœuds du graphe. Deux sous-problèmes sont envisagés dans ce travail de thèse :

- le calcul de l'ensemble des chemins non-dominés (approche *a posteriori*) abordé dans le chapitre 4,
- le calcul d'une solution de meilleur compromis (approche *a priori*) abordé dans le chapitre 5.

Comme nous l'avons vu dans la section 3.2, chacune de ces approches a ses avantages et ses inconvénients. Calculer l'ensemble des chemins non-dominés permet de sélectionner et de ne proposer que les chemins les plus intéressants à l'utilisateur, mais peut potentiellement prendre plus de temps de calcul car il faut énumérer toutes les solutions. Par contre, se concentrer sur le calcul d'un seul chemin de meilleur compromis est plus rapide mais nécessite de connaitre les préférences de l'utilisateur (compromis entre distance et sécurité), voire également son profil (récréatif, utilitaire, etc.).

Il est important de garder à l'esprit que l'objectif est bien de pouvoir intégrer nos méthodes à un calculateur sur un site internet ou pour des applications mobiles. Le temps de calcul est donc un critère très important dans l'évaluation de nos méthodes. Dans cette thèse, nous fixons cette limite de temps à environ 3 secondes. Face à l'importance du temps de calcul, nous verrons que les méthodes existantes de la littérature ont besoin d'être accélérées pour pouvoir être intégrées sur une plate-forme web.

Notons que les méthodes présentées aux chapitres 4 et 5, prenant en compte distance et sécurité, sont génériques et donc adaptables à d'autres types d'objectifs de type somme. Les méthodes vues au chapitre 5 peuvent également être étendues à plus de deux objectifs, comme le montrent les expérimentations réalisées dans le chapitre 6.

3.4 Conclusion du chapitre

Dans ce chapitre, le problème de plus court chemin mono-objectif a été introduit. Il s'agit d'un problème largement étudié et depuis [Dijkstra 59], de nombreuses méthodes classiques et avancées permettent de calculer des chemins sur des graphes routiers de très grande taille.

Ensuite, les problèmes d'optimisation multiobjectif ont été présentés. La principale difficulté est liée au nombre de solutions dites de compromis qui peuvent être très nombreuses. En plus de la complexité combinatoire qu'implique le calcul de l'ensemble de ces solutions, il n'est pas évident pour le décideur de faire un choix parmi toutes ces solutions. Différentes approches, *a priori*, *interactive* et *a posteriori*, permettent d'aborder différemment ces problèmes, de manière à aider le décideur à ne retenir qu'une seule, ou un sous-ensemble de solutions.

Enfin, la modélisation du graphe routier et des coûts a été abordée. Pour modéliser des critères complexes comme le temps, l'effort ou certains aspects de la sécurité, nous avons montré qu'une modélisation classique des coûts peut ne pas être suffisante. La représentation du réseau routier sous la forme d'un graphe adjoint a été introduite pour tenir compte de coûts sur les arcs, sur les nœuds et sur des enchaînements d'arcs. Cet aspect sera développé dans le chapitre 6.

Dans le chapitre 4, l'approche *a posteriori* sera privilégiée en prenant en compte uniquement les critères de distance et de sécurité. Dans le chapitre 5, l'approche *a priori* sera étudiée pour se concentrer sur une seule solution de meilleur compromis et prenant en compte plus de deux critères.

Chapitre 4

Détermination totale du front de Pareto pour le problème de plus court chemin bi-objectif

Nous nous intéressons dans ce chapitre à la détermination de l'ensemble des solutions non-dominées d'un problème de plus court chemin bi-objectif où les deux objectifs sont de type Min-Somme. En effet, si le profil de l'utilisateur ou ses préférences ne sont pas connus, une approche possible consiste à calculer la totalité des itinéraires de compromis. Nous nous trouvons ici dans le cas d'une méthode *a posteriori* où toutes les solutions sont calculées et ensuite, l'utilisateur choisit parmi la totalité des solutions non-dominées. Si cet ensemble est trop important, il est possible de le réduire en sélectionnant les solutions les plus représentatives.

Nous abordons les méthodes classiques de résolution de ce problème en section 4.1 et en particulier l'algorithme de *label-setting* bi-objectif en section 4.2. Nous proposons ensuite, dans la section 4.3, des règles d'élimination d'étiquettes basées sur un calcul préliminaire de bornes inférieures sur les coûts. Une nouvelle approche pour les méthodes de *labelling*, permettant d'accélérer la construction du front de Pareto est ensuite présentée dans la section 4.4 et nous comparons plusieurs prétraitements qui peuvent être combinés avec cette nouvelle méthode dans la section 4.5. Enfin, des résultats expérimentaux, sur des instances réelles, permettant d'évaluer l'ensemble des méthodes proposées, sont présentés dans la section 4.6.

Ce travail a donné lieu à plusieurs communications dans des conférences nationales [Sauvanet 09] et internationales [Sauvanet 10e, Sauvanet 10g] ainsi qu'à une soumission en revue internationale [Sauvanet 10a].

Données concernant le graphe G :	
G :	le graphe
V :	l'ensemble des nœuds
A :	l'ensemble des arcs
K :	l'ensemble des objectifs
$n = \|V\|$:	le nombre de nœuds
$m = \|A\|$:	le nombre d'arcs
$q = \|K\|$:	le nombre d'objectifs
s :	le nœud de départ
t :	le nœud de destination
(i, j) :	l'arc reliant le nœud i au nœud j
c_{ij} :	le vecteur de coûts associé à l'arc (i, j)
c_{ij}^k :	le coût selon l'objectif k associé à l'arc (i, j)

Données relatives à une étiquette l :	
l :	l'étiquette courante
P_l :	sous-chemin associé à l'étiquette l
d_l :	le vecteur de coûts associé à l'étiquette l
d_l^k :	le coût selon l'objectif k associé à l'étiquette l
$nœud(l)$:	le nœud de l'étiquette l
$l \rhd nœud(l)$:	l'étiquette l associée au *nœud(l)*

Données relatives à un nœud v :	
$étiquettes(v)$:	l'ensemble des étiquettes permanentes du nœud v
LB_v^k :	la borne inférieure de l'objectif k pour relier v à t
UB_v^k :	la borne supérieure de l'objectif k pour relier v à t

TABLE 4.1 – Table des notations du chapitre 3

4.1 État de l'art

Dans la littérature, il existe deux types de méthodes pour calculer l'ensemble des solutions non-dominées d'un problème de plus court chemin multiobjectif : les méthodes dites de *ranking* et les méthodes dites de *labelling*.

Les méthodes de *ranking* [Clímaco 82] sont basées sur le problème du k-ème plus court chemin. Ces méthodes déterminent le plus court chemin optimisant un seul objectif, plus le 2-ème plus court, etc.

Les méthodes de *labelling* peuvent être vues comme une généralisation des méthodes existantes d'étiquetage pour résoudre le problème mono-objectif. Parmi ces méthodes, l'algorithme de Dijkstra [Dijkstra 59] est le plus connu pour résoudre le problème mono-objectif. Une étiquette (label) est associée à chaque nœud et permet de préciser le coût du chemin reliant le nœud de départ et ce nœud. A chaque itération, l'étiquette de coût minimal est sélectionnée et étendue. Pour cela, de nouvelles étiquettes sont calculées en ajoutant au coût initial de l'étiquette, les coûts des arcs sortants. Une étiquette venant d'être étendue remplacera l'étiquette d'un nœud uniquement si son coût est inférieur au coût de celle-ci. Dans [Dijkstra 59], une queue d'exploration est utilisée pour ordonner la visite des étiquettes non traitées en fonction de leur coût (cf. algorithme 1).

Dans le cas multiobjectif, une étiquette n'est plus représentée par un seul coût mais par un vecteur de coûts. Il peut donc exister de nombreuses étiquettes non-dominées entre elles sur un même nœud. On subdivise généralement ces méthodes multiobjectif en deux : d'un côté les méthodes de *label-correcting* et de l'autre les méthodes de *label-setting*. Ces méthodes sont très similaires et diffèrent souvent uniquement par l'ordre d'exploration des étiquettes. Dans le cas des méthodes de type *label-setting*, nous sommes assurés qu'une étiquette explorée correspond à un chemin non-dominé. Il est possible de s'assurer de cela en utilisant, par exemple, un ordre d'exploration lexicographique. La première méthode de *label-setting* a été proposée dans [Hansen 80] et généralisée dans [Martins 84].

Par ailleurs, les méthodes de *label-correcting* utilisent généralement un ordre simple, comme l'ordre FIFO par exemple, pour parcourir la liste des étiquettes restantes à explorer. Contrairement aux méthodes de *label-setting*, une étiquette explorée à un moment donné peut correspondre à un chemin qui sera dominé par la suite. La première méthode de *label-correcting* a été établie dans [Vincke 74]. Il y a deux types de sélection différents dans cette méthode : la sélection par étiquette et la sélection par nœud. Dans la sélection par étiquette, chaque étiquette est gérée séparément alors que dans la sélection par nœud, toutes les étiquettes sur le nœud sélectionné sont étendues. Dans [Brumbaugh-Smith 89], plusieurs stratégies pour sélectionner les nœuds sont comparées. Dans [Guerriero 01], une comparaison est effectuée entre les stratégies de sélection par nœud et de sélection par étiquette. La sélection par nœud est communément utilisée pour les méthodes de *label-correcting*.

A côté des méthodes de *labelling* et de *ranking*, la méthode dite à deux phases est très couramment utilisée pour résoudre le problème bi-objectif. Ce concept a été introduit par [Ulungu 95]. Le principe est de calculer séparément les solutions supportées et les solutions non-supportées. Ainsi, toutes les solutions supportées sont calculées lors de la première

phase, en utilisant une méthode mono-objectif qui optimise une combinaison linéaire des deux objectifs. Une recherche dichotomique permet de calculer l'ensemble des solutions supportées. La seconde phase énumère les solutions non-supportées. Dans cette seconde phase, l'espace de recherche est réduit grâce aux solutions calculées lors de la première phase. En effet, en se plaçant dans l'espace des critères, les solutions non-supportées ne peuvent se trouver que dans un triangle situé entre chaque paire consécutive de solutions supportées. Il a été montré dans [Raith 09] que cette approche était efficace appliquée à des problèmes de plus court chemin bi-objectif.

Un des problèmes des méthodes proposées dans la littérature est que ces méthodes ont rarement été comparées sur des graphes de taille importante et avec des critères réellement conflictuels. Par exemple, dans [Raith 09], les auteurs ont comparé plusieurs stratégies pour déterminer l'ensemble des solutions non-dominées sur le problème bi-objectif. Les expérimentations ont été effectuées sur des graphes routiers en considérant les critères de distance et de temps. Sur le graphe du New Jersey (330 386 nœuds et 1 202 248 arcs), les auteurs obtiennent de 2 à 21 solutions non-dominées. Sur le réseau routier du département de l'Indre-et-Loire (136 199 nœuds et 345 145 arcs), utilisé dans le cadre d'expérimentations préliminaires avec les critères de distance et de sécurité, nous obtenons de 1 à 1033 solutions non-dominées pour un ensemble d'instances définies aléatoirement. Ainsi, pour pouvoir comparer les résultats de la littérature, il est important de prendre en compte la taille du graphe mais également le caractère conflictuel des critères.

4.2 Algorithme de *label-setting* bi-objectif

Comme nous l'avons vu dans la section 4.1, le problème de plus court chemin bi-objectif peut être résolu par des méthodes de *labelling* ou de *ranking*. Nous nous intéressons dans ce chapitre aux méthodes de *labelling* et plus particulièrement aux méthodes dites de *label-setting*. Cependant, toutes les améliorations proposées par la suite peuvent également être appliquées aux méthodes de *label-correcting*.

Nous présentons ici l'algorithme de *label-setting* classique [Martins 84]. Les algorithmes de *label-setting* sont une extension des méthodes mono-objectif. La principale différence est que pour un nœud v il n'y a pas qu'une seule étiquette mais un ensemble d'étiquettes noté *étiquettes*(v). Dans le cas de la sélection par étiquette, toutes les étiquettes à traiter sont insérées dans une queue d'exploration Q. L'algorithme de *label-setting* consiste uniquement à retirer l'étiquette l la plus petite du point de vue lexicographique. Grâce à cela, nous sommes assurés que l'étiquette l ne sera pas dominée par la suite sur ce même nœud. Cette propriété est importante car toutes les étiquettes construites à partir d'une étiquette dominée sont également dominées.

Dans l'algorithme 3, la première étape consiste à initialiser la liste des étiquettes associées à chaque nœud à l'ensemble vide, sauf pour le nœud s qui contient une première étiquette $(0, 0)$. Nous ajoutons ensuite $(0, 0) \rhd s$ à la queue d'exploration Q correspondant à l'étiquette $(0, 0)$ associée au nœud s (ligne 1 à 3).

Chaque itération correspond à l'extraction de l'élément minimal $l_i \rhd i$ de Q si celle-ci n'est pas vide, et à étendre cette étiquette l_i vers les arcs sortants $(i, j) \in A$ du nœud i

Algorithme 3 Algorithme de *label-setting*

1: *étiquettes*$(i) = \phi, i = \{1, ..., n\} \setminus \{s\}$
2: *étiquettes*$(s) = (0, 0)$
3: $Q \leftarrow \{(0, 0) \rhd s\}$
4: **tant que** $Q \neq \phi$ **faire**
5: $(l_i \rhd i) \leftarrow Q.Min()$
6: $Q \leftarrow Q \setminus (l_i \rhd i)$
7: **pour tout** $(i, j) \in A$ **faire** // *Pour chaque arc sortant du nœud* i
8: $l_j \leftarrow l_i + c_{ij}$
9: **si** $\nexists l \in$ *étiquettes*(j) tel que $l \prec_p l_j$ **alors**
10: *étiquettes*$(j) \leftarrow$ *étiquettes*$(j) \cup l_j$ // *On ajoute l'étiquette* l_j
11: $Q \leftarrow Q \cup l_j \rhd j$
12: **pour tout** $l \in$ *étiquettes*(j) tel que $l_j \prec_p l$ **faire**
13: // *On supprime les étiquettes dominées*
14: *étiquettes*$(j) \leftarrow$ *étiquettes*$(j) \setminus l$
15: **si** $(l \rhd j) \in Q$ **alors**
16: $Q \leftarrow Q \setminus (l \rhd j)$
17: **fin si**
18: **fin pour**
19: **fin si**
20: **fin pour**
21: **fin tant que**

associés à cette étiquette l_i (ligne 4 à 7).

Pour chaque arc sortant $(i, j) \in A$, une nouvelle étiquette $l_i + c_{ij}$ peut être construite et comparée aux étiquettes *étiquettes*(j) déjà présentes sur le nœud j. Si cette nouvelle étiquette n'est dominée par aucune des étiquettes de l'ensemble *étiquettes*(j), alors elle peut être ajoutée à cet ensemble, ainsi qu'à la queue d'exploration Q. De même, il faut supprimer les étiquettes de l'ensemble *étiquettes*(j) si elles sont dominées par cette nouvelle étiquette et également les retirer de la queue d'exploration Q si elles y sont présentes (ligne 8 à 19).

Notons que *étiquettes*(t) correspond, à la fin de l'algorithme de *label-setting*, à l'ensemble des coûts des chemins non-dominés entre le nœud s et le nœud t. Cet ensemble est le front de Pareto $\mathcal{PF}*$. Cependant, durant l'exécution de l'algorithme de *label-setting*, *étiquettes*(t) ne correspond pas au front de Pareto réel $\mathcal{PF}*$. Ainsi, par la suite, nous parlerons du *front de Pareto approximé* pour désigner *étiquettes*(t) qui est l'ensemble des vecteurs de coûts des chemins complets déterminés à l'itération courante.

Dans les sections suivantes, plusieurs méthodes sont proposées pour améliorer les performances de l'algorithme de *label-setting*. L'algorithme 4 présente comment s'articule et se combine l'ensemble des améliorations proposées. Nous commençons par introduire des règles d'élimination des étiquettes (cf. section 4.3.2) basées sur un calcul préliminaire de bornes inférieures et supérieures sur les coûts (cf. section 4.3.1) ainsi que sur l'utilisation du *front de Pareto approximé*. Ensuite, nous proposons une nouvelle méthode permettant

Algorithme 4 Cadre général de la détermination du front de Pareto

1: // s, t *les nœuds de départ et de destination*
2: Calcul des bornes inférieures et supérieures sur c^1 et c^2 pour l'ensemble des nœuds (cf. section 4.3.1)
3: Initialisation du *front de Pareto approximé* (cf. section 4.5)
4: // *Méthode de* label-setting *(cf. section 4.2)*
5: **tant que** Il existe des étiquettes à traiter **faire**
6: Sélectionner l'étiquette selon l'ordre lexicographique
7: Étendre l'étiquette courante
8: Éliminer l'étiquette si elle est non-prometteuse (cf. section 4.3.2)
9: Compléter le *front de Pareto approximé* (cf. section 4.4)
10: **fin tant que**
11: **retourner** le front de Pareto réel (correspondant au *front de Pareto approximé* courant)

d'accélérer la construction du *front de Pareto approximé* (cf. section 4.4). Enfin, nous terminons en comparant plusieurs initialisations possibles du *front de Pareto approximé* (cf. section 4.5).

4.3 Réduction du nombre d'étiquettes traitées

Comme le nombre d'étiquettes développées sur un nœud peut être exponentiel en théorie, il peut être intéressant de réduire leur nombre. Une méthode consiste à écarter une étiquette dès que l'on est assuré que celle-ci ne mènera qu'à des chemins dominés. Pour cela, des traitements préliminaires sont nécessaires pour calculer des bornes sur chacun des critères et à chaque nœud.

4.3.1 Calcul des bornes

Nous présentons ici (cf. algorithme 5) une adaptation de l'algorithme de Dijkstra [Dijkstra 59], présenté en section 3.1, qui pour chaque nœud i, calcule la distance de i à t pour un critère donné et la borne supérieure associée sur l'autre critère. Par exemple, $(LB_i^1; UB_i^2)$ représente le plus court chemin de i à t minimisant le critère 1, où LB_i^1 est la borne inférieure du critère 1, et UB_i^2 la borne supérieure associée du critère 2.

Ainsi, pour chaque nœud i, des bornes inférieures LB_i^1, LB_i^2 et supérieures UB_i^1, UB_i^2 peuvent être calculées. Les bornes inférieures correspondent aux coûts minimum des chemins reliant le nœud i au nœud t sur chacun des critères. Elles peuvent être déterminées en exécutant une recherche mono-objectif inverse pour chacun des deux critères. C'est-à-dire que ces recherches s'effectuent du nœud destination t vers tous les autres nœuds du graphe G et avec tous les arcs inversés. L'algorithme utilisé est une simple adaptation de [Dijkstra 59] qui optimise un des objectifs et stocke en plus le coût de l'autre objectif sur les nœuds.

Algorithme 5 Algorithme de calcul des bornes LB_i^1 et UB_i^2

1: $Q \leftarrow \{(0 \rhd t)\}$
2: $LB_t^1 \leftarrow 0$ and $LB_i^1 = +\infty, i = \{1, ..., n\} \setminus \{t\}$
3: $UB_t^2 \leftarrow 0$ and $UB_i^2 = +\infty, i = \{1, ..., n\} \setminus \{t\}$
4: **tant que** $Q \neq \phi$ **faire**
5: $\quad (c \rhd i) \leftarrow Q.Min()$
6: $\quad Q \leftarrow Q \setminus (c \rhd i)$
7: $\quad$ **pour tout** $(j, i) \in A$ **faire** *// Pour chaque arc entrant au nœud i*
8: $\quad\quad$ *// Si critère 1 amélioré ou critère 1 identique mais critère 2 amélioré*
9: $\quad\quad$ **si** $(LB_i^1 + c_{ji}^1 < LB_j^1)$ **ou** $(LB_i^1 + c_{ji}^1 = LB_j^1$ **et** $UB_i^2 + c_{ji}^2 < UB_j^2)$ **alors**
10: $\quad\quad\quad LB_j^1 \leftarrow LB_i^1 + c_{ji}^1$
11: $\quad\quad\quad UB_j^2 \leftarrow UB_i^2 + c_{ji}^2$
12: $\quad\quad\quad$ **si** $j \in Q$ **alors**
13: $\quad\quad\quad\quad Q.majPriorite(LB_j^1 \rhd j)$
14: $\quad\quad\quad$ **sinon**
15: $\quad\quad\quad\quad Q \leftarrow Q \cup (LB_j^1 \rhd j)$
16: $\quad\quad\quad$ **fin si**
17: $\quad\quad$ **fin si**
18: $\quad$ **fin pour**
19: **fin tant que**

L'algorithme 5 optimise l'objectif 1 et permet donc de calculer les bornes LB_i^1 et UB_i^2 pour chaque nœud i du graphe G. La condition à la ligne 8 est différente de ce qui est fait dans [Dijkstra 59], de manière à changer la borne UB_j^2 dans le cas où celle-ci serait améliorée alors que la borne LB_j^1 ne changerait pas de valeur. Il faut noter que cette adaptation de l'algorithme de Dijkstra conserve la même complexité en $O(m + n \cdot \log n)$. Les seules différences concernent les lignes 9 et 11. Dans l'algorithme de Dijkstra, une étiquette n'était mise à jour que si la valeur du critère considéré était améliorée. Sur la ligne 9, si la valeur du critère 1 est identique à celle de l'étiquette mais que la valeur du critère 2 est améliorée, l'étiquette est mise à jour. La ligne 11 consiste à garder en mémoire la borne supérieure du critère 2.

Pour résumer, cet algorithme est exécuté à deux reprises (en considérant successivement le critère 1 puis le critère 2) avant l'exécution de l'algorithme de *labelling* bi-objectif. Notons qu'à l'issue de ce calcul des bornes inférieures et supérieures, le *front de Pareto approximé* peut être initialisé avec les solutions (LB_s^1, UB_s^2) et (UB_s^1, LB_s^2).

4.3.2 Règles d'élimination des étiquettes

Ces règles d'élimination sont basées sur les travaux de [Tung 92] qui exploitent les bornes inférieures pour écarter le plus tôt possible des étiquettes dominées.

La première règle d'élimination revient à compléter le chemin partiel avec les bornes inférieures et à le comparer aux deux solutions non-dominées que sont : $(LB_s^1; UB_s^2)$ qui optimise l'objectif 1 et $(UB_s^1; LB_s^2)$ qui optimise l'objectif 2.

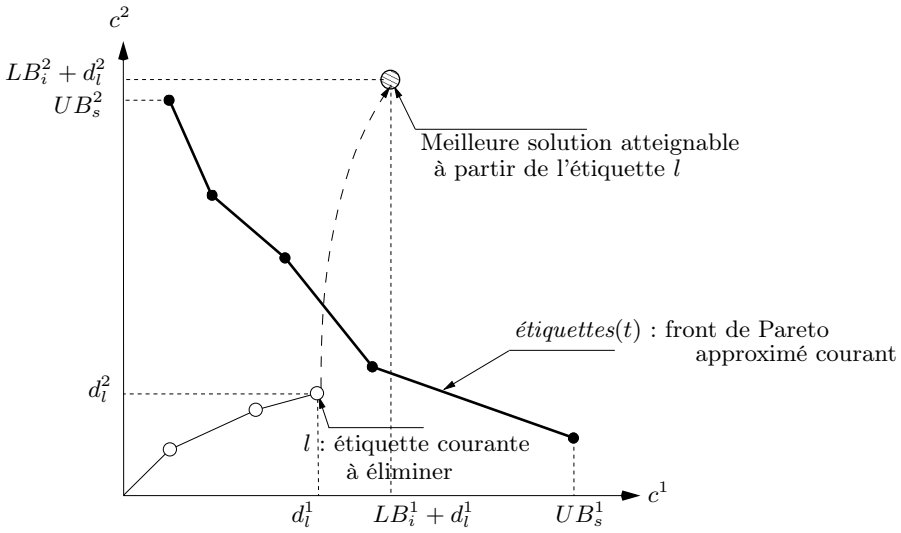


FIGURE 4.1 – Règle d'élimination utilisant les valeurs maximales des objectifs

Soit $l = (d_l^1; d_l^2)$ l'étiquette courante associée au nœud i et correspondant au coût d'un chemin du nœud s au nœud i.

Propriété 1. *Si $d_l^1 + LB_i^1 > UB_s^1$ ou $d_l^2 + LB_i^2 > UB_s^2$, alors l'étiquette courante peut être écartée. [Tung 92]*

La propriété 1 peut être vérifiée en temps constant lors du traitement de chaque étiquette.

Par exemple, sur la figure 4.1, l'étiquette $l = (d_l^1, d_l^2)$ correspond à un sous-chemin du nœud s au nœud i et la ligne en gras correspond aux vecteurs de coûts de l'ensemble des chemins complets, c'est-à-dire à une approximation du front de Pareto. Ici, $d_l^2 + LB_i^2 > UB_s^2$ donc toutes les étiquettes construites à partir de l'étiquette l correspondent à des chemins dominés. Ainsi le meilleur chemin pouvant être construit, c'est-à-dire le chemin composé du sous-chemin associé à l'étiquette l et le meilleur sous-chemin optimisant l'objectif 2 du nœud courant i jusqu'à t, est dominé par la solution de coût (LB_s^1, UB_s^2).

La seconde règle d'élimination est une extension de la précédente dans le sens où la solution partielle complétée par les bornes inférieures est cette fois-ci comparée à l'ensemble des solutions non-dominées, c'est-à-dire à *étiquettes*(t), le *front de Pareto approximé* courant.

Soit $l = (d_l^1; d_l^2)$ l'étiquette courante associée au nœud i et correspondant au coût d'un chemin du nœud s au nœud i.

Propriété 2. *Si $\exists l_f \in$ étiquettes(t) tel que $l_f \prec_p (d_l^1 + LB_i^1, d_l^2 + LB_i^2)$, alors l'étiquette courante peut être écartée. [Tung 92]*

Par exemple, sur la figure 4.2, le vecteur de coûts $(d_l^1 + LB_i^1, d_l^2 + LB_i^2)$, qui correspond au meilleur chemin potentiel, est dominé par un des vecteurs de coûts de l'ensemble des chemins complets. Donc, comme la meilleure solution qui peut être construite à partir de

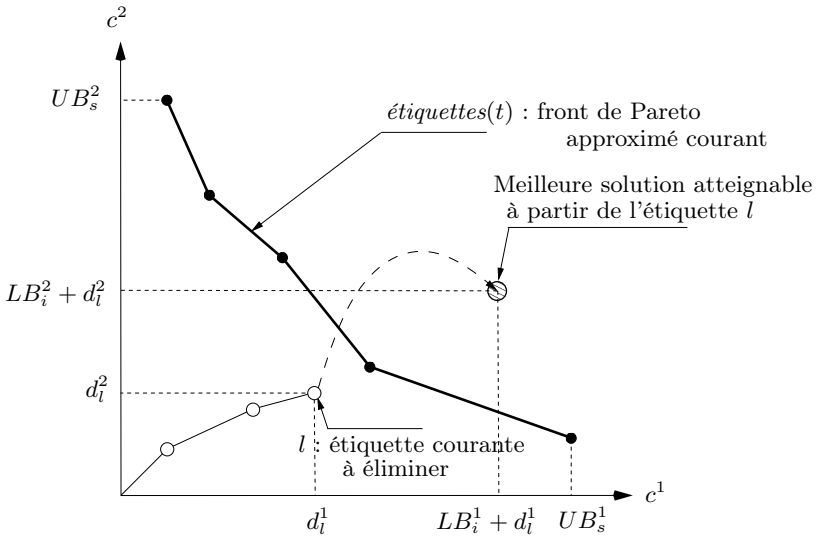


FIGURE 4.2 – Règle d'élimination utilisant les coûts des chemins complets du front de Pareto

cette étiquette est dominée par un chemin déjà trouvé, alors toutes les solutions complètes construites à partir de cette étiquette sont elles aussi dominées.

Cette deuxième règle peut être vue comme une généralisation de la première règle, il n'est donc pas nécessaire d'utiliser la première si l'on vérifie déjà la deuxième. Cependant, la première règle est beaucoup plus rapide à vérifier puisque seules deux solutions non-dominées sont considérées.

Cette dernière règle d'élimination est très efficace, surtout lorsque le nombre de chemins complets déjà calculés est important, c'est-à-dire lorsque le *front de Pareto approximé* est une bonne approximation du front réel. Le problème est que, d'une part, au début de l'algorithme de *label-setting*, les solutions complètes n'existent pas (à part les deux solutions optimisant chacun des deux objectifs) et donc cette règle d'élimination est mal exploitée. D'autre part, à cause de l'ordre d'exploration lexicographique des étiquettes, les solutions complètes sont mal réparties sur le *front de Pareto approximé*, ce qui rend la règle d'élimination moins efficace.

Nous proposons donc une méthode permettant d'accélérer la construction du front de Pareto et de l'approximer très rapidement, de manière à rendre beaucoup plus efficace la deuxième règle d'élimination des étiquettes.

4.4 Accélération de la construction du front de Pareto

La méthode proposée ici vise à accélérer l'identification de solutions non-dominées pour améliorer la convergence de l'approximation du front de Pareto vers le front de Pareto réel.

Définissons $opt1_i = (LB_i^1, UB_i^2)$ comme le vecteur de coûts du sous-chemin reliant le nœud i au nœud t optimisant le premier objectif et $opt2_i = (UB_i^1, LB_i^2)$ comme le vecteur

de coûts du sous-chemin reliant le nœud i au nœud t optimisant le deuxième objectif. Pour chaque nœud i, $opt1_i$ and $opt2_i$ peuvent être déterminés par les bornes inférieures LB_i^1, LB_i^2 et supérieures UB_i^1, UB_i^2 calculées lors de la phase de calcul des bornes.

Soit $l = (d_l^1; d_l^2)$ l'étiquette courante associée au nœud i et correspondant au coût d'un chemin du nœud s au nœud i.

Propriété 3. *A chaque itération de l'algorithme de* label-setting*, lorsqu'une étiquette* $(d_l^1; d_l^2)$ *associée à un nœud i est ajoutée à la queue d'exploration, deux vecteurs de coûts complets* $(d_l^1 + LB_i^1, d_l^2 + UB_i^2)$ *et* $(d_l^1 + UB_i^1, d_l^2 + LB_i^2)$*, correspondant à des solutions réalisables, peuvent être construits comme la concaténation du vecteur de coûts courant avec* $opt1_i$ *et* $opt2_i$*. [Sauvanet 10g]*

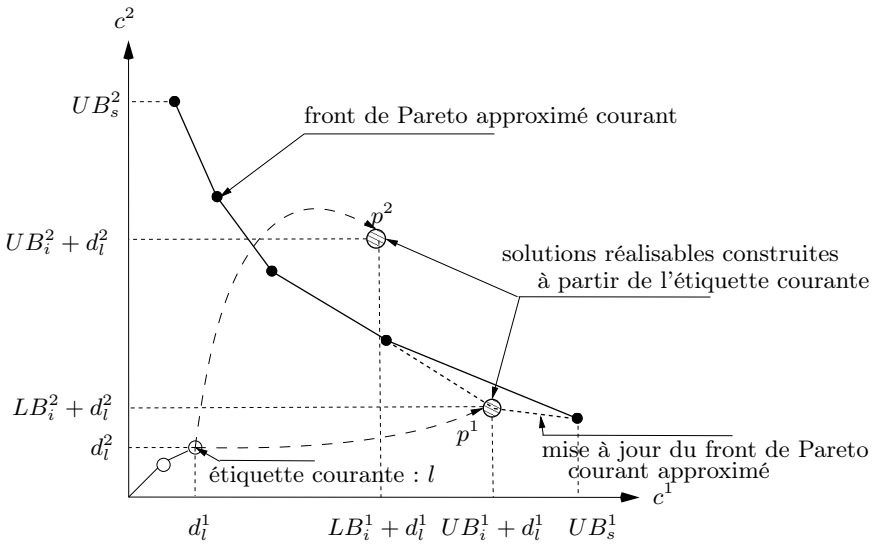


FIGURE 4.3 – Solutions réalisables construites à partir des bornes

Les deux vecteurs de coûts décrits à la propriété 3 correspondent à deux chemins complets réels. Ils sont en effet composés du sous-chemin de s à i correspondant à l'étiquette l et d'un sous-chemin de i à t optimisant l'un des objectifs. Ces deux vecteurs de coûts peuvent être fusionnés à l'ensemble *étiquettes*(t) pour maintenir le *front de Pareto approximé*. Cette opération de fusion consiste à ajouter un vecteur de coûts à cet ensemble, uniquement s'il n'est pas dominé et à supprimer les étiquettes qui sont dominées dans cet ensemble par ce nouveau vecteur de coûts. En bi-objectif, un tri lexicographique des ensembles d'étiquettes permet de ne pas comparer une étiquette avec l'ensemble complet. L'opération de fusion est détaillée dans la sous-section 4.6.1.

Par exemple sur la figure 4.3, l'étiquette $(d_l^1 + UB_i^1, d_l^2 + LB_i^2)$ n'est dominée par aucune étiquette de t donc il est possible de l'ajouter au *front de Pareto approximé étiquettes*(t) et de supprimer de cet ensemble les étiquettes qu'elle domine. Au contraire, l'étiquette $(d_i^1 + LB_i^1, d_i^2 + UB_i^2)$ est dominée et elle n'est donc pas prise en compte.

La figure 4.4 permet d'illustrer la convergence du *front de Pareto approximé* vers le front de Pareto réel. L'instance considérée est relativement facile : environ 35 km séparent

le nœud s du nœud t et le front de Pareto réel comporte 54 solutions non-dominées. Cette figure représente l'évolution du *front de Pareto approximé* dans l'espace des objectifs. Sur la figure, n est le nombre d'étiquettes explorées, la ligne noire correspond au front de Pareto exact et la ligne grise est notre *front de Pareto approximé*. Le front de Pareto exact n'est représenté que pour la comparaison et il n'est pas connu durant l'exécution de l'algorithme.

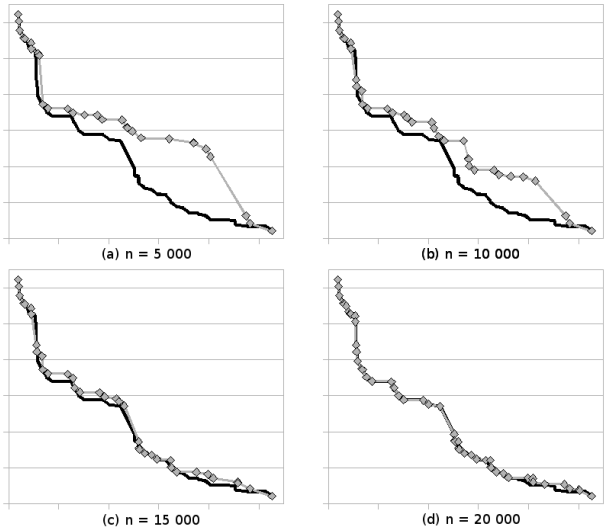


FIGURE 4.4 – Évolution du *front de Pareto approximé*

Nous pouvons remarquer que la méthode approxime très rapidement le front de Pareto. Cette approximation est importante car elle permet d'éliminer rapidement des étiquettes dominées grâce à la deuxième règle d'élimination (cf. propriété 2).

L'algorithme 6 correspond à l'algorithme de *label-setting* avec les règles d'élimination des étiquettes et l'approximation du front de Pareto. Dans cet algorithme, la fonction $fusion(l_j, étiquettes(j))$ (ligne 10 par exemple) permet de supprimer les étiquettes du nœuds j dominées par l'étiquette l_j et de retourner si cette étiquette est dominée ou non par une des étiquettes de l'ensemble $étiquettes$(j).

4.5 Initialisation des méthodes de *labelling*

L'ensemble des améliorations proposées dans la section précédente utilisent le *front de Pareto approximé* de manière à éliminer le plus tôt possible les étiquettes dominées. Au lancement d'une méthode de *labelling*, cet ensemble n'est composé que de deux solutions obtenues par des recherches mono-objectif. L'idée principale développée dans cette section est d'initialiser cet ensemble avec une approximation la plus proche possible du front de Pareto réel. Alors, la méthode de *labelling* pourra éliminer un plus grand nombre d'étiquettes au cours de sa recherche. Bien entendu, le calcul d'une bonne approximation de cet ensemble peut être long. La difficulté est de trouver le bon compromis entre temps de calcul et qualité de l'approximation initiale.

Dans cette section, deux approches très différentes sont étudiées : le calcul de l'ensemble

Algorithme 6 Algorithme de *label-setting* avec règles d'élimination et approximation du front de Pareto

1: $étiquettes(i) = \phi, i = \{1, ..., n\} \setminus \{s\}$
2: $étiquettes(s) = (0, 0)$
3: $Q \leftarrow \{(0, 0) \rhd s\}$
4: **tant que** $Q \neq \phi$ **faire**
5: $(l \rhd i) \leftarrow Q.Min()$
6: $Q \leftarrow Q \setminus (l \rhd i)$
7: **pour tout** $(i, j) \in A$ **faire** // *Pour chaque arc sortant du nœud i*
8: $l_j = (d^1_{l_j}; d^2_{l_j}) \leftarrow l + c_{ij}$
9: **si** $d^1_{l_j} + LB^1_j \leq UB^1_s$ **et** $d^2_{l_j} + LB^2_j \leq UB^2_s$ **alors** // *règle 1*
10: $estDominéeParJ \leftarrow fusion(l_j, étiquettes(j))$
11: $estDominéeParT \leftarrow$ **faux**
12: **si** $estDominéeParJ =$ **faux alors**
13: **pour tout** $l_t \in étiquettes(t)$ **faire**
14: **si** $l_t \prec_p (d^1_{l_j} + LB^1_j, d^2_{l_j} + LB^2_j)$ **alors**
15: $estDominéeParT \leftarrow$ **vrai**
16: **fin si**
17: **fin pour**
18: **fin si**
19: **si** $estDominéeParJ =$ **faux et** $estDominéeParT =$ **faux alors** // *règle 2*
20: $Q \leftarrow Q \cup (l_j \rhd j)$
21: $étiquettes(j) \leftarrow étiquettes(j) \cup l_j$
22: $l_{opt1} \leftarrow (d^1_{l_j} + LB^1_j, d^2_{l_j} + UB^2_j)$
23: $l_{opt2} \leftarrow (d^1_{l_j} + UB^1_j, d^2_{l_j} + LB^2_j)$
24: $Opt1estDominéeParT \leftarrow fusion(l_{opt1}, étiquettes(t))$
25: $Opt2estDominéeParT \leftarrow fusion(l_{opt2}, étiquettes(t))$
26: **si** $Opt1estDominéeParT =$ **faux alors**
27: $étiquettes(t) \leftarrow étiquettes(t) \cup l_{opt1}$
28: **fin si**
29: **si** $Opt2estDominéeParT =$ **faux alors**
30: $étiquettes(t) \leftarrow étiquettes(t) \cup l_{opt2}$
31: **fin si**
32: **fin si**
33: **fin si**
34: **fin pour**
35: **fin tant que**

des solutions supportées (cf. section 4.5.1) et une méthode de prétraitement inspirée de la méthode mono-objectif ALT [Goldberg 05a] (cf. section 4.5.2).

4.5.1 Calcul des solutions supportées

Cette initialisation est inspirée de la méthode à deux phases [Ulungu 95]. Cette méthode est très couramment utilisée dans la résolution des problèmes bi-objectif. Elle consiste à diviser la recherche des solutions en deux phases. Les solutions supportées sont d'abord calculées dans la première phase, puis les solutions non-supportées sont déterminées dans la deuxième phase. Les solutions supportées se trouvent par définition sur l'enveloppe convexe du front de Pareto dans l'espace des objectifs. Ces solutions peuvent être calculées très simplement avec une méthode mono-objectif minimisant une combinaison linéaire des deux objectifs, en faisant varier les poids relatifs des deux objectifs. Pour déterminer les solutions non-supportées, il est nécessaire d'utiliser une méthode bi-objectif. Cependant, les solutions supportées calculées lors de la première phase permettent de réduire l'espace de recherche à un ensemble de triangles dans l'espace des objectifs. Ce principe est illustré sur la figure 4.5 où les solutions supportées sont représentées dans l'espace des objectifs. Entre chaque paire de solutions supportées consécutives, le triangle hachuré correspond à la zone où des solutions non-supportées peuvent se trouver. A noter qu'aucune solution ne peut se trouver dans le triangle inférieur, auquel cas il s'agirait obligatoirement d'une solution supportée. Or, les solutions supportées ont été calculées lors de la première phase.

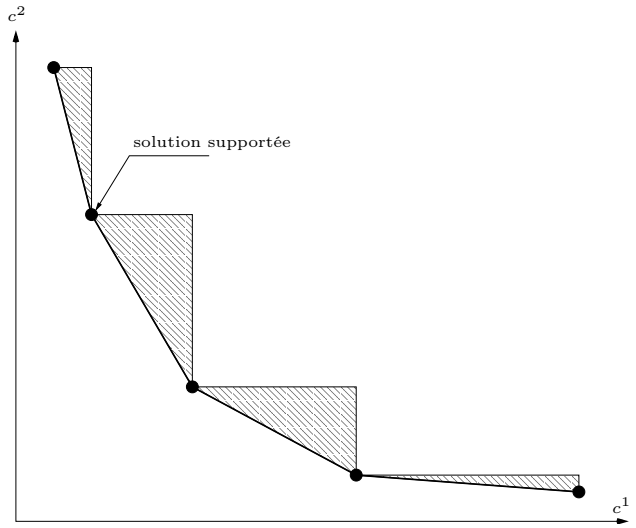


FIGURE 4.5 – Réduction de l'espace de recherche par les solutions supportées

La plupart des méthodes à deux phases appliquées au problème de plus court chemin bi-objectif consistent à lancer une recherche bi-objectif pour chaque paire de solutions supportées consécutives sur le front de Pareto. Comme dans [Raith 09] la méthode retenue ici consiste uniquement à calculer les solutions supportées pour initialiser nos méthodes de *labelling*, c'est-à-dire pour initialiser l'approximation du front de Pareto *étiquettes*(t). Ainsi, la méthode de *labelling* n'est ici exécutée qu'une unique fois.

Le calcul de l'ensemble des solutions supportées peut être effectué par une recherche dichotomique initialisée par les deux seules solutions optimisant chacun des deux objectifs. A chaque étape, cette méthode tente de trouver une solution supportée entre deux solutions a et c en appliquant une recherche mono-objectif, où l'objectif est la minimisation d'une combinaison linéaire des deux critères. La première étape vise à déterminer les poids associés à chacun des objectifs pour déterminer une solution supportée potentielle b en modifiant les poids de la combinaison linéaire. Si cette solution b est trouvée, la méthode est alors relancée entre a et b, et, entre b et c.

4.5.2 Calcul de solutions approchées

La détermination de l'ensemble des solutions supportées peut être coûteuse en temps de calcul. En effet, même si elle ne fait intervenir que des recherches mono-objectif, calculer toutes ces solutions sur une instance difficile peut nécessiter plusieurs secondes. Le but recherché avec cette seconde méthode d'initialisation est de pouvoir approximer le front de Pareto le plus rapidement possible, quitte à obtenir une approximation du front moins précise que dans la première méthode.

La méthode proposée s'inspire de la méthode ALT en mono-objectif [Goldberg 05a]. Dans ALT, un prétraitement important est réalisé en amont du calcul du plus court chemin. Ce dernier n'est effectué qu'une seule et unique fois pour un graphe donné et permet d'accélérer par la suite la résolution des instances. Lors de ce prétraitement, l'ensemble des plus courts chemins est calculé entre tous les nœuds V et un sous ensemble de nœuds $L \in V$ appelés Landmarks (repères en français). Ensuite, pour chaque instance à résoudre, ces informations couplées au principe d'inégalité triangulaire sont utilisées afin d'améliorer l'heuristique de la méthode mono-objectif A*.

Concernant le problème bi-objectif, comme un chemin optimal est par définition composé de sous-chemins optimaux, l'idée est de calculer et de sauvegarder en prétraitement un certain nombre de chemins optimaux. Ces chemins peuvent ensuite être utilisés pour approcher au mieux le front de Pareto d'une instance donnée. Par la suite nous nous référerons à cette méthode sous la dénomination LBA : Landmarks Based Approximation (approximation basée sur les *landmarks*).

Nous allons voir dans un premier temps comment réaliser la phase de prétraitement, et dans un second temps, pour une instance donnée, comment tirer parti de ces informations pour calculer une bonne approximation du front de Pareto.

Prétraitement

La première étape de cette phase de prétraitement consiste à sélectionner les *landmarks* $L \in V$, c'est-à-dire un sous ensemble de V. Nous reviendrons plus tard sur les différentes stratégies de sélection des *landmarks*. Les *landmarks* L correspondent à l'ensemble des nœuds d'un nouveau graphe $G' = (L, A')$. La deuxième étape calcule l'ensemble A' correspondant aux vecteurs de coûts des chemins non-dominés entre les *landmarks*. Pour réduire la taille du graphe G' et accélérer les calculs de chemins sur ce dernier, nous n'effectuons pas une recherche bi-objectif entre chaque couple de *landmarks*, mais seulement entre chaque

couple de *landmarks* « géographiquement voisins ». Ainsi, pour chaque *landmark*, nous recherchons les chemins non-dominés vers les k plus proches voisins à vol d'oiseau. Ce principe est illustré sur la figure 4.6 où pour le nœud i, l'ensemble des chemins non-dominés est calculé vers ses quatre plus proches voisins représentés ici en gris.

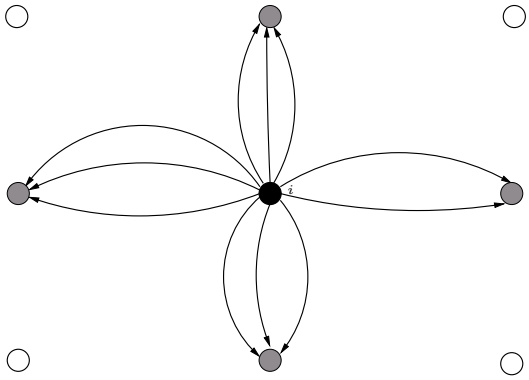


FIGURE 4.6 – Construction du graphe réduit G' reliant les *landmarks*

Bien sûr, pour chacune des recherches bi-objectif entre deux *landmarks*, il est possible d'utiliser n'importe quelle méthode de résolution (*label-setting* avec ou sans améliorations, *label-correcting*, etc.) sans aucune contrainte de temps vu que cette phase n'est effectuée qu'une seule fois en prétraitement.

Tout comme pour la méthode ALT en mono-objectif, la sélection des *landmarks* reste l'étape la plus critique. Pour commencer, il faut choisir le nombre de *landmarks* à sélectionner. S'il y a trop peu de *landmarks*, l'approximation du front de Pareto risque d'être très éloignée du front réel, et si le nombre de *landmarks* est trop important, le temps de calcul pour la phase d'approximation du front d'une instance risque de prendre trop de temps. Ce choix doit se faire en fonction de la taille du graphe.

Après avoir choisi le nombre de *landmarks*, il faut décider quels nœuds parmi l'ensemble V seront sélectionnés. La première idée est d'essayer de répartir les *landmarks* de façon géographique sur une grille carrée (par exemple un landmark tout les 5 kilomètres). Intuitivement, il semble plus intéressant de sélectionner comme *landmarks* les nœuds par lesquels beaucoup de chemins non-dominés passent. Il peut s'agir par exemple, sur un graphe routier, d'intersections importantes, de ponts, etc. Pour cela, nous proposons de lancer un nombre important de recherches mono-objectif, c'est-à-dire de choisir aléatoirement le nœud de départ, le nœud de destination et le coefficient de compromis de la combinaison linéaire de la fonction objectif, sur le graphe initial. Pour chaque chemin calculé, les compteurs présents sur chaque nœud emprunté par le chemin sont incrémentés. Cette étape permet de faire ressortir les nœuds fréquemment utilisés sur le graphe, c'est-à-dire appartenant fréquemment à des solutions non-dominées. Pour garder tout de même une bonne répartition des *landmarks* sur l'ensemble du graphe, nous sélectionnons pour

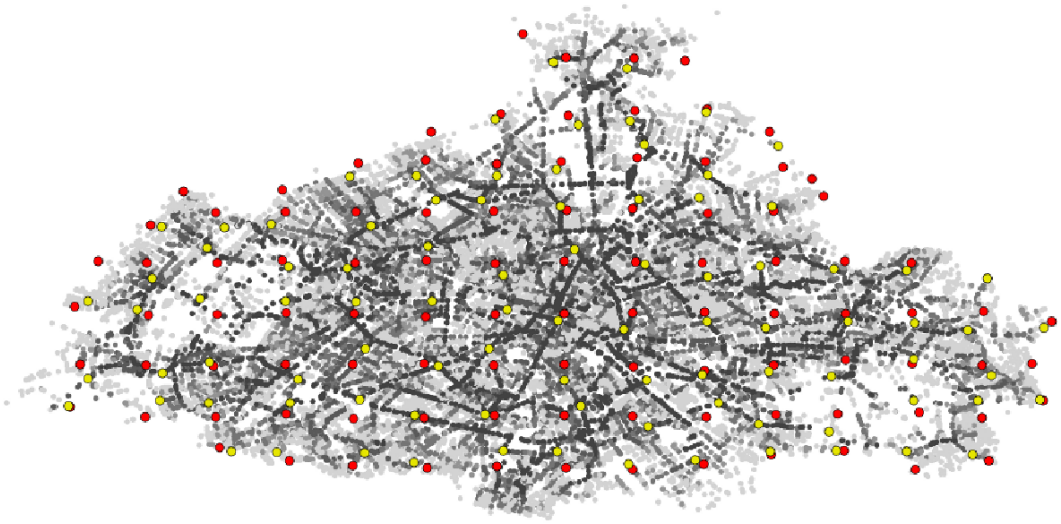


FIGURE 4.7 – Stratégies de sélection des *landmarks* appliquées sur le graphe de Paris

chaque zone géographique le nœud le plus fréquemment utilisé.

Ces deux stratégies de sélection des *landmarks* sont illustrées sur la figure 4.7 où les cercles rouges sont les *landmarks* sélectionnés en utilisant une répartition géographique stricte et les cercles jaunes sont, eux, obtenus par répartition des nœuds les plus fréquemment utilisés à l'intérieur des zones géographiques précédemment définies. L'ensemble des nœuds du graphe est représenté par des petits cercles : plus un nœud est sombre et plus il est fréquemment traversé par des chemins non-dominés. Ce système de représentation permet de distinguer visuellement des nœuds très empruntés et des zones peu utilisées.

Approximation du front de Pareto

Pour une instance donnée notée x définie par un nœud de départ s et de destination t, nous souhaitons pouvoir approximer le front de Pareto en lançant une recherche bi-objectif sur le graphe réduit $G' = (L, A')$. Les nœuds s et t n'appartenant pas forcément à G', construisons le graphe $G_x = (V_x, A_x)$ spécifique à l'instance x. Le graphe G_x peut se construire simplement à partir de G' :

- $V_x = L \cup s \cup t$, l'ensemble V_x est composé des *landmarks* auxquels on ajoute les nœuds s et t,
- $A_x = A' \cup A_{sl} \cup A_{lt}$ avec A_{sl} les chemins non-dominés du nœud s vers les *landmarks* voisins de s, et A_{lt} les chemins non-dominés des *landmarks* voisins de t vers le nœud t.

La principale difficulté est ici de calculer les ensembles A_{sl} et A_{lt}. D'abord, il parait inutile de calculer l'ensemble des sous-chemins non-dominés, que ce soit entre s et l'ensemble

des *landmarks* ou entre l'ensemble des *landmarks* et t. Comme pour la construction de l'ensemble des arcs A' du graphe G', nous proposons d'effectuer les recherches uniquement sur les *landmarks* voisins des nœuds s et t.

Ensuite le calcul de l'ensemble des chemins non-dominés pour calculer les ensembles A_{sl} et A_{lt} peut être trop long. Comme l'objectif est ici de calculer une approximation rapide du front de Pareto, nous proposons de ne calculer que les deux chemins optimisant respectivement chacun des deux objectifs pour chaque paire de nœuds. La figure 4.8 montre un exemple de graphe G_x construit pour une instance donnée. Par souci de lisibilité, tous les arcs ne sont pas représentés : chaque arc sur la figure correspond donc en réalité à un ensemble d'arcs.

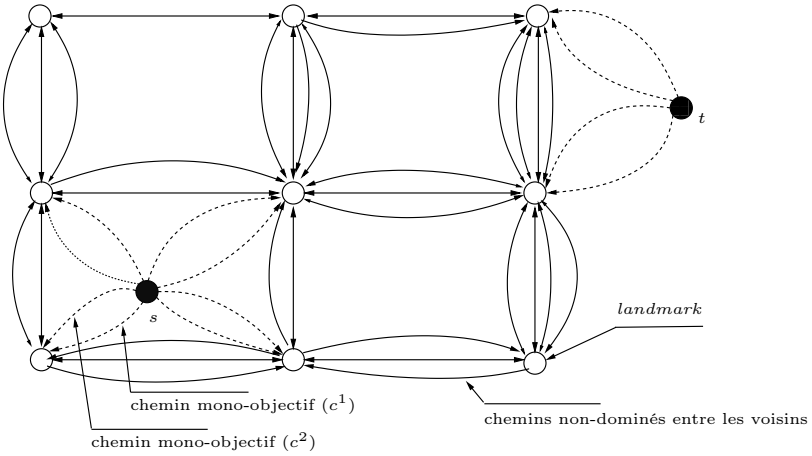


FIGURE 4.8 – Construction du graphe réduit G_x propre à une instance x

Avec la définition précédente des ensembles A_{sl} et A_{lt}, cette étape peut être réalisée rapidement. En effet, il n'est pas nécessaire de lancer des recherches mono-objectif pour calculer l'ensemble A_{lt} : celui-ci a déjà été calculé lors de l'étape de calcul des bornes (cf. section 4.3.1) de la méthode de *label-setting*. En revanche, pour l'ensemble A_{sl}, il est nécessaire de lancer deux recherches mono-objectif du nœud s, ce qui reste raisonnable étant donné que nous pouvons stopper ces deux recherches dès qu'elles ont atteintes l'ensemble des *landmarks* voisins à s.

Une fois G_x construit, il ne reste plus qu'à lancer une recherche bi-objectif de s vers t sur le graphe $G_x = (V_x, A_x)$. Les solutions obtenues constituent alors l'approximation du front de Pareto permettant d'initialiser nos méthodes.

4.6 Expérimentations

Pour déterminer les performances des nouvelles méthodes proposées dans ce chapitre, ces dernières ont été implémentées puis testées sur des graphes réels de taille différente.

4.6.1 Implémentation

Nous présentons ici quelques détails à propos de l'implémentation en langage C de ces différents algorithmes. Pour l'ensemble des structures du graphe, la librairie LEDA (pour Library of Efficient Data types and Algorithms) a été utilisée. Pour les algorithmes de *labelling* il existe deux principales difficultés :

- Quelle structure de données adopter pour gérer la liste des étiquettes restantes à explorer ?

- Comment détecter le plus rapidement possible si une étiquette est dominée par un ensemble d'étiquettes ou si elle domine une partie de cet ensemble d'étiquettes (problème de fusion) ?

Pour gérer la liste des étiquettes, une queue de priorité, implémentée par des piles de Fibonacci [Fredman 87] a été utilisée pour toutes les recherches mono-objectif, comme le calcul des bornes. Dans un contexte bi-objectif avec la méthode de *label-setting*, il est nécessaire d'explorer les étiquettes dans un ordre lexicographique. La structure de données utilisée pour cela est une séquence triée utilisant l'implémentation « Skiplists » [Pugh 90]. Pour ces deux structures de données, les opérations d'insertion et de suppression prennent un temps en $O(log\ n)$ et la sélection de l'étiquette la plus petite d'un point de vue lexicographique est réalisée en $O(1)$.

Le problème de fusion d'une étiquette avec un ensemble d'étiquettes est la difficulté principale concernant l'implémentation d'un algorithme de *labelling* (cf. lignes 8 à 19 de l'algorithme 3). Cette opération de fusion est également rencontrée dans la mise à jour de l'approximation du front de Pareto (cf. algorithme 6 appelé LSAP). Pour résoudre ce problème en bi-objectif, il est possible d'utiliser une séquence triée à chaque nœud pour stocker la liste des étiquettes non-dominées. Cette structure de données utilise l'implémentation « Skiplists » [Pugh 90]. Avec cette structure, toutes les étiquettes sur un nœud sont ordonnées dans l'ordre lexicographique (la clé de l'élément). L'avantage de cette structure est qu'elle permet de trouver très rapidement l'élément précédent ou suivant à partir d'une certaine valeur de clé. Par exemple, les opérations d'insertion et de suppression prennent un temps en $O(log\ n)$ et les instructions pour situer une étiquette précédente ou suivante à partir d'une certaine valeur de clé prennent un temps en $O(1)$.

Pour connaitre si une étiquette l est dominée par un ensemble d'étiquettes $\acute{e}tiquettes(i)$, il suffit de situer l'étiquette précédente dans $\acute{e}tiquettes(i)$ à partir de la valeur de clé correspondante à l. Si l'étiquette récupérée ne domine pas l, alors nous sommes certain qu'aucune étiquette de $\acute{e}tiquettes(i)$ ne domine l. De la même façon, pour savoir si l'étiquette l domine des éléments de $\acute{e}tiquettes(i)$, il suffit de lister les étiquettes de $\acute{e}tiquettes(i)$ en commençant par l'étiquette suivante correspondant à la valeur de clé de l.

Cette structure de données permet donc de ne pas comparer entièrement l'ensemble des étiquettes lors de l'opération de fusion. L'inconvénient est que cette méthode n'est possible que dans un contexte bi-objectif.

4.6.2 Résultats

Toutes les méthodes présentées dans ce chapitre, et les chapitres suivants, ont été testées sur trois graphes réels correspondant à des réseaux routiers. Ils proviennent tous du projet libre OpenStreetMap et sont donc librement utilisables.

Le premier graphe est celui de la ville de Paris et des 30 communes limitrophes où sont présentes les stations vélib'. Ce graphe a été choisi car il s'agit d'une des zones couvertes par Géovélo et pour laquelle l'association a vérifié les données sur le terrain et complété les informations sur OpenStreetMap. Ce graphe est le plus petit des trois : il est composé de 29 086 nœuds et 64 538 arcs.

Le deuxième graphe retenu est celui de Berlin. Il a été choisi car l'Allemagne est le pays où la communauté OpenStreetMap est la plus active. Les données concernant les aménagements cyclables sont ainsi très détaillées. Ce graphe est d'une taille plus importante que celui de Paris : il est composé de 59 673 nœuds et 145 840 arcs, ce qui représente environ deux fois la taille du graphe de Paris.

Le troisième et dernier graphe est celui de la baie de San francisco. Il a été choisi pour de nombreuses raisons. D'abord, même si les aménagements cyclables ne sont pas encore très bien renseignés aux Etats-Unis sur OpenStreetMap, la source de données TIGER (du domaine public), qui a été entièrement importée sur OpenStreetMap, permet d'obtenir une bonne couverture de l'ensemble de la voirie. De plus, la structure du réseau routier est très différente aux Etats-Unis (rues en forme de quadrillage) et il peut être intéressant de voir les répercussions sur nos méthodes. Enfin, ce graphe est le plus grand des trois avec 174 975 nœuds et 435 959 arcs, ce qui représente environ deux fois et demi la taille du graphe de Berlin. Ce graphe permet de voir comment se comportent les algorithmes de *labelling* sur des graphes de taille importante.

Pour chaque graphe, 200 instances ont été construites à partir d'une sélection aléatoire de nœuds de départ et de destination. Ces instances ont été regroupées en fonction du nombre de solutions non-dominées. En effet, des résultats préliminaires montrent que la difficulté d'une instance est fortement corrélée au nombre d'étiquettes développées et donc au nombre de solutions non-dominées. Le tableau 4.2 montre la répartition des instances pour chacun des graphes. Chaque classe d'instances comporte 50 instances.

Pour chaque instance, deux méthodes principales ont été testées. La première, **LSLB**, est un algorithme de *label-setting* classique avec l'ajout d'un test de dominance entre la meilleure solution potentielle, construite à partir des bornes inférieures, et les solutions du front de Pareto en cours de construction (cf. propriétés 1 et 2 correspondant à [Tung 92]). La deuxième, **LSAP**, est la même méthode mais avec l'amélioration permettant d'accélérer la convergence de l'approximation du front de Pareto (cf. propriété 3). Ces deux méthodes ont été testées sans initialisation du front de Pareto, et avec trois initialisations différentes : avec les solutions supportées (cf. section 4.5.1), et avec deux répartitions différentes **LBA1** (répartition géographique) et **LBA2** (répartition par nœuds les plus fréquemment utilisés) de la méthode d'approximation (cf. section 4.5.2). Le nombre de *landmarks* est respectivement de 102 pour le graphe de Paris, 226 pour le graphe de Berlin et de 479 pour le graphe de San Francisco. Pour déterminer la position des *landmarks* utilisés par la méthode LBA2, des itinéraires ont été calculés de manière aléatoire et sont au nombre de 2 000 pour Paris,

Classe	Graphe	Nombre de solutions
P1	Paris	$0 \leq \lvert\mathcal{PF}*\rvert < 50$
P2	Paris	$50 \leq \lvert\mathcal{PF}*\rvert < 100$
P3	Paris	$100 \leq \lvert\mathcal{PF}*\rvert < 200$
P4	Paris	$200 \leq \lvert\mathcal{PF}*\rvert$
B1	Berlin	$0 \leq \lvert\mathcal{PF}*\rvert < 100$
B2	Berlin	$100 \leq \lvert\mathcal{PF}*\rvert < 200$
B3	Berlin	$200 \leq \lvert\mathcal{PF}*\rvert < 300$
B4	Berlin	$300 \leq \lvert\mathcal{PF}*\rvert$
SF1	San Francisco	$0 \leq \lvert\mathcal{PF}*\rvert < 100$
SF2	San Francisco	$100 \leq \lvert\mathcal{PF}*\rvert < 300$
SF3	San Francisco	$300 \leq \lvert\mathcal{PF}*\rvert < 600$
SF4	San Francisco	$600 \leq \lvert\mathcal{PF}*\rvert$

TABLE 4.2 – Nombre de solutions non-dominées par classe d'instances

3 000 pour Berlin et 10 000 pour San Francisco.

Les tests ont été effectués sur un ordinateur portable avec un processeur Intel dual core à 2,20 Ghz et 3 Go de mémoire vive. L'ensemble des résultats sont reportés sur les tableaux de résultats suivants :

- tableau 4.4 : nombre d'étiquettes développées des méthodes basées sur LSLB,
- tableau 4.3 : temps d'exécution des méthodes basées sur LSLB,
- tableau 4.6 : nombre d'étiquettes développées des méthodes basées sur LSAP,
- tableau 4.5 : temps d'exécution des méthodes basées sur LSAP.

	LSLB	LSLB+1Phase	LSLB+LBA1	LSLB+LBA2
P1	0,13	0,20	0,21	**0,12**
P2	0,44	0,39	0,51	**0,24**
P3	1,37	0,64	1,27	**0,63**
P4	7,77	**2,05**	6,61	3,21
B1	0,33	0,49	0,40	**0,30**
B2	2,25	1,35	2,04	**0,89**
B3	7,73	**2,56**	6,70	2,77
B4	24,55	**6,15**	22,85	11,11
SF1	**0,55**	1,00	0,88	0,85
SF2	7,16	**3,02**	7,30	5,83
SF3	40,93	**10,10**	40,47	31,71
SF4	269,56	**67,06**	270,75	221,45

TABLE 4.3 – Résultats : temps d'exécution (secondes) LSLB et initialisations

Concernant la méthode classique de *labelling* LSLB, nous pouvons confirmer que le nombre d'étiquettes traitées est fortement corrélé au nombre de solutions non-dominées

	LSLB	LSLB+1Phase	LSLB+LBA1	LSLB+LBA2
P1	27 351	**3 597**	21 799	8 614
P2	114 288	**18 639**	95 085	42 921
P3	306 915	**55 827**	251 491	132 949
P4	1 196 473	**317 769**	1 005 880	553 948
B1	67 952	**11 976**	56 646	26 637
B2	430 030	**84 201**	360 657	166 551
B3	1 175 655	**252 257**	999 322	501 022
B4	2 935 651	**721 281**	2 667 328	1 516 065
SF1	63 838	**12 283**	62 335	54 075
SF2	1 108 660	**187 912**	1 077 553	873 747
SF3	4 587 243	**1 037 622**	4 495 707	3 681 313
SF4	21 879 306	**6 442 366**	21 730 266	17 921 156

TABLE 4.4 – Résultats : nombres d'étiquettes traitées LSLB et initialisations

de l'instance. Ainsi sur le graphe de Paris, nous passons de 27 351 étiquettes en moyenne pour les instances de moins de 50 solutions à 1 196 473 étiquettes en moyenne pour les instances de plus de 200 solutions. Ce constat est encore plus flagrant sur les graphes de taille importante comme celui de San Francisco où le nombre moyen d'étiquettes pour les instances de la classe SF1 est de 63 838 et celui de la classe SF4 est de 21 879 306 étiquettes. Le temps d'exécution moyen de la méthode est bien sûr fortement corrélé au nombre d'étiquettes qui passe de moins d'une seconde pour les instances de la classe SF1 sur San Francisco à presque 5 minutes pour les instances de la classe SF4.

Pour réduire le temps de calcul, il faut donc essayer de réduire le nombre d'étiquettes traitées et c'est l'objectif de la méthode LSAP. Avec cette méthode, on constate une réduction du nombre d'étiquettes moyen allant de 67% à 85%, toutes classes d'instances confondues par rapport à la méthode LSLB. Cependant, les meilleurs résultats sont obtenus sur les instances les plus faciles (85% pour la classe P1 sur Paris) et les résultats sont moins bons sur les instances de taille plus importante (67% pour la classe SF4 sur San Francisco). En termes de temps de calcul, ce dernier est réduit de 23% à 79%. Contrairement à ce que l'on pourrait penser, cette réduction de 23% du temps de calcul correspond à la classe P1 du graphe sur Paris. Cette faible réduction du temps de calcul, bien que la réduction en termes d'étiquettes explorées soit importante, s'explique par les traitements ajoutés par la méthode LSAP et le faible nombre de solutions sur chaque nœud. Ainsi, sur les instances de très petite taille, la méthode LSAP ne montre pas de grandes améliorations par rapport à la méthode LSLB. Par contre, pour les instances de grande taille, la réduction en termes de temps de calcul est bien visible : 75% pour la classe P4, 76% pour la classe B4 et 69% pour la classe SF4.

Concernant maintenant l'ajout d'un prétraitement à la méthode LSLB, nous pouvons constater qu'une telle combinaison permet toujours de réduire le nombre d'étiquettes traitées. C'est l'initialisation avec les solutions supportées qui est la plus efficace, avec une réduction variant de 71% à 87%. Concernant les initialisations LBA, on constate des résultats beaucoup plus variables en fonction de la structure du graphe et du choix des *landmarks*. Ainsi concernant le graphe de Paris, l'initialisation LBA1 permet de réduire le

	LSAP	LSAP+1Phase	LSAP+LBA1	LSAP+LBA2
P1	**0,10**	0,19	0,14	**0,10**
P2	**0,15**	0,39	0,24	0,16
P3	**0,32**	0,63	0,42	0,33
P4	1,92	1,94	2,08	**1,89**
B1	**0,16**	0,49	0,27	0,26
B2	**0,49**	1,33	0,61	0,57
B3	**1,51**	2,49	1,73	1,58
B4	5,90	6,03	6,25	**5,83**
SF1	**0,38**	1,00	0,71	0,72
SF2	**1,52**	3,01	1,82	1,87
SF3	**9,36**	10,10	**9,36**	9,51
SF4	83,34	**66,92**	81,59	84,40

TABLE 4.5 – Résultats : temps d'exécution (secondes) LSAP et initialisations

	LSAP	LSAP+1Phase	LSAP+LBA1	LSAP+LBA2
P1	4 130	**2 536**	4 122	3 348
P2	22 761	**13 151**	22 706	18 303
P3	61 647	**42 030**	60 978	55 062
P4	338 142	**253 258**	337 902	304 946
B1	15 476	**9 098**	15 010	12 045
B2	93 458	**64 565**	92 932	80 461
B3	282 319	**196 289**	280 321	254 822
B4	837 684	**601 887**	834 957	781 041
SF1	12 824	**8 874**	12 817	12 678
SF2	237 868	**147 354**	237 889	232 480
SF3	1 260 826	**858 649**	1 260 529	1 242 433
SF4	7 215 147	**5 693 741**	7 214 997	7 152 569

TABLE 4.6 – Résultats : nombres d'étiquettes traitées LSAP et initialisations

nombre moyen d'étiquettes entre 16% et 20% contre 54% et 69% pour LBA2. Pour Berlin, les résultats sont similaires avec une réduction allant de 9% à 17% pour LBA1 contre 48% à 61% pour LBA2. Enfin, pour le graphe de San Francisco, les résultats obtenus sont moins bons, avec une réduction du nombre moyen d'étiquettes allant de 1% à 3% pour LBA1 contre 15% à 21% pour LBA2. Nous pouvons donc constater que la répartition des *landmarks* est primordiale et que la deuxième répartition semble plus efficace. Cependant, si l'initialisation avec les solutions supportées reste la plus efficace en termes de réduction du nombre d'étiquettes traitées, il faut prendre en compte que cette initialisation reste la plus longue au niveau du temps calcul. Ainsi, LBA2 est plus rapide sur les petites classes d'instances : P1, P2 et P3 pour Paris, B1 et B2 pour Berlin et SF1 pour San Francisco.

	Meilleure méthode LSLB		Meilleure méthode LSAP	
	méthode	temps	méthode	temps
P1	LSLB+LBA2	0,12	**LSAP**	**0,10**
P2	LSLB+LBA2	0,24	**LSAP**	**0,15**
P3	LSLB+LBA2	0,63	**LSAP**	**0,32**
P4	LSLB+1Phase	2,05	**LSAP+LBA2**	**1,89**
B1	LSLB+LBA2	0,30	**LSAP**	**0,16**
B2	LSLB+LBA2	0,89	**LSAP**	**0,49**
B3	LSLB+1Phase	2,56	**LSAP**	**1,51**
B4	LSLB+1Phase	6,15	**LSAP+LBA2**	**5,83**
SF1	LSLB	0,55	**LSAP**	**0,38**
SF2	LSLB+1Phase	3,02	**LSAP**	**1,52**
SF3	LSLB+1Phase	10,10	**LSAP**	**9,36**
SF4	LSLB+1Phase	67,06	**LSAP+1Phase**	**66,92**

TABLE 4.7 – Résultats : comparaison entre la meilleure méthode LSLB et LSAP

Concernant l'ajout d'un prétraitement à la méthode LSAP, cela permet presque toujours de réduire le nombre d'étiquettes traitées. Encore une fois, c'est l'initialisation avec les solutions supportées qui est la plus efficace. Mais contrairement à la combinaison LSLB et initialisation avec les solutions supportées, la réduction varie ici de 21% à 42% contre 71% à 87% pour LSLB. Cette réduction en termes de nombre d'étiquettes traitées n'est pas toujours suffisante pour compenser le temps de calcul des solutions supportées. Ainsi, la combinaison LSAP initialisée avec les solutions supportées est plus rapide que la méthode LSAP seule uniquement sur la classe SF4. Pour les initialisations LBA, nous constatons que LBA1 n'apporte que très peu d'amélioration à LSAP : la réduction du nombre d'étiquettes traitées varie entre 0% et 3% seulement. LBA2 est plus performante avec de 10% à 19% d'étiquettes en moins pour Paris et de 7% à 22% des étiquettes en moins pour Berlin. Par contre, les résultats sont moins bons sur San Francisco, avec seulement de 1% à 2% de réduction. En termes de temps de calcul, LBA ne parvient à être plus performant que sur les classes P4 et B4.

Le tableau 4.7 synthétise l'ensemble des résultats et compare les temps d'exécution de la meilleure des combinaisons de LSLB avec la meilleure des combinaisons de LSAP. Nous remarquons que les méthodes basées sur LSAP sont toujours plus rapides que les méthodes LSLB. Concernant LSLB, les initialisations LBA2 sont efficaces pour les instances de petite

taille alors que le calcul des solutions supportées est plus adapté au instances de grande taille. Du côté des méthodes LSAP, la combinaison la plus efficace pour les petites instances est la méthode LSAP sans initialisation. Par contre, pour les instances de grande taille, l'initialisation LBA2 est efficace pour les classe P4 et B4, et l'initialisation avec les solutions supportées est la plus efficace pour les instances de la classe SF4.

Pour conclure, la méthode LSAP est très efficace car elle permet de réduire de façon très importante le nombre d'étiquettes traitées. Même si la réduction est moins importante qu'avec la méthode LSLB initialisée avec les solutions supportées, la méthode LSAP ne perd pas de temps à initialiser un ensemble de solutions de départ. Cependant, sur les instances de très grande taille (classe SF4), le calcul des solutions supportées est particulièrement efficace.

4.7 Conclusion du chapitre

Dans cette partie, nous avons considéré le cas bi-objectif du problème du plus court chemin où la distance et l'insécurité sont minimisées. Ne connaissant pas les préférences de l'utilisateur, l'ensemble du front de Pareto est calculé par des méthodes de *labelling*. Cependant, pour pouvoir être utilisé sur un site web, le temps de calcul est limité et il s'est avéré nécessaire d'améliorer les méthodes de la littérature.

Dans ce chapitre, des règles d'élimination et des améliorations de l'approximation du front de Pareto ont été proposées. Les méthodes ont été testées sur des instances réelles. Les améliorations proposées permettent de calculer l'ensemble des solutions du front de Pareto sur les instances de petite et moyenne taille. Cependant, sur les instances de grande taille, cela n'est pas suffisant pour une utilisation sous la forme d'un site web par exemple.

Plusieurs axes de recherche sont possibles. D'une part, le calcul de plusieurs centaines de solutions n'est pas intéressant pour l'utilisateur. Il est donc possible d'avoir recours à différentes méthodes d'échantillonnage pour réduire le nombre de solutions calculées. D'autre part, dans ce chapitre nous avons fait l'hypothèse que les préférences ou le profil de l'utilisateur n'étaient pas connus (méthode de résolution *a posteriori*). Pourtant, avec un service sous la forme d'un site web, ce type d'information peut tout à fait être connu et pris en compte par le calculateur d'itinéraires. Dans ce cas, il est opportun de se tourner vers des méthodes de résolution *a priori*, où l'objectif n'est pas de déterminer toutes les solutions non-dominées, mais la ou les solutions de meilleur compromis. Ces méthodes sont développées dans le chapitre 5.

Chapitre 5

Recherche d'une solution de meilleur compromis pour un problème de plus court chemin multiobjectif

Dans le chapitre précédent (cf. chapitre 4), nous avons vu que la détermination complète des solutions non-dominées était possible mais trop coûteuse en termes de temps de calcul sur des instances de taille importante, comme sur les graphes de Berlin ou de San Francisco. En effet, le calcul d'un itinéraire ne doit pas prendre plus de trois secondes, étant donné que le service prend la forme d'un site web. De plus, dans le chapitre précédent, nous avons fait l'hypothèse qu'un contexte bi-objectif était suffisant alors qu'en réalité, de nombreux critères doivent être pris en compte dans le choix d'un itinéraire adapté aux cyclistes (cf. section 2.2).

Dans ce chapitre, nous ne nous intéressons plus à la détermination complète du front de Pareto, mais au calcul d'une solution de meilleur compromis d'un problème de plus court chemin multiobjectif. Nous nous plaçons donc dans le cas où les préférences de l'utilisateur sont connues (approches *a priori*) et où l'on cherche la solution satisfaisant au mieux ses préférences. Une méthode de la littérature est ici reprise [Futtersack 00] et améliorée [Sauvanet 10b] en utilisant des prétraitements rapides.

Nous commencerons par présenter dans la section 5.1 la notion de solution de meilleur compromis ainsi que les méthodes classiques de résolution. Dans la section 5.2, nous introduirons la méthode BCA* particulièrement adaptée à ce problème. Ensuite, dans la section 5.3, nous montrerons comment il est possible d'améliorer considérablement cette méthode. Enfin, des résultats expérimentaux, sur des instances réelles, permettant d'évaluer l'ensemble de ces méthodes, sont présentés dans la section 5.4.

Ce travail a été présenté dans une conférence internationale [Sauvanet 10b] et a donné lieu à une soumission en revue internationale [Sauvanet 10c].

Données concernant le graphe G :	
G :	le graphe
V :	l'ensemble des nœuds
A :	l'ensemble des arcs
K :	l'ensemble des objectifs
$n = \|V\|$:	le nombre de nœuds
$m = \|A\|$:	le nombre d'arcs
$q = \|K\|$:	le nombre d'objectifs
s :	le nœud de départ
t :	le nœud de destination
(i, j) :	l'arc reliant le nœud i au nœud j
c_{ij} :	le vecteur de coûts associé à l'arc (i, j)
c_{ij}^k :	le coût selon l'objectif k associé à l'arc (i, j)

Données relatives à une étiquette l :	
l :	l'étiquette courante
P_l :	sous-chemin associé à l'étiquette l
d_l :	le vecteur de coûts associé à l'étiquette l
d_l^k :	le coût selon l'objectif k associé à l'étiquette l
$nœud(l)$:	le nœud de l'étiquette l
$l \rhd nœud(l)$:	l'étiquette l associée au *nœud(l)*

Données relatives à un nœud v :	
étiquettes(v) :	l'ensemble des étiquettes permanentes du nœud v
LB_v^k :	la borne inférieure de l'objectif k pour relier v à t
UB_v^k :	la borne supérieure de l'objectif k pour relier v à t
LC_v^k :	le coût de l'objectif k selon une combinaison linéaire des objectifs pour relier v à t

Données relatives au compromis :	
r :	vecteur de coûts de la solution idéale
ω :	vecteur des préférences
f :	fonction d'agrégation des objectifs

TABLE 5.1 – Table des notations du chapitre 4

5.1 État de l'art

Comme nous l'avons vu dans le chapitre précédent, le calcul de l'ensemble des solutions dominées peut être relativement important, même pour deux critères. Pourtant, l'objectif est bien d'intégrer ces méthodes dans une plate-forme en ligne avec des temps de réponse les plus courts possibles (quelques secondes au maximum). De plus, il n'est pas pertinent de proposer à l'utilisateur plusieurs centaines d'itinéraires.

Lorsque les attentes de l'utilisateur sont connues *a priori*, il est possible d'utiliser un autre type de méthode se concentrant sur la détermination de la meilleure solution de compromis du point de vue des préférences de l'utilisateur. Nous nous intéressons donc dans ce chapitre à une méthode de type *a priori*, contrairement au chapitre précédent qui traitait une méthode de type *a posteriori*. Nous verrons tout d'abord ce qu'est une solution de meilleur compromis et comment elle peut être modélisée. Puis nous aborderons les méthodes qui permettent de calculer cette solution de meilleur compromis.

5.1.1 Solution de meilleur compromis

Contrairement au calcul de l'ensemble des solutions non-dominées où les préférences de l'utilisateur n'interviennent pas et où le choix entre plusieurs solutions non-dominées est laissé à l'utilisateur, une solution est définie comme solution de meilleur compromis à partir des préférences de l'utilisateur. Deux éléments sont alors essentiels pour déterminer la meilleure solution de compromis :

- les paramètres de préférences qui représentent l'importance relative de chacun des objectifs,
- une fonction d'agrégation des objectifs qui permet d'évaluer la performance d'une solution.

Paramètres de préférences

Soit l un vecteur de coûts correspondant à un chemin solution P_l que l'on cherche à évaluer.

Différents types de paramètres de préférences peuvent être distingués. Le plus simple dans un premier temps peut être de demander à l'utilisateur l'importance de chacun des critères sous la forme d'un vecteur de poids λ défini tel que $\lambda \in \mathbb{R}^q$ avec $q = |K|$ et $\lambda^k \geq 0, \forall k \in K$. Par exemple, dans un contexte bi-objectif (distance, insécurité), le vecteur $\lambda = (1, 10)$ indique que, pour l'utilisateur, la sécurité du chemin est dix fois plus importante que la distance. La plupart du temps, $\sum_{k \in K} \lambda^k = 1$.

Un autre type de paramètre est le niveau de réserve r qui correspond, dans un problème de minimisation, au seuil sur chacun des objectifs à ne pas dépasser. Par exemple, dans un contexte bi-objectif (distance, insécurité), le niveau de réserve $r = (5, 8)$ signifie que l'on ne veut pas obtenir de chemin dont la distance dépasserait la valeur 5, ou dont l'insécurité dépasserait la valeur 8.

Le dernier type de paramètres est le point de référence r. Il s'agit du point visé dans l'espace des objectifs, et la solution de meilleur compromis sera celle qui s'en approchera le plus.

Une méthode de calcul de la solution de meilleur compromis peut utiliser un ou plusieurs de ces paramètres en fonction du problème et des besoins.

Un des problèmes qui se pose est alors de savoir comment déterminer ces paramètres. Dans une approche interactive, c'est au décideur de déterminer ces paramètres. Cependant, il peut être intéressant de réduire au maximum l'intervention du décideur. Par exemple, le décideur peut ne pas être expert du domaine ou bien il souhaite avoir une première proposition avant d'affiner ses préférences. Dans [Vanderpooten 89a], plusieurs alternatives sont présentées pour spécifier la valeur de ces paramètres. Le cas le plus classique est de prendre comme point de référence r le point idéal, c'est-à-dire le vecteur de coûts composé pour chaque objectif de sa valeur minimale. La valeur minimale d'un objectif peut être obtenue en traitant le problème mono-objectif. Par exemple, si le chemin le plus court correspond au vecteur de coûts $(5, 20)$ et le chemin le plus sécurisé correspond au vecteur de coûts $(10, 12)$ alors le point idéal r sera défini par le vecteur de coûts $(5, 12)$.

Concernant le vecteur de poids λ, il peut être déterminé à partir de la dispersion de chacun des objectifs [Benayoun 71] ou d'une mesure de l'entropie [Zeleny 82]. Utiliser la dispersion dans l'espace des objectifs [Benayoun 71] permet de normaliser l'importance des différents objectifs qui peuvent avoir des plages de valeurs complètement différentes. Pour cela, il suffit de calculer $\lambda^k = \frac{1}{(UB_s^k - LB_s^k)}$, $\forall k \in K$. Grâce à cette méthode, un objectif dont les valeurs iraient de 1000 à 5000 aura la même importance qu'un second objectif dont les valeurs iraient de 1 à 10.

La méthode de l'entropie [Zeleny 82] consiste à calculer l'entropie d'information portée pour chaque objectif. C'est à dire que la méthode calcule la moyenne d'information générée par l'ensemble des solutions sur chacun des critères.

A noter que toutes ces méthodes nécessitent des solutions déjà calculées pour déterminer la valeur de λ. Les optimisations mono-objectif sont alors couramment utilisées.

Enfin, même si ces méthodes permettent d'obtenir automatiquement un vecteur de poids λ, il est toujours possible de le pondérer pour prendre en compte les préférences du décideur.

Fonctions d'agrégation

Dans [Vanderpooten 89a, Vanderpooten 89b], différents types de fonctions d'agrégation sont proposés :

– les fonctions basées sur une **somme pondérée** :

$$f(l, \lambda) = \sum_{k \in Q} \lambda^k d_l^k \tag{5.1}$$

– les fonctions basées sur une **norme pondérée** :

$$f(l,\lambda,r) = ||d_l - r||_p^{\lambda} = \left[\sum_{k\in Q}\{|\lambda^k(d_l^k - r^k)|^p\}\right]^{1/p} \tag{5.2}$$

Avec $p \in \{1, 2, ..., \infty\}$. Ce type de fonction utilise un point de référence r. La norme de Tchebycheff est couramment utilisée ($p = \infty$) :

$$f(l,\lambda,r) = \max_{k\in Q}\{|\lambda^k(d_l^k - r^k)|\} \tag{5.3}$$

– les fonctions basées sur une **norme pondérée augmentée**, comme la norme de Tchebycheff augmentée [Steuer 83] :

$$f(l,\lambda,r) = \max_{k\in Q}\{|\lambda^k(d_l^k - r^k)|\} + \varepsilon\sum_{k\in Q}\lambda^k(d_l^k - r^k) \tag{5.4}$$

– les fonctions basées sur les **niveaux d'aspiration** :

$$f(l,\lambda,r) = \max_{k\in Q}\{\lambda^i(d_l^k - r^k)\} \tag{5.5}$$

Deux propriétés des fonctions d'agrégation sont intéressantes à vérifier. La première propriété consiste à déterminer si l'optimisation de la fonction f permet toujours d'obtenir une solution non-dominée. Il est en effet logique de vouloir obtenir uniquement des solutions non-dominées lorsque l'on optimise f.

La deuxième propriété consiste à déterminer s'il existe toujours, pour chaque solution non-dominée, au moins un vecteur λ permettant de l'atteindre à partir de la fonction f. En d'autres termes, cette deuxième propriété signifie que l'optimisation de la fonction f permet de couvrir l'ensemble des solutions non-dominées. Cette propriété est essentielle pour permettre de parcourir l'ensemble des solutions non-dominées à partir de f.

Concernant les fonctions basées sur une somme pondérée (cf. équation 5.1), leur optimisation permet bien de vérifier la première propriété mais pas la deuxième. En effet, elles ne permettent d'obtenir que les solutions supportées [Geoffrion 68].

L'optimisation de la norme de Tchebycheff (cf. équation 5.3) satisfait, elle, la deuxième propriété mais pas la première. En effet, il est possible, en optimisant une norme de Tchebycheff, d'obtenir une solution faiblement efficace, donc dominée par une autre solution [Wierzbicki 86].

Il n'existe malheureusement pas de fonction d'agrégation vérifiant ces deux propriétés. En pratique, un bon compromis est d'utiliser une fonction basée sur l'optimisation d'une norme de Tchebycheff augmentée (cf. équation 5.4) [Zeleny 74, Wierzbicki 86]. En effet, le second terme $\varepsilon\sum_{k\in Q}\lambda^k(d_l^k - r^k)$ permet de distinguer deux solutions de même distance de Tchebycheff.

Dans [Galand 08], l'auteur étudie plusieurs fonctions d'agrégation pour rechercher des solutions de compromis dans les problèmes d'optimisation multiobjectif dans les graphes et notamment pour le problème de plus court chemin multiobjectif. L'opérateur OWA

[Yager 88] pour *Ordered Weighted Averaging* (moyenne pondérée ordonnée) permet de nuancer la notion de compromis. En effet, la norme de Tchebycheff se concentre sur la plus mauvaise performance des objectifs, ce qui n'est pas toujours pertinent. L'auteur donne l'exemple d'un problème bi-objectif où les deux objectifs sont à minimiser. En comparant la norme de Tchebycheff, la solution $x_1 = (16, 17)$ est meilleure que $x_2 = (18, 13)$ car la plus mauvaise valeur de x_1 est inférieure à la plus mauvaise valeur de x_2 ($17 < 18$). Cependant, x_2 peut constituer une bonne solution de compromis, car sur le pire des objectifs, la différence entre les deux solutions n'est que d'une unité, alors que sur l'autre objectif, x_2 est meilleur que x_1 de 3 unités ($13 < 16$). L'opérateur OWA permet ainsi de prendre en compte tous les objectifs et pas uniquement la pire performance des objectifs.

Enfin, l'intégrale de Choquet [Choquet 53] repose sur la notion de capacité et permet une gestion beaucoup plus fine des préférences. La notion de capacité permet en effet de définir des interactivités entre les objectifs. A savoir également que la somme pondérée, la norme de Tchebycheff et l'opérateur OWA sont des intégrales de Choquet particulières. L'optimisation de l'intégrale de Choquet pour les problèmes de plus court chemin multiobjectif est abordée dans [Galand 08, Fouchal 11].

5.1.2 Calcul de la solution de meilleur compromis

La complexité du problème de calcul de la solution de meilleur compromis dépend de la fonction d'agrégation utilisée. Si la fonction d'agrégation utilisée est linéaire, comme la somme pondérée (cf. équation 5.1), la complexité du problème est polynomiale car cela revient à un problème mono-objectif. Par contre, si la fonction d'agrégation n'est pas linéaire, comme par exemple la norme de Tchebycheff, le problème est alors NP-Difficile [Yu 98]. Notons également que le principe d'optimalité de Bellman [Bellman 57], qui précise que tout sous-chemin d'un chemin optimal est optimal, n'est pas vérifié ici, contrairement au problème de détermination de l'ensemble des solutions non-dominées.

Pour orienter la recherche du plus court chemin en mono-objectif, la méthode A* [Hart 68] (cf. 3.1.3) est particulièrement efficace. La recherche est orientée par une heuristique permettant d'évaluer un nœud i par l'addition du coût réel du sous-chemin de s à i avec une borne inférieure sur le coût du sous-chemin restant de i à t. Pour que le chemin calculé reste optimal, il est important que l'estimation de l'heuristique reste inférieure ou égale au coût réel du chemin.

Dans [Stewart 91], l'auteur propose une extension de la méthode A* au cas multiobjectif. Cette méthode est appelée MOA*, pour *Multi-Objective A**. Il s'agit d'un algorithme de *label-correcting* où la sélection se fait sur les nœuds. La différence se situe au niveau des étiquettes stockées sur chaque nœud i qui ne correspondent plus aux coûts réels des sous-chemins de s à i. En effet, en plus du coût réel, l'estimation heuristique des coûts de i à t est ajoutée à chaque étiquette de i. Comme pour A*, pour que MOA* calcule des chemins optimaux, l'estimation heuristique doit être inférieure ou égale aux coûts réels.

Dans [Mandow 05], l'auteur améliore MOA* en proposant non plus une sélection par nœud mais par étiquette. Récemment, une étude empirique [Machuca 10] montre que la sélection par étiquette de MOA* serait la plus performante lorsque les objectifs considérés

Algorithme 7 Algorithme MOA*

```
1: étiquettes(i) = φ, i = {1, ..., n} \ {s}
2: étiquettes(s) = (0, 0)
3: Q ← {(0, 0) ▷ s}
4: tant que Q ≠ φ faire
5:     (l_i ▷ i) ← Q.Min(h(i))
6:     Q ← Q \ (l_i ▷ i)
7:     pour tout (i, j) ∈ A faire  // Pour chaque arc sortant du nœud i
8:         l_j ← l_i + c_ij
9:         si ∄l ∈ étiquettes(j) tel que l ≺_p l_j alors
10:            étiquettes(j) ← étiquettes(j) ∪ l_j  // On ajoute l'étiquette l_j
11:            Q ← Q ∪ (l_j ▷ j)
12:            pour tout l ∈ étiquettes(j) tel que l_j ≺_p l faire
13:                // On supprime les étiquettes dominées
14:                étiquettes(j) ← étiquettes(j) \ l
15:                si (l ▷ j) ∈ Q alors
16:                    Q ← Q \ (l ▷ j)
17:                fin si
18:            fin pour
19:        fin si
20:    fin pour
21: fin tant que
```

sont fortement conflictuels. La méthode MOA* avec sélection par étiquette est présentée dans l'algorithme 7. Sur la ligne 5, la sélection d'une étiquette se fait en cherchant l'étiquette pour laquelle la somme de son vecteur de coûts et de son heuristique est minimale.

La méthode MOA* permet d'orienter la recherche pour le calcul de l'ensemble des solutions non-dominées. Mais cette méthode ne permet pas de guider la recherche vers la solution de meilleur compromis. La méthode BCA* [Futtersack 00], pour *Best Compromise A**, est basée sur MOA* avec sélection par étiquette et reprend l'utilisation d'une heuristique pour estimer les coûts minimum du sous-chemin restant. Cependant, l'ordre de sélection des étiquettes n'est plus lexicographique mais dépend d'une fonction d'évaluation dépendant des préférences du décideur. Ainsi BCA* ordonne la sélection des étiquettes, de manière à sélectionner les étiquettes les plus prometteuses d'un point de vue des préférences du décideur.

5.2 Méthode BCA*

La méthode BCA* [Futtersack 00] est une méthode de *label-correcting*. Pour faciliter les explications, nous allons considérer le cas bi-objectif mais cette méthode peut être étendue à plus de deux critères. Une étiquette l est alors représentée par les valeurs des deux objectifs (d_l^1, d_l^2), d'un nœud associé, *nœud*(l), et d'un sous-chemin P_l du nœud s au nœud *nœud*(l).

La spécificité de la méthode BCA* est d'ordonner les étiquettes suivant une fonction d'agrégation f, comme par exemple la norme de Tchebycheff (voir l'équation 5.6), en tenant compte du fait que le vecteur de coûts d'une étiquette est complété par les bornes inférieures du coût du sous-chemin restant pour atteindre le nœud de destination t. Ces bornes sont calculées séparément avant l'exécution de la méthode BCA*, par une recherche mono-objectif inverse pour chacun des critères (voir section 4.3.1). Suite à cette étape, la borne inférieure LB_v^k correspond au coût minimum de l'objectif $k \in Q$ pour atteindre le nœud t depuis le nœud courant v. A noter que durant l'exécution de ces méthodes, nous pouvons également calculer les valeurs des coûts des autres objectifs notés respectivement UB_v^k and UB_v^k.

La fonction $f(l, r, \omega)$ est utilisée pour ordonner la sélection des étiquettes. r est le vecteur de référence et ω est le vecteur de poids.

$$f(l, r, \omega) = \max_{k \in \{1,2\}} \{\omega^k . |(d_l^k + LB^k_{nœud(l)}) - r^k|\} \tag{5.6}$$

La méthode BCA* utilise la norme de Tchebycheff entre la solution idéale (caractérisée par le vecteur de référence r) et la meilleure solution potentielle. Cette norme permet de focaliser la recherche sur le pire des objectifs et de trouver des solutions équilibrées en fonction de ω, modélisant les préférences de l'utilisateur.

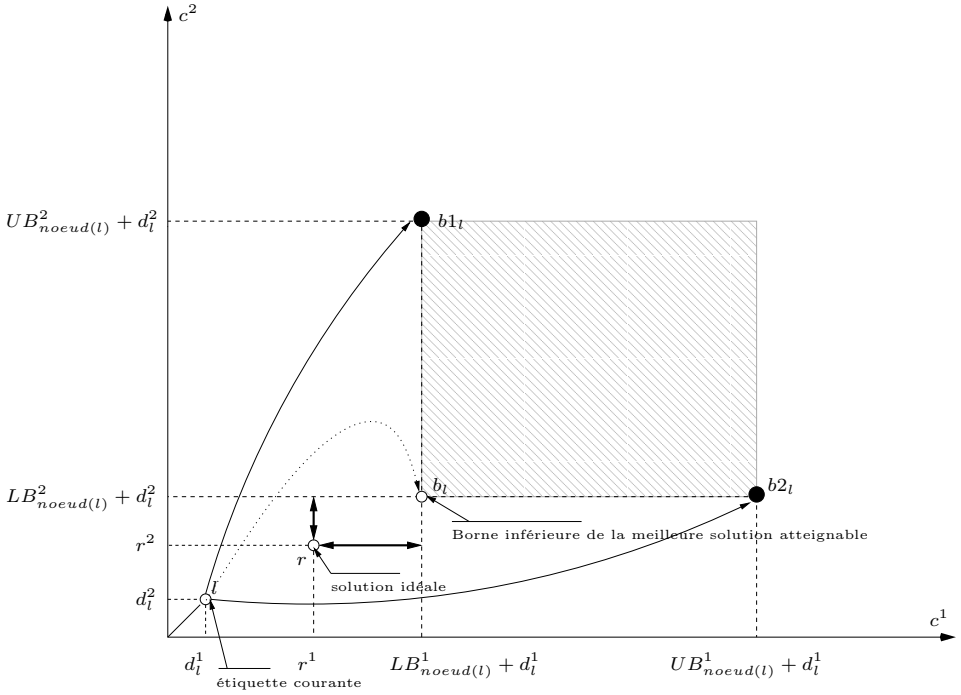


FIGURE 5.1 – Evaluation de la norme de Tchebycheff par la méthode BCA*

La figure 5.1 est un exemple montrant une étiquette l, d'un sous-chemin P_l, qui est représentée dans l'espace des objectifs par son coût (d_l^1, d_l^2). Sur cette figure, $r(r_1, r_2)$ est le point correspondant au vecteur de référence, c'est-à-dire à la solution idéale. Dans cet

exemple, $f(l, r, \omega) = \max\{\omega^1.|(d_l^1 + LB^1_{nœud(l)}) - r_1|; \omega^2.|(d_l^2 + LB^2_{nœud(l)}) - r_2|\}$. Pour plus de clarté, les bornes supérieures $UB^1_{nœud(l)}$ and $UB^2_{nœud(l)}$ ont été ajoutées sur la figure pour représenter $b1_l$ et $b2_l$: ces deux vecteurs de coûts correspondent aux chemins existants obtenus en complétant le sous-chemin actuel P_l par les sous-chemins optimisant soit l'objectif 1, soit l'objectif 2. Ces sous-chemins ont été calculés avant l'exécution de la méthode BCA*, lors du calcul des bornes inférieures et supérieures. Ces chemins complets ne sont pas utilisés dans la méthode BCA*.

Sur la figure 5.1, la zone hachurée correspond à l'ensemble des vecteurs de coûts des solutions non-dominées et réalisables commençant par le sous-chemin P_l. Dans la méthode originale de BCA*, le vecteur de coûts $b_l : (d_l^1 + LB^1_{nœud(l)}, d_l^2 + LB^2_{nœud(l)})$ est utilisé par la fonction f comme le vecteur de coûts correspondant à la meilleure solution potentielle, c'est-à-dire le vecteur de coûts minimisant la norme de Tchebycheff. Il est important de noter que b_l est utilisé pour calculer la borne inférieure de la norme de Tchebycheff entre la solution idéale et la meilleure solution potentielle. Bien sûr, rien ne garantit qu'il existe un chemin complet de s à t et passant par $nœud(l)$ dont le vecteur de coûts est égal à b_l. Comme les étiquettes sont ordonnées par la fonction f, l'idée principale des méthodes proposées ici est d'améliorer la borne inférieure de cette meilleure solution potentielle, dans le but d'accélérer la convergence vers la solution de meilleur compromis.

Pour comprendre les améliorations que nous allons proposer dans la section suivante, il est important de préciser que la méthode BCA* ordonne les étiquettes par l'évaluation de la fonction f, c'est-à-dire la borne inférieure de la norme de Tchebycheff entre la meilleure solution potentielle et la solution idéale. Cette méthode a ainsi deux avantages : elle permet d'explorer les étiquettes les plus prometteuses en premier et elle permet également d'éliminer les étiquettes dont l'évaluation par la fonction f est plus importante que la meilleure solution trouvée jusque là.

5.3 Améliorations de l'ordre d'exploration de BCA*

Comme mentionné précédemment, l'objectif est d'évaluer le plus précisément possible la meilleure solution atteignable depuis le sous-chemin P_l correspondant à l'étiquette courante l. Cette notion de meilleure solution atteignable est définie d'un point de vue de la norme de Tchebycheff. Nous proposons ici des bornes inférieures de cette norme de Tchebycheff entre le vecteur de coûts de la meilleure solution atteignable et le vecteur de coûts de référence. Ces bornes peuvent être utilisées de manière à ordonner l'exploration des étiquettes. Ainsi, plus ces bornes inférieures seront proches de la norme de Tchebycheff réelle et plus la convergence vers la meilleure solution de compromis sera rapide. Ces évaluations utilisent des données calculées par des recherches mono-objectif exécutées en prétraitement.

5.3.1 Approche par point pour l'évaluation de la norme de Tchebycheff

De la même façon qu'il est possible de calculer des bornes inférieures et supérieures sur chaque nœud v, il est envisageable de calculer les coûts LC_v^1 et LC_v^2 obtenus par une recherche mono-objectif optimisant une combinaison linéaire des critères. Pour pondérer

les objectifs, nous utiliserons par la suite le vecteur de préférences ω. Notons que l'usage de ω pour calculer LC_v^1 et LC_v^2 n'est pas obligatoire, mais que cela simplifie grandement les notations pour la suite. Cependant, ce choix est quand même pertinent car il correspond aux préférences de l'utilisateur. Cette recherche mono-objectif permet de déterminer le coût, LC_v^k pour chaque objectif k, correspondant au coût du sous-chemin du nœud v vers le nœud t. Pour chaque étiquette l et son sous-chemin P_l de s à $nœud(l)$, nous pouvons définir l_l comme le vecteur de coûts du chemin complet composé de P_l combiné avec le sous-chemin de $nœud(l)$ vers t, qui optimise la combinaison linéaire des objectifs, dont le vecteur de coûts est $(LC^1_{nœud(l)}; LC^2_{nœud(l)})$.

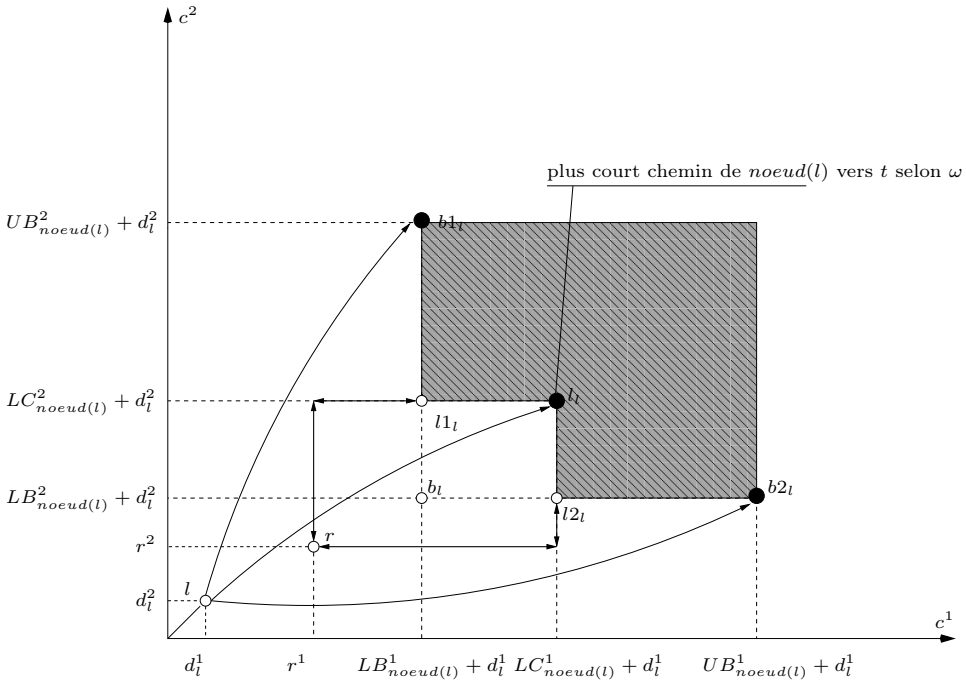


FIGURE 5.2 – Amélioration de l'évaluation de la norme de Tchebycheff - approche par point

Nous définissons les deux points $l1_l$ et $l2_l$ dans l'espace des objectifs, comme sur la figure 5.2. Ces vecteurs de coûts sont définis de la façon suivante : $l1_l = (d_l^1 + LB^1_{nœud(l)}, d_l^2 + LC^2_{nœud(l)})$ et $l2_l = (d_l^1 + LC^1_{nœud(l)}, d_l^2 + LB^2_{nœud(l)})$. Par la suite, nous utiliserons la notation $f(d, r, \omega)$ pour désigner la norme de Tchebycheff entre d et r, même si d n'est pas une étiquette mais simplement un point (d^1, d^2) dans l'espace des objectifs.

Comme il est possible de le voir sur la figure 5.2, à partir de l'étiquette l, il n'est pas possible d'obtenir une solution qui dominerait l_l. Ainsi, il est possible d'utiliser cette information pour améliorer l'évaluation de la norme de Tchebycheff. Pour cela, nous proposons d'utiliser les points $l1_l$ et $l2_l$ qui dominent l'ensemble des solutions atteignables à partir de l. Comme définie dans la proposition 1, une meilleure évaluation de la norme de Tchebycheff d'une étiquette l devient alors $\min\{f(l1_l, r, \omega); f(l2_l, r, \omega)\}$. Nous appelons ceci l'approche par point car l'amélioration est basée sur les propriétés géométriques des deux points $l1_l$ et $l2_l$.

Proposition 1. *Soit z_l défini comme le vecteur de coûts de meilleur compromis qu'il est possible d'atteindre depuis l'étiquette courante l. Alors* $\min\{f(l1_l, r, \omega); f(l2_l, r, \omega)\} \leq f(z_l, r, \omega)$.

PREUVE

$$
\begin{array}{lll}
 & \text{soit} & d^1(l1_l) \leq d^1(z_l) \text{ et } d^2(l1_l) \leq d^2(z_l) \\
 & \text{ou} & d^1(l2_l) \leq d^1(z_l) \text{ et } d^2(l2_l) \leq d^2(z_l) \\
 & & \\
\Leftrightarrow & \text{soit} & d^1(l1_l) - d^1(r) \leq d^1(z_l) - d^1(r) \text{ et } d^2(l1_l) - d^2(r) \leq d^2(z_l) - d^2(r) \\
 & \text{ou} & d^1(l2_l) - d^1(r) \leq d^1(z_l) - d^1(r) \text{ et } d^2(l2_l) - d^2(r) \leq d^2(z_l) - d^2(r) \\
 & & \\
\Leftrightarrow & \text{soit} & \max\{(d^1(l1_l) - d^1(r)); (d^2(l1_l) - d^2(r))\} \\
 & & \quad \leq \max\{(d^1(z_l) - d^1(r)); (d^2(z_l) - d^2(r))\} \\
 & \text{ou} & \max\{(d^1(l2_l) - d^1(r)); (d^2(l2_l) - d^2(r))\} \\
 & & \quad \leq \max\{(d^1(z_l) - d^1(r)); (d^2(z_l) - d^2(r))\} \\
 & & \\
\Leftrightarrow & \text{soit} & f(l1_l, r, \omega) \leq f(z_l, r, \omega) \\
 & \text{ou} & f(l1_2, r, \omega) \leq f(z_l, r, \omega) \\
 & & \\
\Leftrightarrow & & \min\{f(l1_l, r, \omega); f(l2_l, r, \omega)\} \leq f(z_l, r, \omega) \qquad \square
\end{array}
$$

Pour terminer, la complexité de traitement des étiquettes reste la même que pour la méthode BCA*. Le seul coût supplémentaire en termes de temps de calcul reste l'exécution d'une unique recherche mono-objectif (combinaison linéaire des objectifs) avant l'exécution de la méthode BCA*.

5.3.2 Approche par droite pour l'évaluation de la norme de Tchebycheff

Cette amélioration est également basée sur les coûts LC_v^1 et LC_v^2 calculés, avant l'exécution de la méthode BCA*, par une recherche mono-objectif optimisant une combinaison linéaire des objectifs pondérée par ω. Comme nous l'avons vu dans l'amélioration précédente, il n'existe pas de solution pouvant dominer l_l à partir de l'étiquette l. Il est cependant possible d'aller un peu plus loin. En effet, étant donné que les coûts LC_v^1 et LC_v^2 sont calculés par une combinaison linéaire des objectifs, il existe une droite Δ_l (voir figure 5.3) passant par l_l en dessous de laquelle il n'est pas possible d'obtenir une solution non-dominée depuis l'étiquette l. Sur cette droite, la combinaison linéaire des objectifs suivant ω est constante. En effet, s'il existait une solution non-dominée, elle aurait été trouvée lors de la recherche mono-objectif optimisant la combinaison linéaire des objectifs.

La droite Δ_l est ainsi définie par l'ensemble des points $N \in \mathbb{R}^2$ tel que :

$$(N_1 - d_l^1).\omega^1 + (N_2 - d_l^2).\omega^2 = C' \tag{5.7}$$

avec C', une constante définie par l'équation

$$(l_l^1 - d_l^1).\omega^1 + (l_l^2 - d_l^2).\omega^2 = C' \tag{5.8}$$

La droite Δ_r représente l'ensemble des points pour lesquels les deux objectifs sont équilibrés selon le vecteur de préférences ω. La droite Δ_r est ainsi définie par l'ensemble des points $M \in \mathbb{R}^2$ tel que :

$$(M_1 - r_1).\omega^1 = (M_2 - r_2).\omega^2 \tag{5.9}$$

Le point l_r est le point sur la droite Δ_l qui minimise la norme de Tchebycheff, avec le point r correspondant au vecteur de référence. l_r est à l'intersection des droites Δ_l et Δ_r. Il est possible de déterminer ce point à partir des équations 5.7 et 5.9. A noter que si l_r ne se trouve pas dans la zone hachurée des solutions atteignables, nous pouvons utiliser $l1_r$ ou $l2_r$ (comme défini sur la figure 5.3) à la place de l_r.

La figure 5.3 montre l'approche utilisée pour cette amélioration. l est l'étiquette courante, représentée dans l'espace des objectifs. r est le vecteur de coûts de la solution de référence. l_l est le vecteur de coûts correspondant à la solution obtenue en complétant P_l par le sous-chemin de $n\text{œ}ud(l)$ vers t selon l'optimisation de la combinaison linéaire des objectifs. Δ_l et Δ_r sont les droites définies ci-dessus. Sur la figure, les points $l1_l$, $l2_l$, $b1_l$, $b2_l$ et b_l sont également représentés. Ils sont définis de la même façon que dans la sous-section 5.3.1. Sur cette figure, nous pouvons voir que l_r nous permet de calculer une meilleure borne pour la norme de Tchebycheff entre la meilleure solution atteignable depuis l et la solution idéale r.

Il est alors possible d'ordonner l'exploration des étiquettes de la méthode BCA* suivant $f(l_r, r, \omega)$. Contrairement à l'approche précédente, il est ici nécessaire de calculer Δ_l et Δ_r à chaque évaluation d'étiquette. Il parait donc intéressant de comparer ces deux approches par des expérimentations, de manière à déterminer laquelle correspond au meilleur compromis entre précision des bornes et temps de calcul sur chaque étiquette.

5.3.3 Généralisation de l'approche par droite à plus de 2 objectifs

Avec $q = |K|$ objectifs, le problème consiste à déterminer le point d'intersection l_r entre une droite à q dimensions et un hyperplan à q dimensions.

Δ_r est une droite à q dimensions passant par le point de référence r et de vecteur directeur $\overrightarrow{v} \in \mathbb{R}^q$. Le vecteur directeur $\overrightarrow{v}$ correspond aux vecteurs de coûts pour lesquels les objectifs sont équilibrés d'un point de vue du vecteur de poids ω. Cette droite est définie par l'ensemble des points $M \in \mathbb{R}^q$ tel que

$$\forall k \in K, M_k = r_k + C.v_k \tag{5.10}$$

Π_l est un plan à q dimensions passant par le point l_l. Ce plan est défini par l'ensemble des points $N \in \mathbb{R}^q$ tel que

$$\sum_{k \in K} (N_k - d_l^k).\omega^k = C' \tag{5.11}$$

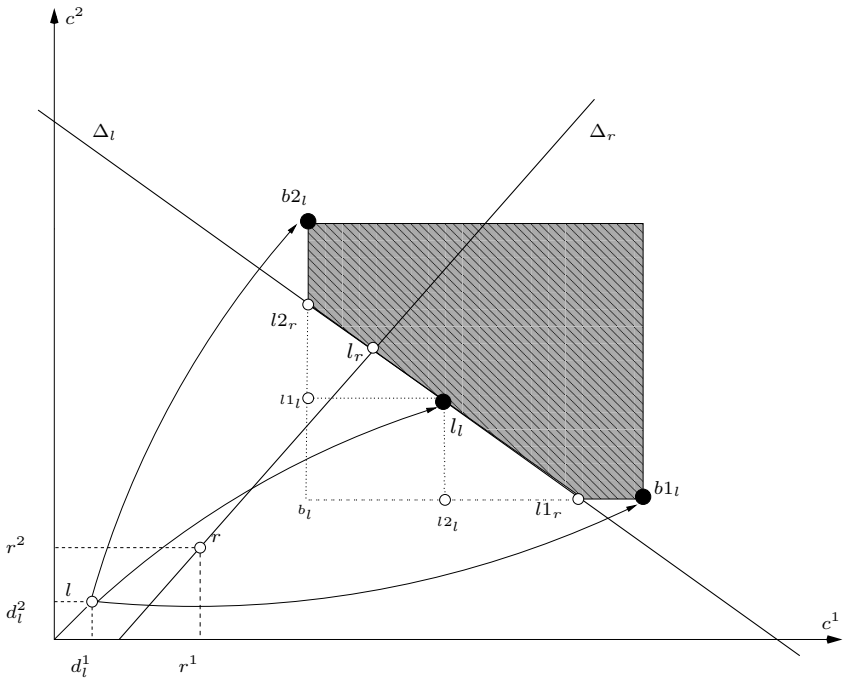


FIGURE 5.3 – Amélioration de l'évaluation de la norme de Tchebycheff - approche par droite

avec C', une constante définie par l'équation

$$\sum_{k \in K} (l_l^k - d_l^k).\omega^k = C' \tag{5.12}$$

Le calcul du vecteur directeur v peut se calculer facilement car nous savons que pour chaque point M appartenant à Δ_r, le produit $(M_k - r_k).\omega^k$ est constant pour tout $k \in K$. Ainsi, en affectant $M_1 = r_1 + 1$, il est possible de calculer

$$M_{k+1} = \frac{\omega^1}{\omega^{k+1}} + r_{k+1}, \forall k \in \{1, 2, ..., K-1\} \tag{5.13}$$

Avec M et r, il est alors possible de calculer

$$v_k = M_k - r_k, \forall k \in K \tag{5.14}$$

Ensuite, nous cherchons la valeur de C.

$$\begin{aligned} \sum_{k \in K} (l_r^k - d_l^k).\omega^k &= C' \\ \sum_{k \in K} (r_k + C.v_k - d_l^k).\omega^k &= C' \\ C &= \frac{C' - \sum_{k \in K}((r_k - d_l^k).\omega^k)}{\sum_{k \in K}(v_k.\omega^k)} \end{aligned} \tag{5.15}$$

Une fois C déterminé, il suffit de calculer la position du point l_r tel que

$$\forall k \in K, l_{r_k} = r_k + C.v_k \tag{5.16}$$

5.4 Expérimentations

Les méthodes présentées dans ce chapitre ont été testées sur les mêmes classes d'instances que celles présentées au chapitre 4. Trois graphes correspondant à des réseaux routiers réels ont été utilisés : Paris, Berlin et la baie de San Francisco. Pour chacun des graphes, 200 instances ont été créées de façon aléatoire et ont été regroupées en quatre classes d'instances, suivant le nombre de solutions non-dominées. Ces instances sont détaillées dans le tableau 4.2 du chapitre 4.

Trois méthodes ont été évaluées :

- la méthode **BCA*** avec l'utilisation de la norme de Tchebycheff,
- l'amélioration **LC-SPA** basée sur l'approche par point pour évaluer la norme de Tchebycheff de la meilleure solution atteignable,
- l'amélioration **LC-LA** basée sur l'approche par droite pour évaluer la norme de Tchebycheff de la meilleure solution atteignable.

Pour chaque instance, le vecteur de poids utilisé correspond à un compromis équilibré entre les différents objectifs. Il est intéressant de noter que les solutions trouvées sont dans 95% des cas des solutions non-supportées, qui n'auraient donc pas pu être déterminées avec une simple combinaison linéaire des objectifs.

Le tableau 5.2 montre les résultats des trois méthodes, du point de vue du nombre d'étiquettes traitées. Comme ces méthodes diffèrent uniquement par l'ordre de traitement des étiquettes (méthodes de *label-correcting*), il est possible qu'une méthode soit plus efficace qu'une autre en moyenne, mais qu'elle soit moins efficace pour certaines instances. Ainsi, les nombres d'étiquettes minimum et maximum sont également indiqués.

Concernant les résultats de la méthode BCA*, les classes d'instances sont, comme pour le chapitre précédent, adaptées à l'évaluation de nos améliorations. Les instances les plus faciles sont résolues en moyenne avec quelques milliers d'étiquettes pour les classes P1, B1 et SF1, et les instances les plus complexes sont résolues en moyenne avec 52 284 étiquettes

		BCA*	LC-SPA		LC-LA	
		Étiquettes	Étiquettes	Amélioration	Étiquettes	Amélioration
P1	moy	651	447	31,34%	254	**60,98%**
	min	17	17	0,00%	16	5,88%
	max	4 507	3 734	17,15%	1 556	65,48%
P2	moy	2 730	1 575	42,31%	629	**76,96%**
	min	189	127	32,80%	77	59,26%
	max	16 299	9 479	41,84%	2 789	82,89%
P3	moy	8 437	5 326	36,87%	1 506	**82,15%**
	min	632	483	23,58%	303	52,06%
	max	46 144	35 996	21,99%	8 389	81,82%
P4	moy	52 284	34 807	33,43%	5619	**89,25%**
	min	4 818	2 802	41,84%	212	95,60%
	max	292 714	194 956	33,40%	31 142	89,36%
B1	moy	2 348	1 541	34,37%	723	**69,21%**
	min	47	29	38,30%	40	14,89%
	max	24 270	15 289	37,00%	4 822	80,13%
B2	moy	13 274	9 937	25,14%	2 599	**80,42%**
	min	467	408	12,63%	181	61,24%
	max	64 291	72 155	**-12,23%**	18 128	71,80%
B3	moy	31 181	17 862	42,72%	3 982	**87,23%**
	min	2 407	906	62,36%	276	88,53%
	max	121 107	90 683	25,12%	24 560	79,72%
B4	moy	101 831	52 841	48,11%	13 939	**86,31%**
	min	7 140	3 142	55,99%	206	97,11%
	max	455 415	339 732	25,40%	89 709	80,30%
SF1	moy	2 346	1 580	32,65%	658	**71,95%**
	min	37	35	5,41%	30	18,92%
	max	12 188	8 220	32,56%	3 288	73,02%
SF2	moy	34 483	23 769	31,07%	5 088	**85,24%**
	min	971	612	36,97%	246	74,67%
	max	124 692	131 774	**-5,68%**	42 259	66,11%
SF3	moy	184 436	116 278	36,95%	12 305	**93,33%**
	min	20 719	10 476	49,44%	266	98,72%
	max	640 010	394 056	38,43%	101 451	84,15%
SF4	moy	1 547 567	1 367 628	11,63%	57 244	**96,30%**
	min	353 467	154 059	56,41%	2 659	99,25%
	max	4 932 972	10 234 060	**-107,46%**	552 910	88,79%

TABLE 5.2 – Résultats : nombres d'étiquettes traitées par BCA* et ses améliorations

pour la classe P4, 101 831 étiquettes pour la classe B4 et 1 547 567 étiquettes pour la classe SF4.

La première amélioration LC-SPA diminue de 11,63% à 48,11% le nombre d'étiquettes traitées. C'est sur la classe d'instances la plus complexe SF4 que les résultats les moins bons sont obtenus. Cela s'explique par des mauvais résultats sur certaines instances : le nombre maximal d'étiquettes passe de 4 932 972 à 10 234 060.

La deuxième amélioration LC-LA est nettement plus efficace. En moyenne, cette méthode permet de réduire le nombre d'étiquettes de 60,98% à 96,30% selon la classe d'instances. Il est intéressant de noter que cette méthode est plus performante sur les classes complexes (89,25% d'amélioration pour la classe P4, 86,31% pour la classe B4 et 96,30% pour la classe SF4) que sur les classes d'instances les plus faciles (60,98% pour la classe P1, 69,21% pour la classe B1 et 71,95% pour la classe SF1). Le nombre maximal d'étiquettes traitées par classe d'instances est également réduit de 65,48% à 89,36%.

Le tableau 5.3 montre les mêmes résultats que le tableau 5.2, du point de vue du temps d'exécution des trois méthodes. Les deux améliorations LC-SPA et LC-LA nécessitent un prétraitement supplémentaire (une recherche mono-objectif inverse optimisant une combinaison linéaire des objectifs pour calculer les coûts LC_v^1 et LC_v^2 pour chaque nœud v) et des calculs en plus à chaque traitement d'une étiquette. Il est donc nécessaire de comparer le temps d'exécution complet entre chacune de ces méthodes pour vérifier si, malgré les traitements supplémentaires, ces améliorations sont efficaces.

La première amélioration LC-SPA est moins rapide que la méthode BCA* classique pour les classes d'instances les plus simples. Ainsi les classes P1, P2, B1, B2 et SF1 sont de 6,45% à 33,88% plus lentes avec LC-SPA qu'avec la méthode BCA*. Cela s'explique principalement par le prétraitement supplémentaire sur l'amélioration LC-SPA et le faible nombre d'étiquettes développées sur ces classes d'instances. A noter tout de même que ces classes d'instances sont résolues en moins de 0,5 seconde en moyenne et ne posent donc pas de problème par rapport à notre contrainte de temps fixée à environ 3 secondes. Pour les classes d'instances les plus difficiles, le temps d'exécution moyen est réduit de 34,61% pour la classe P4, de 47,80% pour la classe B4 et de 10,16% pour la classe SF4. Ce dernier résultat pour la classe SF4 est pénalisé par certaines instances dont le temps de calcul est augmenté avec l'amélioration LC-SPA : le temps maximal augmente en effet de 146,57% pour cette classe d'instances.

Concernant la deuxième amélioration LC-LA, le même constat est fait pour les classes d'instances simples, même si moins de classes sont concernées. Les classes P1, P2, B1 et SF1 voient ainsi leur temps d'exécution augmenter en moyenne de 9,15% à 26,86% par rapport à la méthode BCA*. Les meilleurs résultats sont obtenus encore une fois sur les classes d'instances les plus complexes : le temps d'exécution moyen est ainsi réduit de 81,49% pour la classe P1, de 75,85% pour la classe B4 et de 96,29% pour la classe SF4. En moyenne, l'amélioration LC-LA permet de résoudre toutes les classes d'instances sous la contrainte de temps de 3 secondes, sauf la classe SF4 (4,81 secondes). En analysant le temps maximal, l'amélioration LC-LA ne peut pas résoudre l'ensemble des instances des classes B4 (4,60 secondes), SF3 (4,53 secondes) et SF4 (63,09 secondes). Cependant, mis à part pour la classe SF4 où le temps maximal obtenu est très éloigné de la contrainte de

		BCA*	LC-SPA		LC-LA	
		Temps	Temps	Amélioration	Temps	Amélioration
P1	moy	0,05	0,06	-33,88%	0,06	**-26,86%**
	min	0,01	0,00	100,00%	0,00	100,00%
	max	0,11	0,16	-45,45%	0,14	-27,27%
P2	moy	0,10	0,12	-20,37%	0,11	**-9,15%**
	min	0,02	0,03	-50,00%	0,02	0,00%
	max	0,25	0,18	28,00%	0,15	40,00%
P3	moy	0,18	0,17	3,16%	0,13	**27,54%**
	min	0,06	0,07	-16,67%	0,07	-16,67%
	max	0,91	0,60	34,07%	0,27	70,33%
P4	moy	1,27	0,83	34,61%	0,23	**81,49%**
	min	0,12	0,14	-16,67%	0,11	8,33%
	max	11,58	6,14	46,98%	1,03	91,11%
B1	moy	0,12	0,15	-24,37%	0,14	**-18,44%**
	min	0,01	0,01	0,00%	0,00	100,00%
	max	0,38	0,33	13,16%	0,29	23,68%
B2	moy	0,34	0,36	-6,45%	0,26	**22,07%**
	min	0,13	0,15	-15,38%	0,13	0,00%
	max	1,09	1,25	-14,68%	0,50	54,13%
B3	moy	0,74	0,53	27,98%	0,34	**54,02%**
	min	0,19	0,23	-21,05%	0,18	5,26%
	max	3,47	2,25	35,16%	1,03	70,32%
B4	moy	3,17	1,65	47,80%	0,77	**75,85%**
	min	0,22	0,27	-22,73%	0,21	4,55%
	max	23,10	15,97	30,87%	4,60	80,09%
SF1	moy	0,13	0,17	-26,85%	0,17	**-24,93%**
	min	0,01	0,02	-100,00%	0,02	-100,00%
	max	0,53	0,53	0,00%	0,48	9,43%
SF2	moy	0,92	0,80	13,14%	0,48	**47,17%**
	min	0,15	0,22	-46,67%	0,16	-6,67%
	max	3,94	3,71	5,84%	1,96	50,25%
SF3	moy	6,47	4,20	35,13%	0,92	**85,71%**
	min	0,53	0,29	45,28%	0,20	62,26%
	max	26,36	14,73	44,12%	4,53	82,81%
SF4	moy	129,60	116,43	10,16%	4,81	**96,29%**
	min	12,76	4,80	62,38%	0,50	96,08%
	max	553,31	1 364,32	-146,57%	63,09	88,60%

TABLE 5.3 – Résultats : temps d'exécution pour BCA* et ses améliorations

temps de 3 secondes, les autres résultats sont très encourageants pour la mise en place d'un calculateur sous la forme d'un site web avec des données réelles.

Pour confirmer que les améliorations LC-SPA et LC-LA pouvaient être utiles avec plus de deux objectifs, de nouvelles classes d'instances ont été produites avec trois objectifs sur le graphe de Berlin. Ces classes comportent uniquement 20 instances par classe. Le troisième objectif utilisé est également de type somme et correspond au critère d'effort simplifié. Le coût sur un arc (i, j) est le dénivelé positif en mètres entre les deux nœuds i et j. Même si l'ajout d'un troisième objectif rend la résolution de la plupart des instances impossible sous la contrainte de temps de 3 secondes, il est intéressant de voir si les bons résultats de nos améliorations proposées se confirment avec 3 objectifs.

	BCA*	LC-SPA	LC-LA
B1_3	20	20	20
B2_3	20	20	20
B3_3	20	20	20
B4_3	12	14	**19**

TABLE 5.4 – Résultats : instances résolues avec BCA* et améliorations pour 3 objectifs

Le tableau 5.4 présente le nombre d'instances résolues pour chacune des classes, sous une contrainte de temps de 60 minutes. Dans les classes les plus simples, toutes les instances sont résolues sous la contrainte de temps. Cependant, pour la classe B4_3, la méthode BCA* ne permet de résoudre que 12 instances sur 20. La première amélioration LC-SPA fait un peu mieux avec 14 instances résolues et la deuxième amélioration permet d'en résoudre 19 sur 20.

Le tableau 5.5 compare les temps d'exécution et le nombre d'étiquettes développées en moyenne pour les instances résolues par les trois méthodes.

	Temps d'exécution (sec)			Nombre d'étiquettes		
	BCA*	LC-SPA	LC-LA	BCA*	LC-SPA	LC-LA
B1_3	0,045	0,036	0,017	1 927	1 697	832
B2_3	40,068	16,729	**4,710**	167 562	142 068	**38 261**
B3_3	70,713	41,996	9,250	201 919	155 701	31 339
B4_3	855,203	572,595	101,324	1 627 351	1 277 019	**200 672**

TABLE 5.5 – Résultats : temps d'exécution / nombres d'étiquettes pour BCA* et ses améliorations avec 3 objectifs

D'abord, il est important de constater la différence de temps d'exécution entre les instances simple et difficile : la méthode BCA* permet de résoudre en moyenne les instances de la classe B1_3 en 0,045 seconde contre 14 minutes et 15 secondes en moyenne pour les instances de la classe B4_3. Comme pour les résultats avec 2 objectifs, ces performances sont directement liées au nombre d'étiquettes développées : de 1 927 en moyenne pour les instances de la classe B1_3 à 1 627 351 pour les instances de la classe B4_3.

Avec la méthode LC-SPA, les résultats sont améliorés sur toutes les classes d'instances par rapport à la méthode BCA*. Cependant, c'est avec la méthode LC-LA que les résultats

sont les plus intéressants car le nombre d'étiquettes et le temps d'exécution sont réduits d'un facteur 8 sur les classes les plus difficiles.

5.5 Conclusion du chapitre

Contrairement au chapitre précédent, nous avons ici considéré le problème de la détermination d'une solution de meilleur compromis dans le contexte du problème de plus court chemin multiobjectif. La notion de solution de meilleur compromis est définie par une solution idéale et les préférences de l'utilisateur.

Dans ce chapitre, la méthode BCA* a été présentée et étudiée. Il s'agit d'une méthode de *labelling* qui permet de se concentrer sur la détermination de la solution de meilleur compromis en orientant la recherche vers les étiquettes les plus prometteuses. Plusieurs améliorations basées sur des prétraitements simples ont été proposées. Les expérimentations basées sur des données réelles ont prouvé la performance de ces améliorations, notamment sur les instances les plus difficiles. À la fin de ce chapitre, des expérimentations ont également été menées sur des instances avec 3 objectifs et ont montré que les performances de ces améliorations ne sont pas limitées qu'au contexte bi-objectif.

Ce type d'approche est adapté à notre problème et permet de résoudre des instances de la taille d'une agglomération française comme Paris, sous la contrainte de temps fixée à 3 secondes. Cependant, avec plus de deux objectifs, il est difficile de respecter cette contrainte de temps. De plus, comme nous le verrons dans le chapitre suivant, la prise en compte uniquement de la distance et de la sécurité n'est pas suffisante, et il peut être nécessaire d'augmenter la taille du graphe pour prendre en compte de nouvelles contraintes et objectifs. Une alternative pour respecter cette contrainte de temps peut être alors de s'autoriser, sous certaines conditions, à s'éloigner du front de Pareto.

Chapitre 6

Modélisation avancée d'un réseau routier

Dans ce chapitre, nous nous intéressons à la modélisation avancée d'un réseau routier du point de vue d'un cycliste. Si un graphe classique, c'est-à-dire construit simplement à partir des intersections pour les nœuds et des tronçons de route pour les arcs (cf. section 3.3.1), permet de modéliser des critères basiques comme le temps ou la distance avec des coûts sur les arcs, cela n'est pas suffisant pour des critères plus évolués comme, par exemple, la linéarité de l'itinéraire. L'objectif de ce chapitre est donc d'étudier comment il est possible de modéliser le réseau routier pour prendre en compte de nouveaux critères ou contraintes, tout en gardant les propriétés nécessaires à l'utilisation des algorithmes de plus court chemin classiques ainsi que des méthodes présentées dans les précédents chapitres.

Après l'introduction de plusieurs éléments de modélisation spécifiques au vélo dans la section 6.1, un état de l'art sur les différentes modélisations possibles des réseaux routiers est présenté dans la section 6.2. L'objectif est de pouvoir prendre en compte des manœuvres interdites, des coûts sur les nœuds et des coûts sur des enchaînements d'arcs. Enfin, des expérimentations sont réalisées dans la section 6.3 en utilisant ces nouveaux éléments, couplés à la méthode de calcul de solution de meilleur compromis présentée au chapitre 5.

Ce travail a donné lieu à une communication en conférence nationale [Sauvanet 11].

Données concernant le graphe initial G :	
G :	le graphe
V :	l'ensemble des nœuds
A :	l'ensemble des arcs
$n = \|V\|$:	le nombre de nœuds
$m = \|A\|$:	le nombre d'arcs
s :	le nœud de départ
t :	le nœud de destination
p :	un chemin
$\Gamma^-(v)$:	ensemble des nœuds prédécesseurs du nœud v
$\Gamma^+(v)$:	ensemble des nœuds successeurs du nœud v
$\delta(v)$:	le degré du nœud v
$deb(a)$:	le nœud source de l'arc a
$fin(a)$:	le nœud destination de l'arc a
$c(p)$:	le vecteur de coûts associé au chemin p
$c_A(u, v)$:	le coût associé à l'arc (u, v)
$c_V(v)$:	le coût associé au nœud n
$c_{A\to A}(a_1, a_2)$:	le coût associé à l'enchaînement d'arcs a_1 puis a_2

Données concernant le graphe adjoint G_A :	
G_A :	le graphe
V_A :	l'ensemble des nœuds
A_A :	l'ensemble des arcs
$n_A = \|V_a\|$:	le nombre de nœuds
$m_A = \|A_a\|$:	le nombre d'arcs
$c_{A_A}(u, v)$:	le coût associé à l'arc (u, v)

TABLE 6.1 – Table des notations du chapitre 5

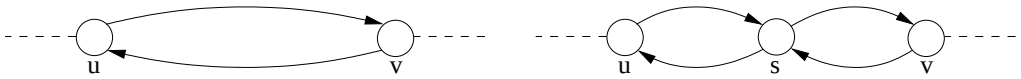
6.1 Éléments de modélisation spécifiques au vélo

Comme nous l'avons vu dans la section 3.3.1, la modélisation classique du réseau routier sous la forme d'un graphe $G = (V, A, c_A)$, c'est-à-dire associant les tronçons de route à des arcs et les intersections à des nœuds, n'est pas toujours suffisante. Actuellement, celle-ci ne prend en compte que des coûts sur les arcs en utilisant une fonction de coût c_A qui associe à chaque arc un vecteur de coûts. Dans la suite de ce chapitre, nous parlerons de *graphe initial* pour désigner un réseau routier représenté avec cette modélisation classique.

Dans cette section, différentes spécificités propres au déplacement à vélo sont présentées et modélisées.

6.1.1 Source et destination quelconques sur un tronçon

Les algorithmes de plus court chemin ne permettent de calculer des chemins que de nœud à nœud alors que pour tout usager, comme un cycliste, il est plus pratique de se déplacer entre deux points situés à n'importe quel endroit du réseau, c'est-à-dire sur des points appartenant à des tronçons de route correspondant à des arcs.



(a) Graphe initial (b) Graphe modifié

FIGURE 6.1 – Modélisation d'un point de départ quelconque

La manière de procéder la plus simple consiste à découper les arcs concernés en y ajoutant des nœuds temporaires correspondant aux positions exactes du départ et de la destination. La figure 6.1 montre les modifications nécessaires pour spécifier un point de départ quelconque sur le tronçon de route entre les nœuds u et v :

- suppression des arcs (u, v) et (v, u),
- ajout du nœud s correspondant à la position exacte du point de départ,
- ajout des arcs entre le nœud s et les nœuds u et v.

Concernant les coûts sur les nouveaux arcs, ces derniers sont calculés en fonction de la position de s sur le tronçon de route et de façon à ce que les coûts des arcs supprimés soient conservés : $c_A(u, s) + c_A(s, v) = c_A(u, v)$ et $c_A(v, s) + c_A(s, u) = c_A(v, u)$.

Cet élément de modélisation ne pose donc pas de problème et peut être utilisé avec le graphe initial et les algorithmes présentés dans les précédents chapitres.

6.1.2 Coûts sur les nœuds

Il peut être intéressant d'intégrer des coûts sur les nœuds. Le premier exemple concerne le temps de parcours, qui comme nous l'avons vu dans la section 2.2 du chapitre 2, peut se

révéler plus pertinent que la distance de l'itinéraire car ce critère peut prendre en compte un nombre d'éléments plus important jouant un rôle dans la notion d'itinéraire direct. Ce dernier peut être caractérisé par des coûts sur les arcs, modélisant le temps de parcours des tronçons de route, et des coûts sur les nœuds, modélisant le temps d'arrêt aux intersections. Ce modèle permet ainsi de prendre en compte des temps d'arrêt différents pour des feux tricolores, des stops, cédez-le-passage, etc.

Un autre exemple de l'utilité des coûts sur les nœuds concerne le critère de sécurité. Jusqu'ici nous n'avons considéré que des coûts sur les arcs. Cependant, il peut être pertinent d'intégrer des coûts d'insécurité sur les nœuds correspondants par exemple à des intersections dangereuses.

S'il est simple d'ajouter une fonction c_V à notre graphe initial qui associe à chaque nœud un vecteur de coûts, le problème est que les algorithmes de *labelling* ne prennent pas en compte ce type de coûts. En effet, les algorithmes de *labelling* n'utilisent que des coûts sur les arcs. Il est alors nécessaire, soit d'adapter les algorithmes, soit de modifier le graphe.

6.1.3 Manœuvres interdites

Avec les modèles classiques de graphe obtenus directement à partir du réseau routier, un autre problème se pose pour modéliser des manœuvres interdites. En effet, il arrive sur certaines intersections qu'un enchaînement de certains tronçons de route ne soit pas possible. Par exemple, comme le montre la figure 6.2, sur des intersections importantes, tourner à gauche peut être interdit. Notons que dans le contexte d'itinéraire à vélo sécurisé, la prise en compte de ces manœuvres interdites est important car celles-ci sont potentiellement sources d'accidents.

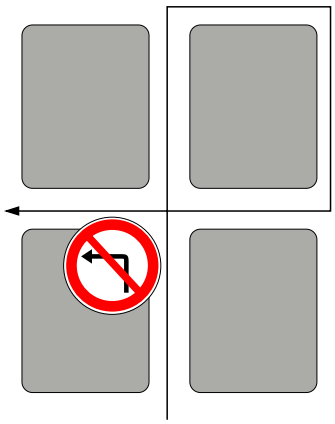


FIGURE 6.2 – Présentation d'une manœuvre interdite

En utilisant la modélisation initiale, plusieurs problèmes se posent. Tout d'abord, comment modéliser cette information qui concerne un enchaînement d'arcs et non un élément, c'est-à-dire un arc ou un nœud, du graphe. Ensuite, les manœuvres interdites peuvent impliquer, comme sur la figure 6.2, de passer à deux reprises sur le même nœud. Sur cette

figure, il est interdit de tourner à gauche à l'intersection, ce qui oblige à faire un détour pour revenir sur l'intersection par une autre voie. Pourtant, passer à plusieurs reprises sur le même nœud n'est pas possible avec un algorithme de *labelling*.

6.1.4 Coûts sur les enchaînements d'arcs

Plus généralement, en plus des coûts sur les nœuds et les arcs, il peut être intéressant d'affecter des coûts sur des enchaînements d'arcs. Il peut s'agir, par exemple, de modéliser plus finement le critère de temps de parcours. Ainsi à une intersection importante, il peut être intéressant de prendre en compte le temps nécessaire pour traverser cette intersection, qui dépend fortement de l'enchaînement des arcs. Il est logique que d'aller tout droit à une intersection sera souvent plus rapide que, de tourner à gauche, de la même façon qu'il est souvent plus facile de tourner à droite qu'à gauche à cause de la circulation du sens opposé qu'il faut traverser. Notons que dans ce cas, la sécurité peut également être impactée.

L'un des problèmes lorsque l'on optimise des objectifs comme la distance ou la sécurité à vélo est que les itinéraires calculés comportent de nombreux changements de direction et sont donc souvent peu pertinents pour les cyclistes. Un autre exemple concerne donc un nouveau critère lié à la linéarité qui rend compte de la simplicité de l'itinéraire pour le cycliste. Il peut s'agir de minimiser les angles des intersections, le nombre de changements de types de voies, etc. Il est alors nécessaire de pouvoir définir des coûts liés à des enchaînements d'arcs.

Ces deux exemples, temps nécessaire pour traverser une intersection et linéarité de l'itinéraire, nécessitent de pouvoir définir des coûts sur des enchaînements d'arcs, alors que le graphe utilisé ne permet pas de définir ce genre de coûts et il n'est également pas possible de les prendre en compte dans des algorithmes de plus court chemin.

6.2 Modélisations avancées d'un réseau routier

Les modélisations présentées dans cette section visent à prendre en compte les éléments introduits dans la section précédente :

- les coûts sur les nœuds,
- les enchaînements d'arcs interdits,
- les coûts sur les enchaînements d'arcs.

6.2.1 Doublement des nœuds

Une solution pour intégrer les coûts sur les nœuds est par exemple de transformer le graphe initial pour obtenir un nouveau graphe avec des coûts uniquement sur les arcs. Une première méthode possible est de doubler le nombre de nœuds pour insérer un nouvel arc, entre les nœuds initiaux et les nouveaux nœuds, qui permet d'intégrer les coûts sur les nœuds. Ainsi, pour chaque nœud u :

- un nœud u' est créé,
- l'ensemble des arcs (u, v) sont déplacés en (u', v),
- un nouvel arc (u, u') est créé et les coûts du nœud u sont associés à ce nouvel arc.

Plus directement, il est possible d'intégrer les coûts d'un nœud à l'ensemble des arcs incidents vers l'intérieur de ce nœud, c'est-à-dire que $\forall a \in A, c_A(a) = c_A(a) + c_v(fin(a))$. Pour calculer un plus court chemin du nœud s au nœud t, il convient alors de modifier temporairement le graphe pour supprimer les coûts du nœud t sur l'ensemble des arcs incidents vers l'intérieur de t, de manière à ne pas prendre en compte des coûts associés au nœud t dans le calcul des chemins de s à t.

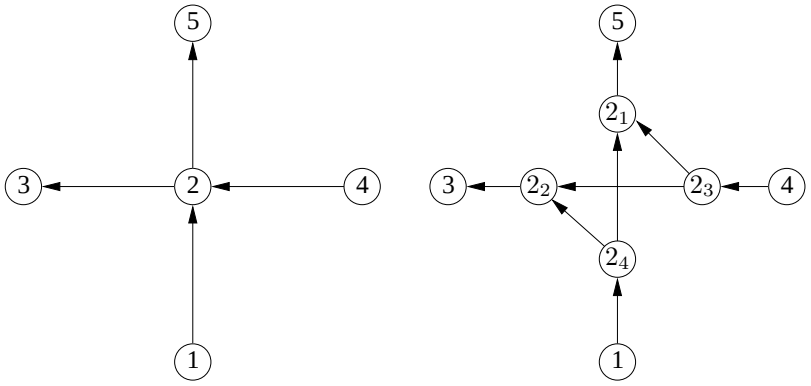
Cette première méthode permet de prendre en compte des coûts sur les nœuds, mais ne permet pas de gérer les manœuvres interdites ni les coûts sur les enchaînements d'arcs.

6.2.2 Méthode *node expansion*

Une solution pour modéliser l'ensemble de ces nouveaux éléments est la méthode *node expansion* [Añez 96], qui consiste à augmenter le nombre de nœuds et à ajouter de nouveaux arcs pour modéliser les enchaînements possibles entre les anciens arcs.

Concrètement, pour un graphe $G = (V, A)$:

- pour chaque nœud $v \in V$, v est remplacé par $\delta(v) = |\Gamma^-(v)| + |\Gamma^+(v)|$ nœuds,
- les anciens arcs sont conservés, et $|\Gamma^-(v)| \times |\Gamma^+(v)|$ nouveaux arcs sont créés pour représenter les enchaînements possibles.



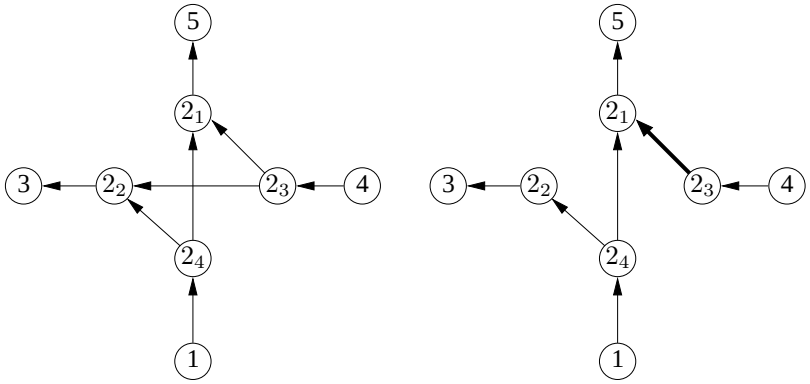
(a) Graphe initial (b) Graphe transformé

FIGURE 6.3 – Transformation d'un graphe avec la méthode *node expansion*

Un exemple est montré sur la figure 6.3 où la méthode est appliquée sur le nœud 2 du graphe 6.3a. Quatre nœuds remplacent le nœud 2 : les nœuds 2_1 et 2_2 sont reliés aux arcs incidents vers l'extérieur de 2 et les nœuds 2_3 et 2_4 sont reliés aux arcs incidents vers l'intérieur du nœud 2. Entre ces nouveaux nœuds, quatre arcs sont créés pour modéliser

les enchaînements possibles. Par exemple, l'arc $(2_4, 2_2)$ dans le nouveau graphe autorise l'enchaînement d'arcs $(1, 2)$ puis $(2, 3)$ dans le graphe initial.

En ce qui concerne les coûts, ceux sur les arcs c_A du graphe initial sont recopiés sur les arcs conservés du nouveau graphe. Les coûts sur les nœuds c_V du graphe initial sont recopiés sur les nouveaux arcs créés puisqu'ils correspondent à des nœuds. Enfin, les coûts sur les enchaînements d'arcs $c_{A \to A}$ peuvent également être affectés sur ces nouveaux arcs créés.



(a) Graphe transformé

(b) Manœuvre interdite (4,2) puis (2,3) ; coût sur enchaînement d'arcs (4,2) puis (2,5)

FIGURE 6.4 – Prise en compte de nouveaux éléments avec la méthode node expansion

Avec cette méthode, il est donc possible d'affecter des coûts sur des enchaînements d'arcs : par exemple la figure 6.4 montre que l'on peut modéliser un coût sur l'enchaînement d'arcs $(4, 2)$ puis $(2, 5)$ en affectant un coût sur l'arc $(2_3, 2_1)$. De la même manière, il est tout à fait possible de modéliser une manœuvre interdite en supprimant certains arcs. Par exemple, toujours sur la figure 6.4, la manœuvre $(4, 2)$ puis $(2, 3)$ est interdite sur le graphe initial. Il suffit alors de supprimer l'arc $(2_3, 2_2)$ sur le nouveau graphe. Enfin, les coûts sur les nœuds du graphe initial, comme par exemple sur le nœud 2, peuvent être affectés sur les nouveaux arcs créés, c'est-à-dire les arcs $(2_4, 2_2)$, $(2_4, 2_1)$ et $(2_3, 2_1)$.

Cette méthode présente néanmoins plusieurs inconvénients. Tout d'abord, le graphe possède alors deux types d'arcs avec des coûts différents : des arcs représentant des tronçons de route et des arcs logiques représentant le fait de pouvoir ou non enchaîner plusieurs tronçons de route. Ensuite, dans [Añez 96], les auteurs montrent que ces modifications de graphe peuvent augmenter significativement la taille du graphe : le nombre de nœuds peut être triplé et le nombre d'arcs doublé. Ainsi, si la modification ne concerne que certains nœuds, de manière à gérer par exemple des manœuvres interdites, cette méthode peut être pertinente. En revanche, s'il s'agit de transformer tout le graphe pour affecter des coûts aux enchaînements d'arcs, la taille du graphe va augmenter de façon importante.

Pour un graphe $G = (V, A)$, dans [Winter 02], l'auteur montre que la taille du nouveau graphe $G_E = (V_E, A_E)$ obtenu avec cette méthode peut être estimée par :

– $|V_E| = \delta_{max}|V|$ nœuds avec δ_{max} le degré maximum des nœuds du graphe,

– $|A_E| = (1 + \frac{\delta_{max}}{2})|A|$ arcs.

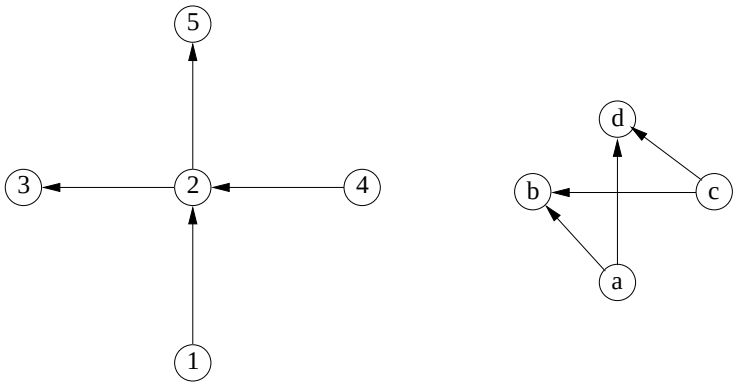
6.2.3 Notion de graphe adjoint

Il existe une autre représentation du graphe routier qui permet de modéliser les éléments présentés ci-dessus en limitant l'augmentation de la taille du graphe. Il s'agit de transformer le graphe initial sous la forme d'un graphe dit adjoint. Cette représentation a l'avantage de pouvoir définir des coûts sur les nœuds, sur les arcs et sur des enchaînements précis d'arcs. Intuitivement, elle consiste à intervertir les nœuds et les arcs : les arcs du graphe initial deviennent les nœuds du graphe adjoint et les arcs du graphe adjoint correspondent aux enchaînements possibles entre les arcs du graphe initial.

L'utilisation de représentations sous la forme de graphes adjoints a été initialement proposée pour des preuves sur la connexité des graphes non-orientés [Whitney 32] et a été réutilisée à plusieurs reprises et sous différents noms : *pseudo-network* [Caldwell 61], graphe adjoint [Harary 69] et graphe dual ou dual linéaire [Añez 96, Winter 01]. Cependant, seul le terme de graphe adjoint [Harary 69] nous semble correct et correspond à cette représentation.

Définition 6. *Pour un graphe initial G défini par $G = (V, A)$ avec V l'ensemble des nœuds et A l'ensemble des arcs, il est possible de construire le graphe adjoint $G_A = (V_A, A_A)$ en suivant deux étapes :*

1. *l'ensemble des arcs A du graphe initial devient l'ensemble des nœuds V_A du graphe adjoint,*
2. *un arc (u', v') de l'ensemble A_A du graphe adjoint est créé pour un enchaînement possible entre deux arcs (u, v) et (v, w) du graphe initial.*



(a) Graphe initial (b) Graphe adjoint

FIGURE 6.5 – Transformation d'un graphe initial vers son graphe adjoint

Cette transformation est illustrée sur la figure 6.5 où le graphe initial (nœuds 1, 2, 3, 4, 5) est représenté à gauche et le graphe adjoint (nœuds a, b, c, d) correspondant est représenté à droite. Sur cette figure, les nœuds c et d représentent les arcs $(4, 2)$ et $(2, 5)$ du graphe initial et l'arc (c, d) représente la possibilité l'enchaînement des arcs $(4, 2)$ et $(2, 5)$ du graphe initial.

La modélisation des manœuvres interdites est relativement simple car un arc du graphe adjoint modélise l'enchaînement successif de deux arcs : il suffit de simplement supprimer l'arc en question du graphe adjoint pour interdire cette manœuvre. Par exemple, sur la figure 6.5, si l'enchaînement d'arcs $(4, 2)$ puis $(2, 3)$ est interdit sur le graphe initial, il suffit de supprimer l'arc (c, b) du graphe adjoint.

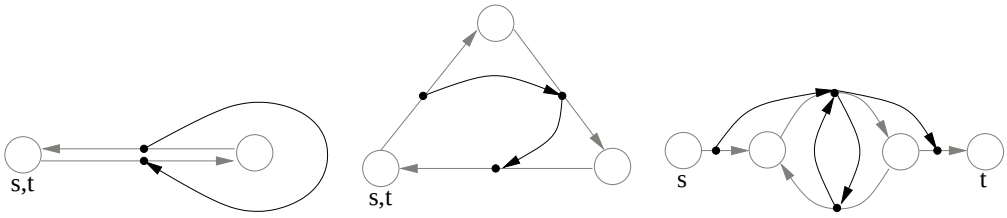
En ce qui concerne les coûts, ceux sur les enchaînements d'arcs $c_{A\to A}$ sont les plus simples à gérer puisque chaque arc du graphe adjoint représente justement un enchaînement d'arcs. Par exemple, sur la figure 6.5, si l'enchaînement d'arcs $(4, 2)$ puis $(2, 5)$ du graphe initial est dangereux, il suffit d'affecter un coût d'insécurité sur l'arc (c, d) du graphe adjoint.

De la même manière que pour la méthode de *node expansion*, les coûts sur les nœuds c_V peuvent être affectés sur les arcs du graphe adjoint car ces derniers représentent le fait de traverser un nœud du graphe initial. Par exemple, sur la figure 6.5, si un feu tricolore est présent sur le nœud 2 du graphe initial, il suffit d'affecter un coût d'insécurité sur les arcs (a, b), (a, d), (c, b) et (c, d) du graphe adjoint.

Concernant les coûts c_A sur les arcs du graphe initial, une possibilité est de les affecter sur les arcs du graphe adjoint, qui mènent à ces arcs du graphe initial.

Pour résumer, deux arcs (u, v) et (v, w) du graphe initial sont transformés en un seul arc (uv, vw) sur le graphe adjoint de coût :

$$c_{A_A}((uv), (vw)) = c_V(v) + c_{A\to A}((u, v), (v, w)) + c_A(v, w) \tag{6.1}$$



(a) Demi-tour (b) Boucle (c) Boucle impossible

FIGURE 6.6 – Exemples de modélisations possibles avec des graphes adjoints

Dans [Winter 02], l'auteur présente plusieurs exemples qui montrent l'intérêt d'un réseau routier modélisé sous la forme d'un graphe adjoint. Ces exemples sont illustrés sur la figure 6.6 où les nœuds s et t représentent les nœuds source et destination, les grands cercles blancs représentent les nœuds du graphe initial et les petits cercles noirs représentent les nœuds du graphe adjoint. Le premier exemple montre qu'il est possible de modéliser le fait de faire demi-tour sur un tronçon de route. Cela peut être intéressant pour calculer des itinéraires, pour les piétons par exemple, mais n'est pas réalisable pour d'autres modes comme la voiture ou le vélo, qui doivent suivre le code de la route. Pour résoudre ce problème, il est possible d'utiliser le graphe adjoint réduit, qui correspond au graphe adjoint auquel on a retiré les arcs correspondant à des enchaînements d'arcs dans le graphe initial du type (u, v) puis (v, u). Le deuxième exemple montre qu'un chemin de type boucle, qui ne peut pas être calculé avec les algorithmes classiques de plus court chemin sur le graphe initial se traduit par un chemin qui n'est pas une boucle sur le graphe adjoint. Enfin, le

dernier exemple montre que les boucles qui passent par un même arc sur le graphe initial ne seront pas prises en compte par le graphe adjoint avec un algorithme de plus court chemin.

Pour un graphe $G = (V, A)$, dans [Winter 02], l'auteur montre que la taille du nouveau graphe $G_A = (V_A, A_A)$ obtenu avec cette méthode peut être estimée par :

- $|V_A| = |A|$ nœuds,
- $|A_A| = \frac{\delta_{max}}{2}|A|$ arcs.

La taille du graphe adjoint d'un graphe initial est donc moins importante que celle du modèle avec la méthode *node expansion*. Le tableau 6.2 présente l'évolution de la taille des graphes utilisés dans les chapitres précédents, en fonction de la modélisation utilisée : modèle initial, modèle avec nœuds étendus, graphe adjoint complet et réduit.

	Paris	Berlin	San Francisco
Modèle initial	29 086 nœuds 64 538 arcs	59 673 nœuds 145 840 arcs	174 975 nœuds 435 959 arcs
node expansion	129 076 nœuds 227 651 arcs	291 680 nœuds 551 892 arcs	871 918 nœuds 1 697 882 arcs
graphe adjoint complet	64 538 nœuds 163 113 arcs	145 840 nœuds 406 052 arcs	435 959 nœuds 1 261 923 arcs
graphe adjoint réduit	64 538 nœuds 118 887 arcs	145 840 nœuds 274 892 arcs	435 959 nœuds 848 335 arcs

TABLE 6.2 – Résultats : taille des graphes selon plusieurs modélisations

Calculer un chemin du nœud s vers le nœud t sur le graphe initial ne se traduit pas par un calcul de chemin de nœud à nœud sur le graphe adjoint correspondant car les nœuds du graphe initial ne se retrouvent pas sur le graphe adjoint. En fait, dans [Winter 02], l'auteur montre que sur le graphe adjoint, le problème devient un calcul de chemin entre un ensemble de nœuds sources et un ensemble de nœuds de destination. Les nœuds sources sur le graphe adjoint correspondent aux arcs incidents vers l'extérieur du nœud s sur le graphe initial, et les nœuds de destination sur le graphe adjoint correspondent aux arcs incidents vers l'intérieur du nœud t sur le graphe initial. Ces deux ensembles d'arcs dans le graphe initial se traduisent par deux ensembles de nœuds $V_{A_s} \in V_A$ et $V_{A_t} \in V_A$ du graphe. Ce principe est illustré sur la figure 6.7 où un graphe initial est représenté en gris avec le nœud de départ s et le nœud d'arrivée t, et le graphe adjoint correspondant est représenté en noir avec les nœuds V_{A_s} et V_{A_t} correspondants.

Pour revenir à un problème classique de plus court chemin de nœud à nœud, il est alors très simple d'ajouter deux nœuds supplémentaires s' et t' puis d'ajouter des arcs reliant le nœud s' aux nœuds de l'ensemble V_{A_s} et des arcs reliant les nœuds de l'ensemble V_{A_t} au nœud t'.

Cependant, comme le montre l'équation 6.1, un arc $((uv), (vw))$ du graphe adjoint n'intègre pas le coût $c_A(u, v)$ du graphe initial. Pour résoudre ce problème, il est possible d'intégrer ces coûts sur les arcs supplémentaires reliant le nœud s' aux nœuds de l'ensemble

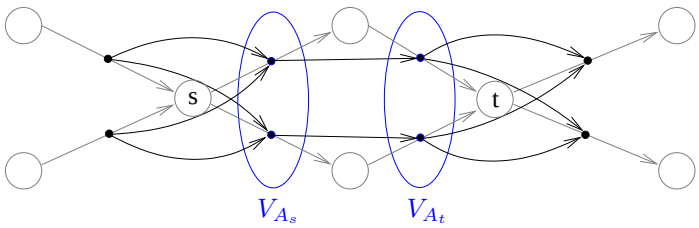


FIGURE 6.7 – Correspondance d'une instance (s,t) sur un graphe et son graphe adjoint

V_{A_s}.

Avec la modélisation du réseau routier sous la forme d'un graphe adjoint, il est possible d'intégrer des coûts à la fois sur les arcs, sur les nœuds, ainsi que sur les enchaînements d'arcs, comme le montre l'équation 6.1.

6.3 Expérimentations

Une première série de tests préliminaires a été effectuée avec deux objectifs, la distance et la sécurité, sur les mêmes instances utilisées que dans les chapitres 4 et 5, afin de confirmer que les chemins calculés sur le graphe adjoint correspondaient bien aux mêmes chemins que ceux calculés sur le graphe initial.

Par la suite, des expérimentations ont été réalisées sur des données réelles et avec des objectifs différents, pour tester l'intérêt du graphe adjoint, pour prendre en compte des coûts sur les nœuds et sur les enchaînements d'arcs dans le graphe initial. Pour cela, plusieurs objectifs ont été testés. La sécurité a été conservée car elle représente un des critères les plus déterminants dans le choix d'un itinéraire adapté aux cyclistes. Cependant, les coûts d'insécurité sur les nœuds, bien que pertinents, n'ont pas été intégrés, faute de données à disposition.

Le deuxième objectif est le temps de parcours, mesurable par des coûts sur les arcs représentant le temps nécessaire pour les parcourir et des coûts sur les nœuds représentant le temps d'attente à certaines intersections, comme les feux tricolores. Une réduction de vitesse, lorsque le cycliste emprunte des voies partagées avec les piétons, a également été prise en compte.

Le dernier objectif est la linéarité de l'itinéraire. Cet objectif peut se modéliser de plusieurs façons très différentes. Dans [Richter 08], l'auteur présente un modèle où des coûts sont associés à chaque changement de direction, en différenciant par exemple le fait de tourner à gauche à une intersection où il n'est possible d'aller qu'à gauche ou à droite et le fait de tourner à gauche alors qu'il est également possible d'aller tout droit. Le nombre d'embranchements sur une intersection est donc pris en compte. L'auteur propose également une méthodologie pour présenter les changements de direction de façon contextuelle à partir d'éléments géographiques, tels que des points d'intérêts, des cours d'eau, des voies ferrées, etc.

Dans [Winter 02], l'auteur aborde plusieurs façons de modéliser ce critère de linéarité

(a) Chemin de compromis entre temps et sécurité

(b) Chemin linéaire minimisant le nombre de changements de rues

FIGURE 6.8 – Chemin de compromis temps/sécurité comparé au chemin linéaire

dans un contexte de calcul d'itinéraires pour les piétons. L'auteur présente plusieurs possibilités : minimiser le nombre de changements de direction, minimiser l'angle maximal, etc. Minimiser le nombre de changements de direction semble pertinent mais reste complexe à calculer. En effet, à partir d'un graphe routier, il n'est pas aisé de connaître quelles manœuvres correspondent à des changements de direction. Par exemple, tourner à gauche à une intersection peut être la continuation d'une même rue et ne pas constituer une réelle coupure dans l'itinéraire de l'utilisateur.

Le modèle retenu ici est très simple : il consiste à minimiser le nombre de changements de rues dans l'itinéraire. Le graphe adjoint permet alors d'associer un coût positif égal à un sur les enchaînements d'arcs où le nom de la rue change, et un coût nul si le nom de la rue est identique. Ce modèle n'est pas parfait, et ne tient par exemple pas compte des grands axes routiers pour lesquels le nom de la voie peut changer, alors qu'il n'y a pas de changement de direction réel. La figure 6.8 montre un exemple d'itinéraire où un compromis équilibré entre temps et sécurité correspond à un itinéraire avec de nombreux changements de direction et donc difficilement exploitable pour un cycliste. Le deuxième itinéraire, minimisant le nombre de changements de rues est, lui, beaucoup plus linéaire et également plus simple à retenir.

Ces trois objectifs permettent de couvrir l'ensemble des nouvelles possibilités offertes par la modélisation d'un réseau routier sous la forme d'un graphe adjoint : coûts sur les arcs (temps et sécurité), coûts sur les nœuds (temps) et coûts sur les enchaînements d'arcs (linéarité).

Contrairement aux expérimentations réalisées dans les chapitres 4 et 5, nous nous plaçons ici dans des conditions de test proches de celles du serveur de production. L'outil de gestion de graphe utilisé est la bibliothèque libre Boost (http://www.boost.org) car la bibliothèque LEDA nécessite une licence entreprise payante pour pouvoir être utilisée en production. Les performances obtenues avec Boost sont moins bonnes qu'avec LEDA, notamment en ce qui concerne les recherches mono-objectif. Pour cette raison et également parce que les données utilisées dans ce chapitre sont plus à jour que les données précédentes (les données provenant d'OpenStreetMap évoluent et s'améliorent chaque jour), les résultats obtenus concernant le temps de calcul ne peuvent donc pas être comparés aux résultats obtenus dans le chapitre 5.

Les tests ont été réalisés sur le graphe de Paris mis à jour (31 258 nœuds et 67 745 arcs contre 29 086 nœuds et 64 538 arcs pour le graphe de Paris utilisé dans les chapitres 4 et 5) où la cyclabilité a été renseignée sur la plupart des voies par un collecteur de terrain. Les classes de tests P1, P2, P3 et P4 sont les mêmes que pour les chapitres précédents, mais avec 3 objectifs (temps, sécurité et linéarité). Notons que sur les 200 instances, seulement quatre solutions de meilleur compromis trouvées correspondent à des solutions supportées.

Le tableau 6.3 présente le nombre d'étiquettes minimum, maximum et moyen développées sur les quatre classes d'instances et avec deux méthodes : BCA* et l'amélioration LC-LA la plus performante présentée dans le chapitre 5. Comme dans les expérimentations du chapitre 5, la méthode LC-LA permet toujours de réduire le nombre d'étiquettes développées. Cette méthode reste plus efficace sur les instances difficiles que sur les instances faciles. Ainsi, le nombre d'étiquettes moyen est réduit de 68,09% sur la classe P1 et de

		BCA*	LC-LA	
		Étiquettes	Étiquettes	Amélioration
P1	moy	1 871	597	68,09%
	min	23	23	0,00%
	max	22 772	4 397	80,69%
P2	moy	8 702	2235	74,32%
	min	209	183	12,44%
	max	81 498	18 612	77,16%
P3	moy	19 064	3960	**79,23%**
	min	327	166	49,24%
	max	99 135	17 025	**82,83%**
P4	moy	77 456	7320	**90,55%**
	min	1749	244	86,05%
	max	855 892	31 950	**96,27%**

TABLE 6.3 – Résultats sur graphe adjoint : nombres d'étiquettes BCA* et amélioration

90,55% sur la classe P4.

		BCA*	LC-LA	
		Temps	Temps	Amélioration
P1	moy	0,69	0,76	-9,54%
	min	0,03	0,05	-66,67%
	max	1,83	1,5	-34,13%
P2	moy	1,30	1,19	8,57%
	min	0,29	0,31	-6,90%
	max	8,05	2,16	73,17%
P3	moy	1,93	1,28	33,58%
	min	0,86	0,89	-3,49%
	max	9,65	1,87	80,62%
P4	moy	12,32	**1,66**	86,54%
	min	1,00	1,14	-14,00%
	max	317,62	4,40	98,61%

TABLE 6.4 – Résultats sur graphe adjoint : temps d'exécution BCA* et amélioration

Le tableau 6.4 présente les mêmes résultats mais en comparant cette fois-ci le temps d'exécution. L'amélioration LC-LA est peu efficace sur les instances de petite taille comme les classes P1 ou P2. Cela s'explique par la recherche mono-objectif supplémentaire qui optimise une combinaison linéaire des objectifs, ainsi que par des calculs supplémentaires à chaque étiquette. Cependant, sur des instances de taille importante, cette amélioration est très efficace : le temps moyen d'exécution est réduit de 33,58% sur la classe P3 et de 86,54% sur la classe P4.

Malgré l'augmentation de la taille du graphe liée à l'utilisation du graphe adjoint et la prise en compte de trois objectifs différents, le calcul d'une solution de meilleur compromis reste rapide, même sur la classe d'instances la plus difficile : 1,66 secondes en moyenne et

4,40 secondes au maximum.

6.4 Conclusion du chapitre

Dans ce chapitre, nous avons montré que, dans la pratique, la modélisation du réseau routier sous la forme d'un graphe classique n'est pas toujours adaptée au calcul de chemins. En effet, en plus de coûts sur des nœuds, la prise en compte de la notion d'enchaînement d'arcs peut être nécessaire pour y associer des coûts ou pour interdire certaines manœuvres. Un graphe construit classiquement à partir d'un réseau routier n'est alors pas adapté et le calcul de chemins nécessite soit de modifier la structure du graphe, soit d'adapter les algorithmes de plus court chemin. Deux transformations de graphe ont été présentées : l'expansion de nœuds et la représentation sous la forme du graphe adjoint. Par rapport à la méthode d'expansion de nœuds, le graphe adjoint est préférable car l'augmentation de la taille du graphe est moins importante. Le graphe adjoint ne dispose que de coûts sur les arcs alors qu'il représente un graphe routier initialement constitué de coûts sur les nœuds, sur les arcs et sur des enchaînements d'arcs.

De nouveaux objectifs ont été modélisés afin de tester le calcul de chemin dans un réseau routier représenté sous la forme d'un graphe adjoint. L'objectif de minimisation de la durée du trajet prend en compte à la fois le temps de parcours des arcs ainsi que le temps d'attente sur des nœuds où des feux tricolores sont présents. Un objectif de linéarité de l'itinéraire a également été proposé. Ce dernier minimise le nombre de changements de rues, en affectant des coûts sur des enchaînements d'arcs. Des expérimentations de calculs de solutions de meilleur compromis avec trois objectifs (temps, sécurité et linéarité) sur le graphe adjoint ont montré que cette modélisation était suffisamment performante pour être utilisée sur un site web à l'échelle d'une grande agglomération.

Chapitre 7

Application : Géovélo

Dans le chapitre 3, le problème du plus court chemin multiobjectif a été présenté et une modélisation a été proposée. Dans les chapitres 4 et 5, deux possibilités ont été abordées :

- résoudre le problème par une méthode *a posteriori* : c'est-à-dire que toutes les solutions non-dominées sont calculées et que quelques unes sont ensuite proposées à l'utilisateur,
- utiliser une méthode *a priori* : les préférences de l'utilisateur sont utilisées pour calculer une solution de meilleur compromis.

Dans ce chapitre, l'application Géovélo est présentée en détail. Il s'agit d'un calculateur d'itinéraires adaptés aux cyclistes et disponible à l'adresse : http://www.geovelo.fr [1]. Les méthodes et améliorations proposées dans le chapitre 5 ont été implémentées sur le calculateur Géovélo. Plusieurs applications clientes ont été développées : un site web et deux applications mobiles.

La section 7.1 présente l'application Géovélo, l'architecture et les outils utilisés par ce projet. La section 7.2 explique le travail réalisé sur les données pour qu'elles puissent être exploitées par le calculateur. Dans la section 7.3, les détails sur le fonctionnement du calculateur et les méthodes utilisées sont présentées. Enfin, les sections 7.4 et 7.5 détaillent le fonctionnement du site web et des applications mobiles.

Ce travail a donné lieu à une communication [Sauvanet 10f] lors de la conférence annuelle State Of The Map 2010 (Girone) du projet OpenStreetMap.

1. Un prototype permettant de spécifier l'importance de chacun des objectifs est également disponible à l'adresse http://www.geovelo.fr/index.php?perso=true

7.1 Présentation de l'application

Du début jusqu'à la fin de ce travail de thèse, plusieurs versions de Géovélo se sont succédées. La figure 7.1 montre les différentes versions et évolutions de l'application. En effet, en plus des changements d'interfaces, les données, les zones couvertes et bien sûr les algorithmes ont évolué.

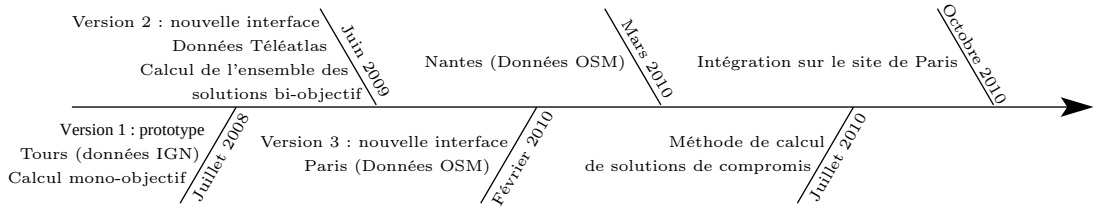


FIGURE 7.1 – Planning des différentes versions

Concernant les données, nous rappelons (cf. section 2.3) qu'une base de données routières de l'IGN a initialement été utilisée pour réaliser la version prototype du service. Suite à la décision de l'IGN d'abandonner cette base de données, nous nous sommes ensuite tournés vers la base de données Tele Atlas (mars 2009). Enfin, suite aux difficultés de mises à jour des données, nous avons finalement décidé d'utiliser les données du projet OpenStreetMap (août 2009).

L'objectif de cette thèse était de mettre en place un calculateur d'itinéraires adaptés au vélo sur l'agglomération de Tours. Cependant, nous avons également travaillé sur la ville de Paris et ses 30 communes limitrophes, ainsi que sur la communauté urbaine de Nantes. Notons que le service Géovélo a été intégré sur le site de la ville de Paris à l'adresse http://vgps.paris.fr (octobre 2010).

Plusieurs algorithmes de calcul d'itinéraires ont été testés sur le site web de Géovélo. Pour la version prototype, un simple calcul mono-objectif avait été implémenté. Celui-ci correspondait à une combinaison linéaire de la distance et de la sécurité. Par la suite, un algorithme de *label-setting* permettait de calculer l'ensemble des solutions non-dominées et la plupart des améliorations (cf. sections 4.3.2 et 4.4) présentées dans le chapitre 4 ont été intégrées sur le site web. Enfin, l'ajout de nouveaux objectifs, comme la linéarité de l'itinéraire ou l'effort a été accompagné par l'implémentation des méthodes de calcul de solutions de meilleur compromis, que nous avons vues au chapitre 5.

La figure 7.2 présente l'architecture générale du projet Géovélo. Celui-ci est composé principalement de trois éléments :

- la base de données,
- le calculateur,
- les applications clientes (site web et applications mobiles).

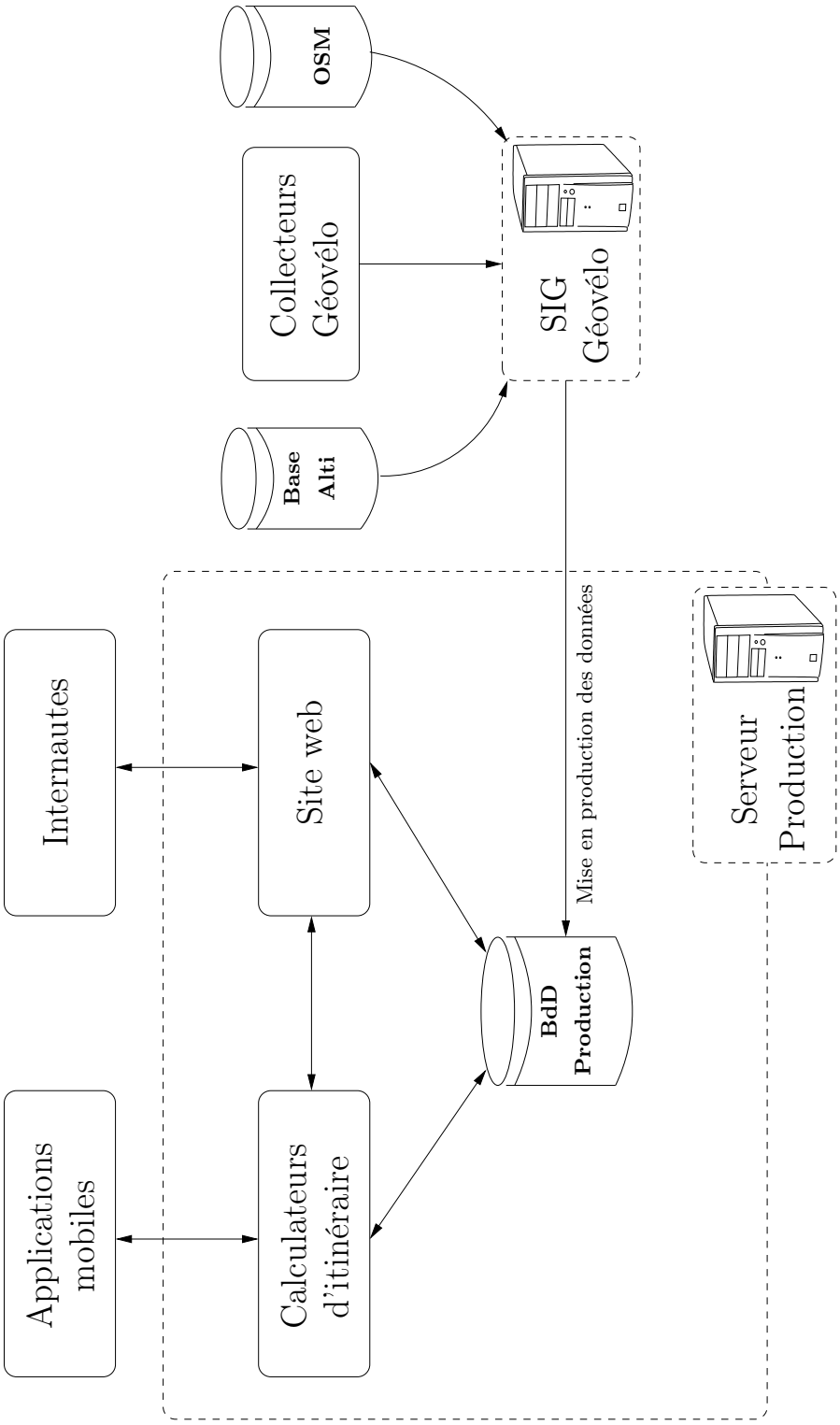


FIGURE 7.2 – Architecture générale de Géovélo

7.2 Travail préliminaire sur les données

Comme nous l'avons vu dans le chapitre 2, plusieurs sources de données existent en France. Nous avons choisi d'utiliser les données du projet libre OpenStreetMap [2] pour plusieurs raisons.

La raison la plus importante qui nous a conduit à utiliser OpenStreetMap est qu'il existe une grande liberté dans le choix des données pouvant être répertoriées. En effet, OpenStreetMap met à disposition des contributeurs des éléments géographiques de base : par exemple, *node* est un point géographique et *way* est un ensemble d'objets de type *node*. Ensuite, des mots-clés (appelés *tags*) peuvent être affectés à ces objets, mais rien n'est imposé. Un contributeur peut utiliser des mots-clés courants (*name* pour le nom d'une rue, *highway* pour le type de route, etc.) mais peut également attribuer de nouveaux mots-clés. Cependant, comme le projet est communautaire, si un contributeur utilise un nouveau mot-clé, il devra le documenter sur le site web dédié à la communauté et en discuter avec les autres contributeurs. S'il ne le fait pas, il y a alors très peu de chance pour que ce nouveau mot-clé soit utilisé et donc adopté par la communauté.

Un autre avantage d'OpenStreetMap est la réactivité des contributeurs. Sur d'autres bases de données comme celles de l'IGN ou de Tele Atlas, les mises à jour sont peu nombreuses et souvent payantes. Sur OpenStreetMap, les milliers de contributeurs français mettent à jour quotidiennement la base de données. Il est en effet plus facile de mettre à jour des données concernant une zone géographique que l'on connait bien : un contributeur peut mettre à jour très précisément les données routières près de chez lui, contrairement aux employés des gestionnaires des bases de données propriétaires qui doivent couvrir bien souvent tout un département. Ceci est particulièrement important pour les données concernant les aménagements cyclables, qui sont bien mieux renseignées sur OpenStreetMap que sur les autres bases de données pour lesquelles elles sont quasiment inexistantes. Par exemple, avant de commencer notre travail de terrain sur Paris, il y avait déjà de nombreuses pistes cyclables présentes sur OpenStreetMap.

Après avoir décidé de travailler avec la base de données OpenStreetMap, nous savions qu'un travail de terrain était nécessaire. Il fallait cependant décider des données qui seraient intégrées à OpenStreetMap et des données qui appartiendraient à l'entreprise « la Compagnie des Mobilités ». Ainsi, le collecteur de données a eu à travailler avec deux bases de données différentes : la base de données OpenStreetMap et la base de données interne à Géovélo.

Une journée type du collecteur de terrain commençait par une demi-journée à vélo pour parcourir un ensemble de rues. Une carte de la zone lui permet de noter l'ensemble des informations nécessaires. Une fois revenu dans son bureau, le collecteur utilise un logiciel de la communauté OpenStreetMap lui permettant de corriger et d'ajouter des données dans la base OpenStreetMap. Il peut s'agir d'ajouter une rue non-répertoriée, de préciser qu'une bande cyclable est présente sur une rue donnée, de signaler la présence d'un contresens cyclable, etc. Ensuite, le collecteur renseigne de nouvelles informations dans la base de données Géovélo. Cette base de données contient uniquement des données liées à la

2. http://www.openstreetmap.org

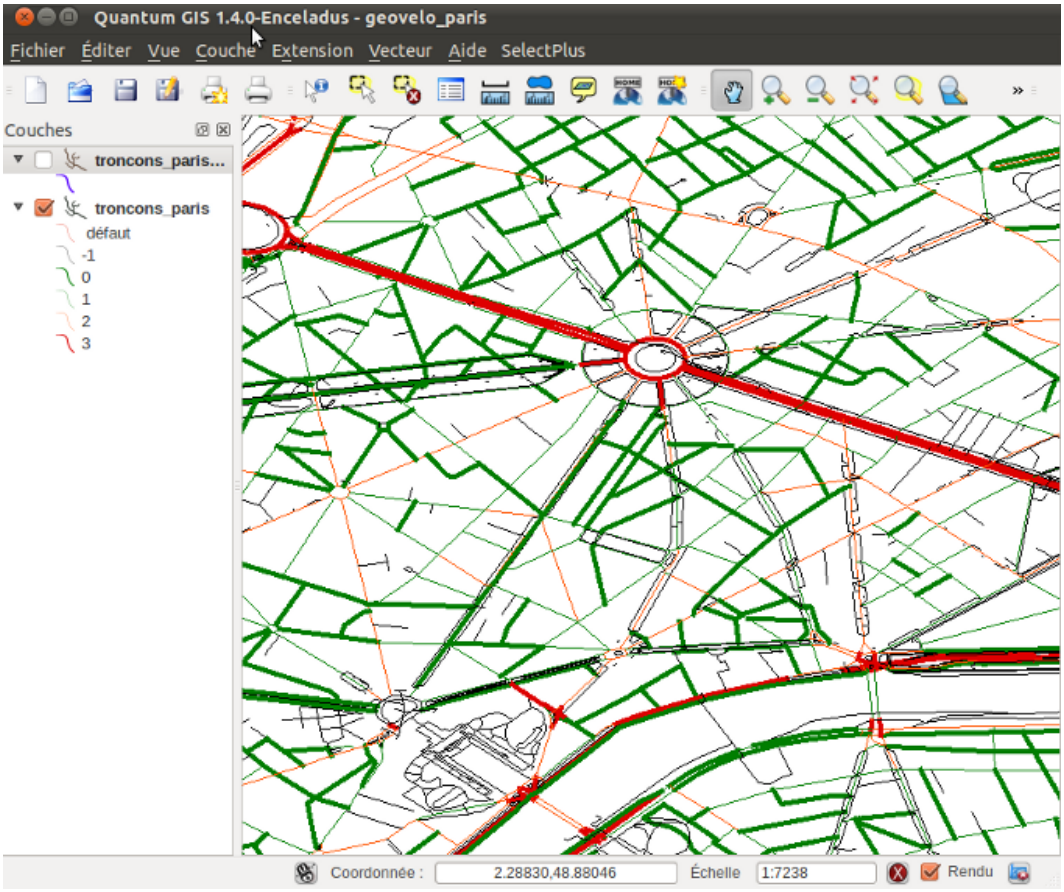


FIGURE 7.3 – Capture d'écran de l'application utilisée par le collecteur de terrain

notion de cyclabilité pour Géovélo : le collecteur indique, pour chaque voie, une note de sécurité ressentie (de 0 à 3) en fonction d'un certain nombre de critères (trafic automobile, vitesse réelle des voitures, largeur de la voie, etc.). Les critères d'attribution de cette note dépendent du type de voie (voie sans aménagement, piste cyclable, voie piétonne, etc.). Une application mobile est en cours de développement pour l'aider dans cette collecte.

Pour réaliser notre système d'information géographique (SIG), nous avons retenu une solution complètement libre composée du système de gestion de base de données PostgreSQL [3], couplé au module spatial PostGIS [4]. Ce dernier peut être vu comme un ensemble de types de données spatiales (points, lignes, polygones), de fonctions géographiques et de systèmes de projection. De même, le logiciel libre Quantum GIS [5] a été utilisé par le collecteur de terrain pour interagir graphiquement avec la base de données géographique. La figure 7.3 est une capture d'écran de cette application. Un style a été appliqué sur les tronçons de route pour visualiser les notes de cyclabilité : de 0 (conseillé) représenté en vert à 3 (fortement déconseillé) représenté en rouge. Cette note de cyclabilité est ensuite combinée au type d'aménagement pour obtenir le coefficient d'insécurité des voies.

Avant de pouvoir utiliser ces données pour faire du calcul d'itinéraires, de nombreuses étapes sont nécessaires. Voici un descriptif de toutes les étapes réalisées chaque nuit sur un des serveurs de Géovélo :

1. **Téléchargement des données OpenStreetMap** : la dernière version des données d'OpenStreetMap sur la France est téléchargée. Il s'agit d'un fichier *XML* qui peut être importé en base de données par l'utilitaire *osmosis* [6] fourni par la communauté.
2. **Découpage des données** : les données sur la France sont découpées en autant de zones couvertes par Géovélo (Paris, Nantes et Tours), de manière à réduire les traitements par la suite. Pour cela, un programme exécuté sur la base de données a été écrit. Ce programme permet également de régler un autre problème. Sur OpenStreetMap, une rue entière correspond à un seul objet, alors que pour faire du calcul d'itinéraires, il est nécessaire de découper une rue en tronçons de route correspondant à un morceau de rue entre deux intersections.
3. **Traitement des données** : les mots-clés d'OpenStreetMap sont convertis en attributs nécessaires au calcul d'itinéraires. La liberté offerte aux contributeurs a entrainé parfois plusieurs façons de modéliser le même élément géographique. Il a donc été nécessaire d'écrire un programme permettant de détecter des combinaisons courantes de mots-clés.
4. **Fusion des données** : la base de données d'OpenStreetMap est fusionnée avec les données internes de La Compagnie des Mobilités. De plus, l'altitude de chaque nœud est récupérée depuis une base de données altimétriques (utile pour générer le profil altimétrique des itinéraires et pour le critère d'effort).
5. **Génération du graphe** : la base de données est utilisée pour générer les nœuds et les arcs du graphe. Cette étape consiste à reprendre les nœuds et les tronçons de la

3. http://www.postgresql.org
4. http://www.postgis.org
5. http://www.qgis.org
6. http://wiki.openstreetmap.org/wiki/Osmosis

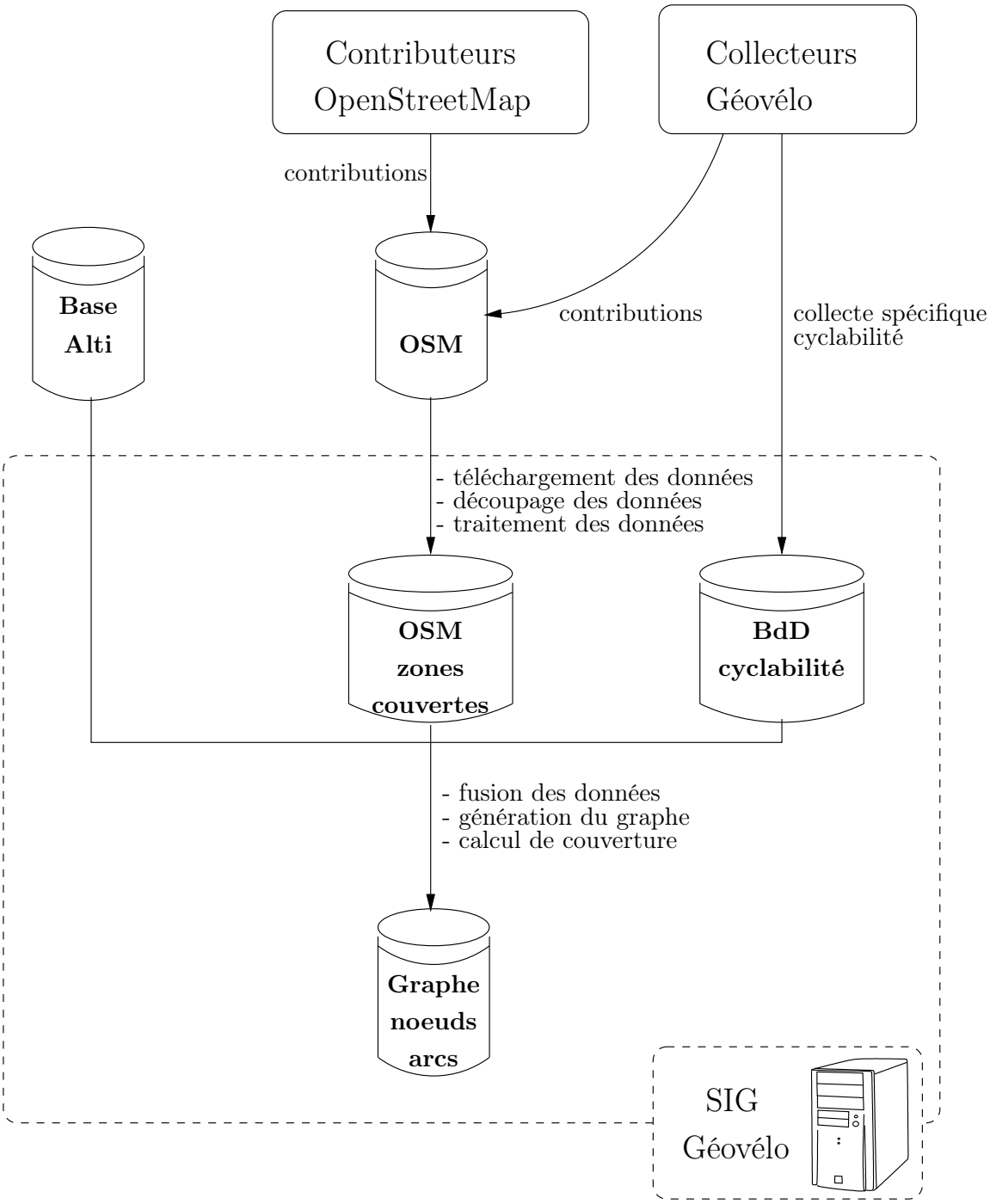


FIGURE 7.4 – Architecture et mise à jour des données

base OpenStreetMap. La création des arcs prend en compte les restrictions d'accès des voies et les sens de circulation.

6. **Calcul de couverture du graphe** : cette dernière étape permet de supprimer les nœuds et arcs qui ne sont pas liés au graphe principal. Il peut s'agir d'erreurs dans la base de données ou encore de voies qui ne sont pas encore reliées au réseau routier.

Pour donner un ordre d'idée, la base de données OpenStreetMap sur la France contient environ 1 360 000 objets de type voie, et environ 12 160 000 points géographiques.

En plus des critères de distance, de sécurité et de linéarité, un quatrième critère d'effort a été ajouté sur une version prototype de Géovélo (http://www.geovelo.fr/index.php?perso=true). La base de données altimétrique SRTM [7] a été utilisée. Un calcul d'interpolation a permis d'associer une altitude à chacun des nœuds du graphe, de manière à pouvoir ensuite calculer des pentes pour l'ensemble des tronçons de route. Des seuils ont ensuite été définis [8] et ont été associés à des coefficients d'effort. Ainsi, pour calculer le coût d'effort du parcours d'un arc, nous avons multiplié la longueur du tronçon de route par son coefficient d'effort.

7.3 Calcul d'itinéraires

Le calcul d'itinéraires est développé en langage *C++*. Pour faciliter la communication entre les applications clientes et le calculateur, il a été décidé de faire du calculateur un véritable web service accessible par requête *HTTP*. Pour permettre une communication entre le serveur *HTTP* et les processus du calculateur, la technique *FastCGI* a été utilisée. Le module *FastCGI* du serveur web *Apache2* permet de configurer le nombre de processus que l'on souhaite lancer et permet de s'affranchir de la gestion des files d'attente des requêtes car celles-ci sont gérées directement par *Apache2*.

Pour réaliser un calcul d'itinéraires sur un réseau routier de taille relativement importante, il est nécessaire d'utiliser des structures de données efficaces. En langage *C++*, il n'existe que très peu de librairies de gestion de graphe. Au début du projet, la librairie *LEDA* [9] a été utilisée pour réaliser les premiers prototypes et faire des tests. Par la suite, nous avons décidé de changer pour la librairie *Boost* [10] qui est de loin la plus utilisée, grâce à sa licence libre et sa communauté active. Nous avons cependant pu constater une perte de performance en termes de temps de calcul lors de la migration, qui peut s'expliquer par des structures de base moins efficaces avec *Boost* qu'avec *LEDA*. La figure 7.5 montre l'architecture général du calculateur.

Le calculateur peut calculer un ou plusieurs itinéraires en fonction de la demande de l'application cliente :

– soit l'application cliente précise le vecteur de poids, et l'itinéraire de meilleur compromis correspondant est renvoyé (cf. figure 7.7),

7. http://www2.jpl.nasa.gov/srtm/

8. Pour choisir les seuils, nous nous sommes inspirés d'un travail existant concernant une carte papier des pentes sur Paris (http://www.velopente.com).

9. http://www.algorithmic-solutions.info

10. http://www.boost.org

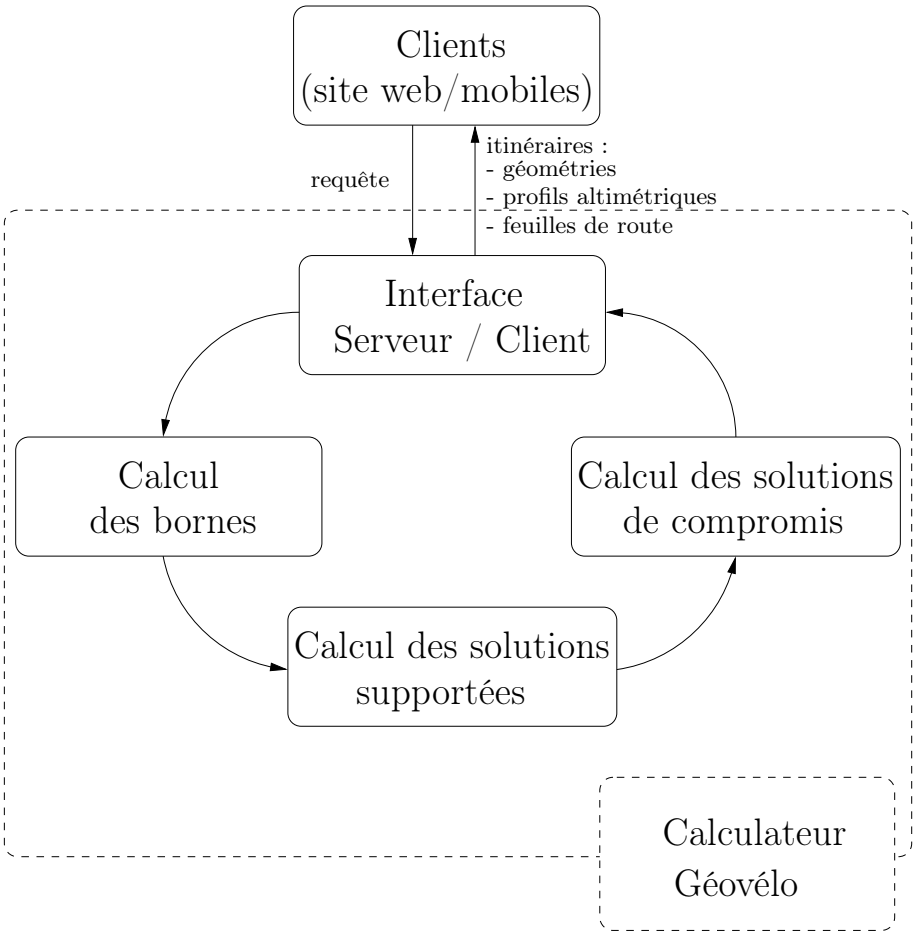


FIGURE 7.5 – Architecture du calculateur

– soit rien n'est précisé, et dans ce cas, trois itinéraires correspondant à des compromis différents (du plus court au plus sécurisé) sont calculés et renvoyés (cf. figure 7.6).

En ce qui concerne la méthode utilisée pour le calcul d'itinéraires, le calculateur utilise la méthode BCA* et ses améliorations présentées dans la section 5.3. Lorsqu'une application cliente envoie une requête au calculateur, ce dernier opère les étapes suivantes :

1. **Calcul des bornes** : pour chacun des objectifs, une recherche optimisant cet objectif est lancée, de manière à calculer les bornes inférieures et supérieures sur les nœuds. L'algorithme utilisé correspond à l'algorithme 5 modifié pour ne parcourir que les nœuds nécessaires et ne pas traiter l'ensemble du graphe.
2. **Calcul des solutions supportées** : pour chacun des compromis fixés par l'application cliente, une solution supportée est calculée avec une recherche mono-objectif optimisant une combinaison linéaire des objectifs correspondant au compromis fixé. Cette recherche a deux intérêts. Premièrement, elle permet de calculer des informations stockées sur chaque nœud pour améliorer la méthode BCA* (vecteur de coûts LC_v pour chaque nœud v concerné, cf. section 5.3). Deuxièmement, elle permet d'obtenir une solution par défaut pour le compromis fixé, qui peut être renvoyé au client si un problème survient avec la méthode BCA*.
3. **Calcul des solutions définitives** : pour chacun des compromis fixés par l'application cliente, la méthode BCA* est lancée en utilisant à la fois les bornes sur les objectifs et les informations sur la solution supportée.

Comme la recherche peut être trop longue dans certains cas (comme nous avons pu le remarquer pour une version prototype couvrant l'ensemble de la région Ile-de-France), il peut être intéressant de limiter la méthode BCA*. Actuellement, la recherche effectuée par BCA* peut être limitée de deux manières différentes. Premièrement, il est possible de limiter la recherche BCA* à un certain nombre d'étiquettes traitées. Si cette limite est atteinte, la méthode s'arrête et renvoie la meilleure solution trouvée qui correspond, dans le pire des cas, à la solution supportée calculée à l'étape précédente.

Une autre façon de limiter la recherche effectuée par la méthode BCA* est de réduire le nombre d'étiquettes de chaque nœud. Pour cela, il est possible d'utiliser la notion de dominance relâchée. Avec cette méthode, une solution peut dominer une autre solution, même si celle-ci n'était pas dominée au sens de Pareto. Par exemple, il est possible de modéliser le fait qu'un itinéraire de 1 200 mètres et avec 20 changements de direction domine un itinéraire de 1 199 mètres et 21 changements de direction. En effet, un cycliste ne jugera pas pertinent d'effectuer un changement de direction supplémentaire pour ne gagner qu'un seul mètre de distance en moins.

Ces deux éléments sont paramétrables sur le calculateur et permettent de s'adapter à la taille du graphe de la zone couverte ainsi qu'au nombre d'objectifs, pour s'assurer que la contrainte de temps de quelques secondes est respectée.

A la fin du calcul des itinéraires, le calculateur renvoie pour chacun des itinéraires :

– la géométrie de l'itinéraire, c'est-à-dire la longitude et la latitude de chacun des points géographiques,

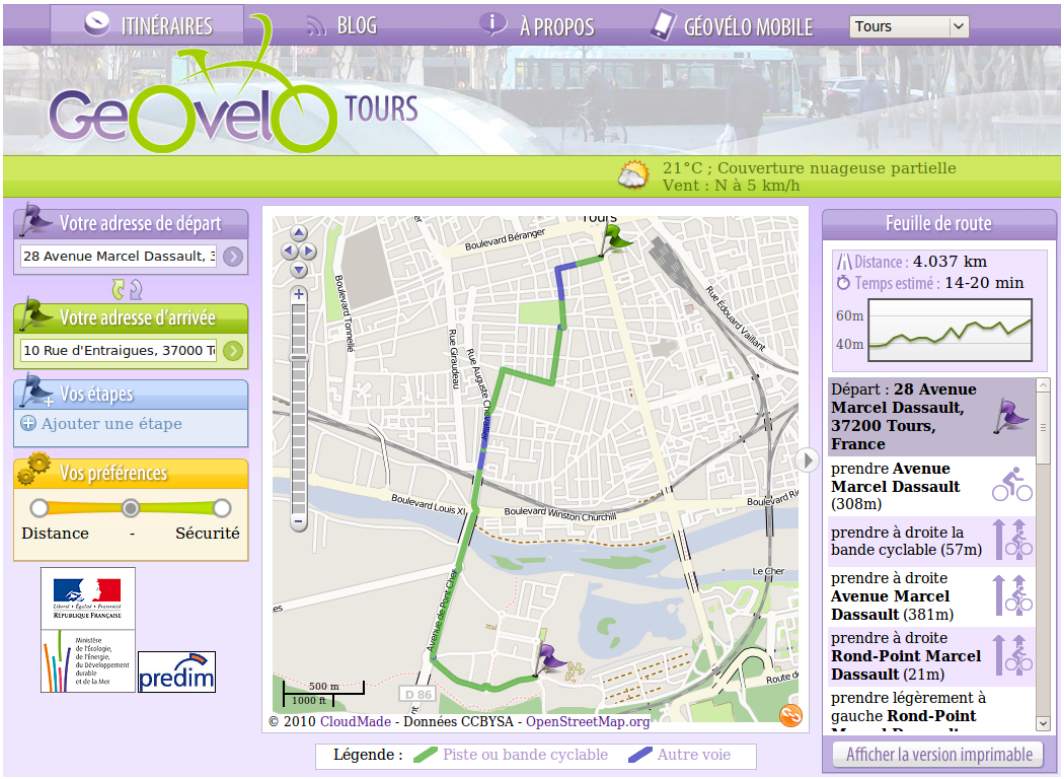


FIGURE 7.6 – Capture d'écran du site web Géovélo

- la feuille de l'itinéraire décrivant les changements de direction,
- le profil altimétrique de l'itinéraire.

Les informations, tels que le nom des rues, la position géographique et l'altitude de chacun des points, sont donc chargées en mémoire par le calculateur.

7.4 Site web

Le site web de Géovélo reprend les fonctionnalités de base de tout calculateur d'itinéraires routier. L'utilisateur peut ainsi choisir une adresse de départ et de destination et l'itinéraire est immédiatement calculé. L'interface a été réduite au maximum pour afficher la carte la plus grande possible. La sélection des adresses peut se faire de trois façons différentes :

- en les entrant textuellement,
- en déplaçant les curseurs de départ ou de destination,
- en les sélectionnant directement sur la carte avec un clic droit.

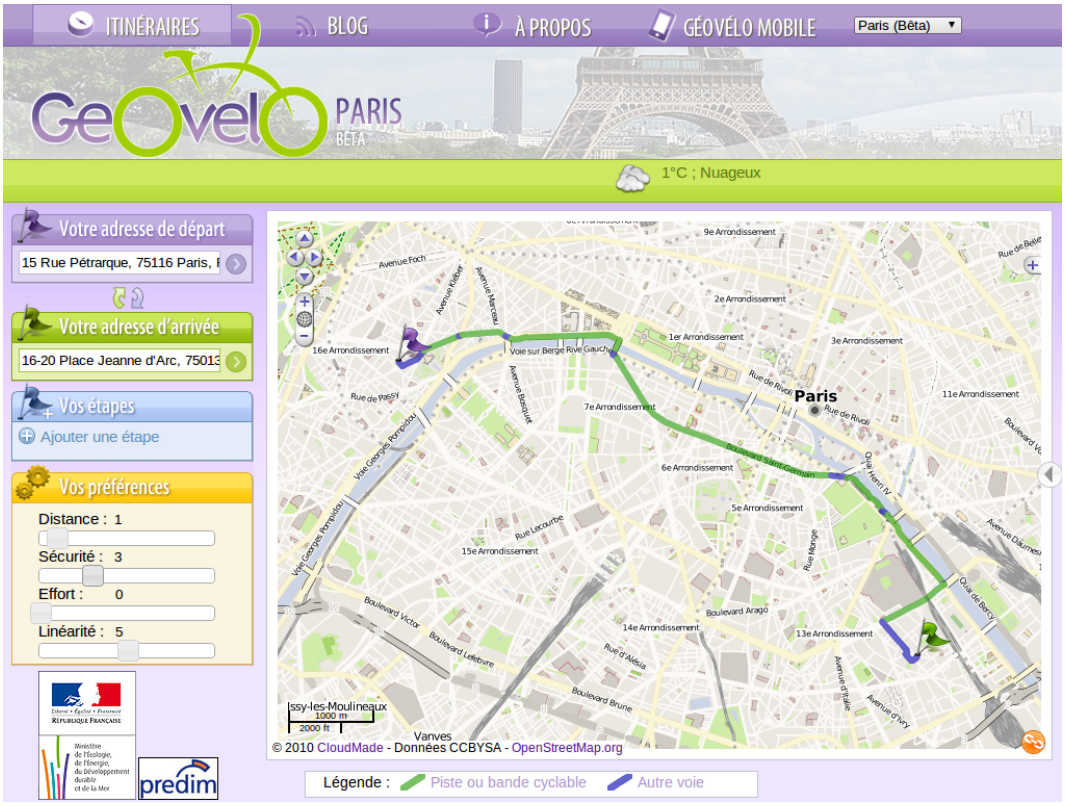


FIGURE 7.7 – Capture d'écran du site web Géovélo avec personnalisation des préférences

Une fois les adresses de départ et de destination entrées, le site web présente à l'utilisateur de un à trois itinéraires : du plus court au plus sécurisé où chaque itinéraire correspond à un compromis différent. Ces itinéraires sont représentés et accessibles depuis une barre de préférences. En passant le curseur sur la barre de préférences, il est possible d'afficher à l'écran deux itinéraires pour pouvoir les comparer. Un itinéraire est affiché sur la carte avec deux couleurs différentes : le vert pour les portions de routes comportant des aménagements cyclables et le bleu pour les autres portions de routes. Il est ainsi possible de visualiser rapidement si un itinéraire comporte de nombreux aménagements cyclables ou non. Notons qu'un rendu cartographique spécifique à l'usage du vélo est en cours de réalisation, en collaboration avec Dominique Andrieu, un collègue cartographe de l'université de Tours.

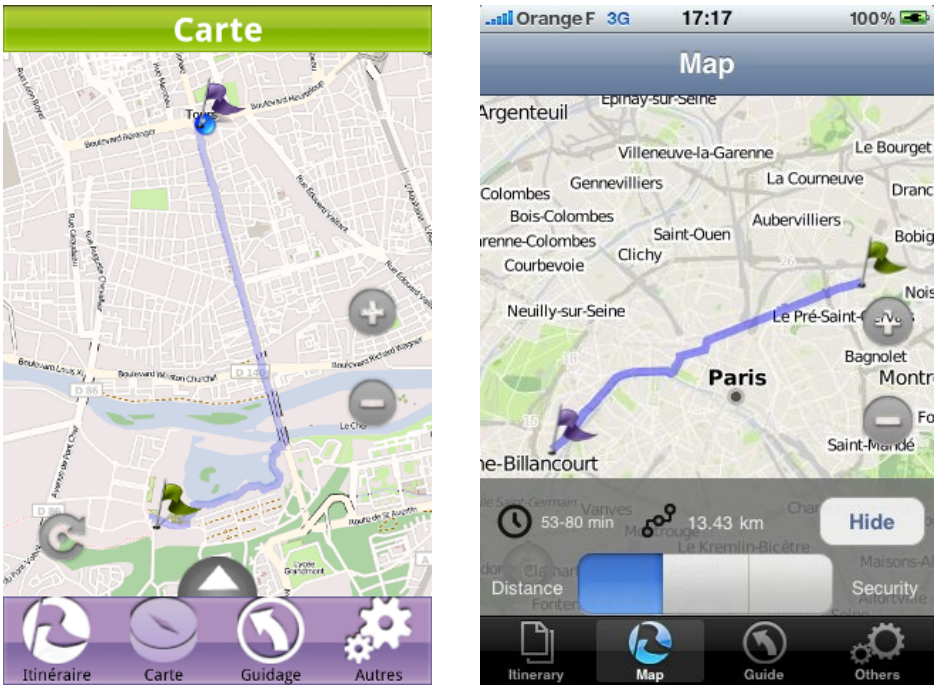
Sur la partie droite de l'écran, la longueur de l'itinéraire et une fourchette de temps de trajet sont affichées. Cette fourchette de temps permet de tenir compte de la variabilité des conditions de circulation, du type de cycliste, etc. Juste en dessous, le profil altimétrique permet à l'utilisateur de se rendre compte des efforts à fournir sur un itinéraire donné. Enfin, une feuille de route présente les changements de direction. Il est possible d'accéder à une version imprimable de la feuille de route.

Un seul site a été créé pour l'ensemble des zones couvertes par Géovélo. Tout en haut de l'écran, il est donc possible de choisir la zone sur laquelle on souhaite calculer un itinéraire.

Techniquement, le site web utilise le langage *php* et la base de données *PostgreSQL*. L'interface cartographique web est basée sur la librairie libre *OpenLayers*[11]. Cette dernière permet de gérer très simplement de nombreuses fonctionnalités : affichage d'une carte, affichage d'objets géographiques (points, polylignes et polygones), gestion des événements sur la carte, etc. Le style même de la carte a été créé à partir de l'outil *CloudMade*[12] qui permet de générer ses propres styles de carte à partir des données OpenStreetMap.

7.5 Applications mobiles

Dès le début du projet, des applications mobiles Géovélo étaient prévues car celles-ci sont indispensables pour une utilisation itinérante. Ces applications peuvent être vues comme des applications clientes, tout comme le site web car le calculateur n'est pas intégré sur ces applications. Ce choix permet de garder un calcul complexe des itinéraires et de soulager les applications mobiles. Par contre, cela oblige les applications mobiles à se connecter au serveur pour calculer un itinéraire.



(a) plate-forme Android (b) plate-forme iPhone

FIGURE 7.8 – Capture d'écran des applications Géovélo mobile

Cependant, comme les applications mobiles sont dédiées à une seule zone couverte, il est possible d'intégrer l'ensemble de la carte sur le téléphone. Ainsi, contrairement à la plupart de ce genre d'application, il n'est pas nécessaire de télécharger régulièrement des images correspondant à la carte où se trouve l'utilisateur.

11. http://www.openlayers.org
12. http://www.cloudmade.com

En termes de fonctionnalités, ces applications mobiles proposent exactement les mêmes que celles du site web. Quelques fonctionnalités ont tout de même été ajoutées. Pour commencer, le *GPS* des smartphones permet de géolocaliser précisément l'utilisateur sur la carte. Enfin, la boussole numérique permet d'ajouter la possibilité de faire pivoter la carte en fonction de l'orientation du smartphone.

Deux applications différentes ont été développées : une version pour les smartphones *iPhone* et une autre pour les smartphones utilisant le système d'exploitation *Android*. Le choix de ces deux plates-formes permet de couvrir la plupart des smartphones du marché actuel. Ce travail a été réalisé en collaboration avec Jiehao Juan, sous contrat avec l'université de Tours, dans le cadre d'une convention Staginno (financement FEDER).

La principale difficulté pour le développement de ces applications a été d'obtenir deux applications les plus proches possibles en termes de design et d'ergonomie, malgré les nombreuses différences entre les deux plates-formes.

Enfin, ces applications mobiles peuvent être téléchargées gratuitement, étant donné qu'elles ont été financées par l'appel à projet *Proxima Mobile*.

7.6 Conclusion du chapitre

Dans ce chapitre, la centrale de mobilité Géovélo a été présentée. Une partie importante du travail a été consacrée aux données géographiques. L'application Géovélo utilise les données libres du projet OpenStreetMap et une grande partie du travail de terrain a été redistribuée à ce projet. Une présentation [Sauvanet 10f] à la conférence internationale annuelle d'OpenStreetMap a permis d'expliquer à la communauté le travail réalisé.

En plus du calculateur et du site web, deux applications mobiles ont été réalisées dans le cadre de l'appel à projet Proxima Mobile financé par la Délégation aux usages de l'Internet. En 2009, Géovélo et la ville de Tours ont reçu le prix des Nouvelles Mobilités, suite au lancement de l'application sur l'agglomération de Tours. En 2010, le calculateur Géovélo a été intégré sur le site web de la ville de Paris.

Enfin, de nouvelles villes sont prévues, comme Lille, Rennes, La Rochelle, Lyon, etc. De même, une version touristique de Géovélo devrait bientôt voir le jour sur le département de l'Indre-Et-Loire et l'intégration des transports en commun est en réflexion pour rendre l'application multimodale.

Chapitre 8

Conclusion

Les travaux réalisés durant cette thèse portent sur le problème de plus court chemin multiobjectif pour la réalisation d'un calculateur d'itinéraires adaptés au vélo, c'est-à-dire prenant en compte plusieurs critères comme la distance, la sécurité et l'effort.

Dans le **chapitre 3**, le problème de plus court chemin multiobjectif a été introduit. Si en mono-objectif, le problème peut être résolu en temps raisonnable, même sur des graphes de très grande taille, en multiobjectif, les solutions de compromis peuvent être très nombreuses. Avec des objectifs de type **Min-Somme**, le problème est NP-Difficile. La problématique de cette thèse était alors de calculer des itinéraires adaptés aux cyclistes dans un contexte multiobjectif, tout en respectant une limite de temps de trois secondes pour pouvoir proposer ce service sous la forme d'un site web.

Deux approches de résolution ont été abordées durant ce travail de thèse :

- une approche *a posteriori* qui consiste à déterminer l'ensemble des solutions non-dominées,
- une approche *a priori* qui consiste à déterminer une unique solution de meilleur compromis correspondant aux préférences du décideur.

Dans le **chapitre 4**, l'approche *a posteriori* a été traitée en ne prenant en compte que deux objectifs, à savoir la distance et l'insécurité. Nous nous sommes concentrés sur les méthodes de *labelling*, et en particulier les méthodes de *label-setting*. Plusieurs règles d'élimination des étiquettes ont été reprises. Nous avons proposé une amélioration, basée sur un calcul des bornes inférieures et supérieures sur les coûts, pour accélérer la construction du front de Pareto et ainsi améliorer les performances des règles d'élimination. Les expérimentations montrent que cette méthode est très efficace sur les classes d'instances petites et moyennes.

Un prétraitement original, inspiré de la méthode ALT [Goldberg 05b] en mono-objectif, a été présenté pour approximer rapidement le front de Pareto. Les expérimentations réalisées montrent qu'il s'agit d'une alternative à la méthode à deux phases, qui est souvent utilisée dans les problèmes bi-objectif. Ce travail a donné lieu à plusieurs communications dans des conférences nationales [Sauvanet 09] et internationales [Sauvanet 10e, Sauvanet 10g]

ainsi qu'à une soumission en revue internationale [Sauvanet 10a].

Dans le **chapitre 5**, l'approche *a priori* a été considérée dans un contexte multiobjectif, où les préférences du décideur sont connues à l'avance. Nous nous sommes intéressés à des améliorations de la méthode BCA* [Futtersack 00] qui permet de se concentrer sur la détermination d'une solution de meilleur compromis, en traitant en premier les étiquettes les plus prometteuses du point de vue des préférences du décideur.

Des améliorations basées sur un calcul préliminaire de vecteurs de coûts ont été proposées. Les expérimentations prouvent la performance de ces améliorations, même sur les classes d'instances les plus complexes, ainsi que leur efficacité sur des instances avec plus de deux objectifs. Ce travail a été présenté dans une conférence internationale [Sauvanet 10b] et a donné lieu à une soumission en revue internationale [Sauvanet 10c].

Dans le **chapitre 6**, nous avons démontré que l'utilisation du graphe adjoint permettait de prendre en compte des critères et contraintes essentiels pour réaliser du calcul d'itinéraires adaptés au vélo. Il est en effet possible de prendre en compte des coûts sur les nœuds, des coûts sur les arcs, des coûts sur les enchaînements d'arcs et des enchaînements d'arcs interdits, tout en utilisant des méthodes classiques de calcul de plus court chemin. Cependant, l'utilisation du graphe adjoint peut parfois poser problème car sa taille est souvent plus grande que celle du graphe initial pour modéliser un réseau routier.

De nouveaux objectifs ont été introduits, comme le temps de parcours qui prend en compte des coûts sur les arcs et sur les nœuds ainsi que la linéarité de l'itinéraire qui tient compte de coûts sur des enchaînements d'arcs. Des expérimentations avec trois objectifs sur le graphe, correspondant au réseau routier de la ville de Paris montrent que cette modélisation est pertinente et qu'elle respecte la contrainte de temps de trois secondes que nous nous sommes fixés. Ce travail a donné lieu à une communication en conférence nationale [Sauvanet 11].

Dans le **chapitre 7**, nous avons présenté le service Géovélo qui a été développé durant cette thèse. Il s'agit d'une base de données cyclable, d'un calculateur d'itinéraires multiobjectif adaptés aux cyclistes, et de deux applications clientes, à savoir un site web et des applications mobiles.

Une partie importante de ce travail repose sur la base de données et l'utilisation du projet libre OpenStreetMap qui est une base de données routières communautaire. Une présentation à la conférence annuel d'OpenStreetMap a expliqué à la communauté comment les données avaient été utilisées, corrigées et complétées pour pouvoir réaliser le service Géovélo. Ce travail a donné lieu à une communication [Sauvanet 10f] lors de la conférence annuelle State Of The Map 2010 (Girone) du projet OpenStreetMap.

La méthode de calcul d'une solution de meilleur compromis et ses améliorations ont été intégrées au calculateur d'itinéraires Géovélo. Ce dernier prend actuellement en compte la distance, l'insécurité, l'effort et la linéarité de l'itinéraire.

Le projet Géovélo mobile a été retenu dans le cadre de l'appel à projet Proxima Mobile financé par la Délégation aux usages de l'Internet. Deux applications mobiles ont ainsi été développées sur les plates-formes de type iPhone et Android. Suite au lancement du service sur l'agglomération de Tours en 2009, Géovélo et la ville de Tours ont reçu le prix

des Nouvelles Mobilités. Fin 2010, le service Géovélo a été intégré sur le site de la ville de Paris (http://vgps.paris.fr).

Plusieurs perspectives de recherche sont envisagées à court et moyen terme :

- **Modification de l'ordre de traitement des étiquettes**. Dans le chapitre 4, nous avons travaillé uniquement avec des méthodes de *label-setting*, en utilisant un ordre d'exploration lexicographique des étiquettes. Il pourrait être intéressant de tester les améliorations proposées sur des méthodes de *label-correcting* [Guerriero 01]. En particulier, la méthode LSAP permet d'approximer la front de Pareto pendant la recherche, mais pas de façon uniforme, étant donné que l'ordre d'exploration des étiquettes est lexicographique.
- **Échantillonnage du front de Pareto**. Dans le chapitre 4, nous avons constaté que le nombre de solutions non-dominées pouvait être très important. Une problématique intéressante est alors d'échantillonner le front de Pareto pour proposer à l'utilisateur un sous-ensemble de solutions représentatives.
- **Utiliser d'autres fonctions d'agrégation**. Dans le chapitre 5, nous avons utilisé la norme de Tchebycheff comme fonction d'agrégation des objectifs. Cependant, il existe d'autres fonctions permettant de définir plus précisément les préférences et relations entre les objectifs, ce qui est d'autant plus important dès lors que l'on travaille avec plus de trois objectifs. Il serait intéressant d'étudier, par exemple, comment les améliorations proposées peuvent être adaptées à l'intégrale de Choquet [Choquet 53, Fouchal 11, Galand 08].
- **Prise en compte du caractère touristique**. L'attrait touristique peut prendre la forme de deux problèmes. La première possibilité est de calculer des itinéraires qui vont maximiser, par exemple, les tronçons de route proches d'un cours d'eau, d'une forêt, etc. La deuxième possibilité est de calculer des itinéraires passant par un ensemble de points d'intérêt, tels que des monuments, jardins, musées, etc. Ce deuxième problème a été en partie étudié durant cette thèse [Sauvanet 10d]. Un projet avec le département de l'Indre-et-Loire est actuellement en cours pour réaliser un calculateur d'itinéraires touristiques adaptés au vélo sur l'ensemble du département.
- **Rendre le service multimodal**. L'utilisation du vélo n'est bien sûr pas adaptée pour les grandes distances. Par contre, l'utilisation du vélo combiné avec les transports en commun possède un potentiel très prometteur. Il est alors intéressant d'étudier la possibilité de calculer des itinéraires multiobjectif et multimodaux. Nous avons initié plusieurs travaux préliminaires sur le sujet [Dewaele 11]. Dans [Gueye 10], l'auteur s'intéresse à ce problème avec deux objectifs : le temps et le nombre de changements de modes. Dans [Gräbener 10], l'auteur considère le même problème mais avec des objectifs plus conflictuels générant plus de solutions non-dominées. Enfin, il pourrait être intéressant d'étudier les approches de type *a priori*, qui ont, à notre connaissance, rarement été considérées pour ce problème.

À noter que ces perspectives seront traitées par un nouveau doctorant qui vient de

commencer sa thèse dans l'entreprise.

Bibliographie

[ade 06] Piétons et cyclistes : de bons clients pour vos commerces. ADEME, 2006.

[Añez 96] J. Añez, T. De La Barra & B. Pérez. *Dual graph representation of transport networks.* Transportation Research Part B : Methodological, vol. 30, no. 3, pages 209–216, 1996.

[Bast 07] H. Bast, S. Funke, D. Matijevic, P. Sanders & D. Schultes. *In Transit to Constant Time Shortest-Path Queries in Road Networks.* In ALENEX. SIAM, 2007.

[Bauer 10] R. Bauer & D. Delling. *SHARC : Fast and robust unidirectional routing.* J. Exp. Algorithmics, vol. 14, pages 4 :2.4–4 :2.29, January 2010.

[Bellman 57] R. Bellman. Dynamic programming. Princeton University Press, 1957.

[Bellman 58] R. Bellman. *On a Routing Problem.* Quarterly of Applied Mathematics, vol. 16, no. 1, pages 87–90, 1958.

[Benayoun 71] R. Benayoun, J. de Montgolfier, J. Tergny & O.I. Larichev. *Linear Programming with Multiple Objective Functions : STEP Method (STEM).* Mathematical Programming, vol. 1, no. 3, pages 366–375, 1971.

[Brumbaugh-Smith 89] J. Brumbaugh-Smith & D. Shier. *An empirical investigation of some bicriterion shortest path algorithms.* European Journal of Operational Research, vol. 43, no. 2, pages 216–224, November 1989.

[Caldwell 61] T. Caldwell. *On finding minimum routes in a network with turn penalties.* Commun. ACM, vol. 4, pages 107–108, February 1961.

[Carré 99] J-R. Carré. Recherche expérimentale sur les stratégies des cyclistes dans la circulation. 1999.

[Carré 03] J-R. Carré. Ecomobilité, les déplacements non motorisés : marche, vélo, roller ; élements clé pour une alternative en matière de mobilité urbaine. 2003.

[Choquet 53] G. Choquet. *Theory of capacities.* Annales de l'Institut Fourier, vol. 5, pages 131–295, 1953.

[Clímaco 82] J.N. Clímaco & E. Martins. *A bicriterion shortest path algorithm.* European Journal of Operational Research, vol. 11, pages 399–404, 1982.

[Climaco 10] J. Climaco & M. Pascoal. *Multicriteria Path and Tree Problems-Discussion on Exact Algorithms.* In Proceedings of ALIO-INFORMS International Meeting, Buenos Aires, 2010.

[Dep 07] Réussir sa politique vélo. outils pratiques pour une communication efficace. Association des Départements cyclables, 2007.

[Dewaele 11] E. Dewaele, G. Sauvanet & E. Néron. *Meilleur compromis multi-objectif et multi-modal.* In Actes du 12^{me} congrés de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF11), Saint-Étienne, France, 2011.

[Dijkstra 59] E.W. Dijkstra. *A note on two problems in connection with graphs.* Numerische Mathematik, no. 1, pages 269–271, 1959.

[Ehrgott 00] M. Ehrgott & X. Gandibleux. *A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization.* OR Spektrum, vol. 22, pages 425–460, 2000.

[Ehrgott 02] M. Ehrgott & X. Gandibleux. *Multiobjective Combinatorial Optimization.* In M. Ehrgott & X. Gandibleux, editeurs, Multiple Criteria Optimization - State of the Art Annotated Bibliographic Surveys, volume 52, pages 369–444. Kluwer Academic Publishers, Boston, MA, 2002.

[Ford 56] L.R. Ford. *Network Flow Theory.* Paper P-923, The RAND Corperation, Santa Moncia, California, August 1956.

[Fouchal 11] H. Fouchal, X. Gandibleux & F. Lehuédé. *Preferred solutions computed with a label setting algorithm based on Choquet integral for Multi-Objective Shortest Paths.* In Proceedings of IEEE Symposium Series on Computational Intelligence 2011, Paris, April 2011. accepted.

[Fredman 87] M.L. Fredman & R.E. Tarjan. *Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms.* Journal of the ACM, vol. 34, pages 596–615, 1987.

[fub 07] Vélo et sécurité routière. FUBicy, 2007.

[Futtersack 00] M. Futtersack & P. Perny. *BCA*, une généralisation d'A* pour la recherche de solutions de compromis dans des problèmes de recherche multiobjectifs.* In Actes de la conférence RFIA'2000, vol. 3, pages 377–386, 2000.

[Galand 08] L. Galand. *Méthodes exactes pour l'optimisation multicritère dans les graphes : recherche de solutions de compromis.* Thèse de doctorat, Université Paris 6, 2008.

[Geisberger 08] R. Geisberger, P. Sanders, D. Schultes & D. Delling. *Contraction Hierarchies : Faster and Simpler Hierarchical Routing in Road Net-*

works. In Catherine C. McGeoch, editeur, WEA, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2008.

[Geoffrion 68] A.M. Geoffrion. *Proper efficiency and the theory of vector minimization*. Journal of Mathematical Analysis and Applications, no. 22, pages 618–630, 1968.

[Gerber 09] C. Gerber, H. Baptiste & G. Sauvanet. Les déplacements en vélo dans la ville : la question de la sécurité. CITERES, 2009.

[Goldberg 05a] A. Goldberg & C. Harrelson. *Computing the shortest path : A* meets graph theory*. Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005), SIAM (2005), 2005.

[Goldberg 05b] A. Goldberg, H. Kaplan & R. Werneck. *Reach for A* : Efficient point-to-point shortest path algorithms*. Demetrescu, C., Sedgewick, R., Tamassia, R., eds. : Proceedings of the 7th Workshop on Algorithm Engineering and Experimentation (ALENEX 05), SIAM (2005), 2005.

[Gräbener 10] T. Gräbener. *Calcul d'itinéraire multimodal et multiobjectif en milieu urbain*. Thèse de doctorat, Université de Toulouse, 2010.

[Guerriero 01] F. Guerriero & R. Musmanno. *Label correcting methods to solve multicriteria shortest path problems*. Journal of Optimization Theory and Applications, vol. 111, no. 3, pages 589–613, 2001.

[Gueye 10] F. Gueye. *Algorithmes de recherche d'itinéraires en transport multimodal*. Thèse de doctorat, INSA Toulouse, 2010.

[Hansen 80] P. Hansen. *Bicriterion path problems*. In G. Fandel & T. Gal, editeurs, Multiple Criteria Decision Making : Theory and Applications, LNEMS 177, pages 109–127. Springer-Verlag, Berlin, 1980.

[Harary 69] F. Harary. *Graph Theory*. Addison-Wesley, 1969.

[Hart 68] E. Hart, N. Nilsson & B. Raphael. *A formal basis for the heuristic determination of minimum cost paths*. IEEE Transactions on Systems, Science and Cybernetics SSC-4, no. 2, pages 100–107, 1968.

[Hilger 09] M. Hilger, E. Köhler & R.H. Möhringand H. Schilling. *Fast point-to-point shortest path computations with arc-flags*. The Shortest Path Problem : Ninth DIMACS Implementation Challenge, pages 73–92, 2009.

[hol 09] Le vélo au pays-bas. Ministère des transports hollandais, 2009.

[Ins 98] Guidelines for cycle audit and cycle reviews. Institution of Highways and Transportation, 1998.

[Laporte 10] S. Laporte, H. Baptiste & G. Sauvanet. Caractérisation de l'effort pour la conception d'itinéraires cyclables : l'émergence d'un modèle. CITERES, 2010.

[Machuca 10] E. Machuca, L. Mandow, J-L Pérez de-la Cruz & A. Ruiz-Sepúlveda. *An Empirical Comparison of Some Multiobjective Graph Search Algorithms.* In R. Dillmann, J. Beyerer, U.D. Hanebeck & T. Schultz, editeurs, KI, volume 6359 of *Lecture Notes in Computer Science*, pages 238–245. Springer, 2010.

[Madre 97] JL. Madre. *D'après enquêtes ménages.* 1997.

[Madre 08] JL. Madre. *Enquêtes ménages déplacements.* 2008.

[Malgras-Serra 06] A. Malgras-Serra. Karl drais - la nouvelle biographie. Goethe-Institut Mannheim-Heidelberg, 2006.

[Mandow 05] L. Mandow & J-L. Pérez de-la Cruz. *A New Approach to Multiobjective A* Search.* In Leslie Pack Kaelbling & Alessandro Saffiotti, editeurs, IJCAI, pages 218–223. Professional Book Center, 2005.

[Martins 84] E.Q.V. Martins. *On a multicriteria shortest path problem.* European Journal of Operational Research, vol. 16, pages 236–245, 1984.

[Noel 03] N. Noel. *Forme urbaine, aménagements routiers et usages de la bicyclette.* Thèse de doctorat, Université de Laval, 2003.

[Pascoal 06] Marta M. B. Pascoal, M. Eugénia V. Captivo & João C. N. Clímaco. *A comprehensive survey on the quickest path problem.* Annals OR, vol. 147, no. 1, pages 5–21, 2006.

[Pugh 90] W. Pugh. *Skip Lists : A Probabilistic Alternative to Balanced Trees.* Communications of the ACM, vol. 33, no. 6, pages 668–676, 1990.

[Raith 09] A. Raith & M. Ehrgott. *A comparison of solution strategies for biobjective shortest path problems.* Computers & OR, vol. 36, no. 4, pages 1299–1331, 2009.

[Richter 08] K-F. Richter & M. Duckham. *Simplest Instructions : Finding Easy-to-Describe Routes for Navigation.* In Proceedings of the 5th international conference on Geographic Information Science, GIScience '08, pages 274–289. Springer-Verlag, 2008.

[Rosinger 91] E. E. Rosinger. *Beyond preference information based multiple criteria decision making.* European Journal of Operational Research, vol. 53, no. 2, pages 217–227, July 1991.

[Sanders 06] P. Sanders & D. Schultes. *Engineering Highway Hierarchies.* In ESA, pages 804–816, 2006.

[Sauvanet 09] G. Sauvanet & E. Néron. *Calcul de plus court chemin bicritère.* In Actes du 10^{me} congrés de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF09), Nancy, France, 2009.

[Sauvanet 10a] G. Sauvanet & E. Néron. *Improvements in labeling methods for Biobjective Shortest Path Problem : application to the distance/safety trade-off.* soumis à European Journal of Operational Research, 2010.

[Sauvanet 10b] G. Sauvanet & E. Néron. *Search for the best compromise solution on Multiobjective shortest path problem.* Electronic Notes in Discrete Mathematics, vol. 36, pages 615–622, 2010.

[Sauvanet 10c] G. Sauvanet & E. Néron. *Search for the best compromise solution on multiobjective shortest path problem.* soumis à Journal of Mathematical Modelling and Algorithms, 2010.

[Sauvanet 10d] G. Sauvanet & E. Néron. *Un problème de tournée touristique bicritère.* In Actes du 11^{me} congrés de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF10), Toulouse, France, 2010.

[Sauvanet 10e] G. Sauvanet, E. Néron & H. Baptiste. *Computation of best-compromise route for cycling.* In Proceedings of the 24^{th} European Conference on Operational Research (EURO10), Lisbon, Portugal, 2010.

[Sauvanet 10f] G. Sauvanet, E. Néron & H. Baptiste. *Geovelo, a route planner for bicycle.* Girona, Spain, 2010.

[Sauvanet 10g] G. Sauvanet, E. Néron & H. Baptiste. *A pre-processing methode for the bi-objective shortest path problem.* In Proceedings of the 9^{th} International Conference on Multiple Objective Programming and Goal Programming (MOPGP10), Sousse, Tunisie, 2010.

[Sauvanet 11] G. Sauvanet & E. Néron. *Calcul de chemin de meilleur compromis dans un graphe multiobjectif dual linéaire.* In Actes du 12^{me} congrés de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF11), Saint-Étienne, France, 2011.

[Schultes 07] D. Schultes & P. Sanders. *Dynamic highway-node routing.* In Proceedings of the 6th international conference on Experimental algorithms, WEA'07, pages 66–79, Berlin, Heidelberg, 2007. Springer-Verlag.

[Serafini 87] P. Serafini. *Some considerations about computational complexity for multiobjective combinatorial problems.* In Lecture Notes in Economics and Mathematical Systems, pages 222–232. Springer-Verlag, 1987.

[Steuer 83] R.E. Steuer & E. Choo. *An Interactive Weighted Tchebycheff Procedure for Multiple Objective Programming.* Mathematical Programming, vol. 26, no. 1, pages 326–344, 1983.

[Stewart 91] B.S. Stewart & C.C. White. *Multiobjective A*.* Journal of the ACM, no. 38(4), pages 775–814, 1991.

[Tarapata 07] Z. Tarapata. *Selected multicriteria shortest path problems : An analysis of complexity, models and adaptation of standard algorithms.* International Journal Of Applied Mathematics And Computer Science, vol. 17, pages 269–287, 2007.

[tou 09] Tours info. Ville de Tours, 2009.

[Tung 92] C. Tung Tung & K.L. Chew. *A multicriteria pareto-optimal path algorithm.* European Journal of Operational Research, vol. 62, pages 203–209, 1992.

[Ulungu 95] E. L. Ulungu & J. Teghem. *The two phases method : An efficient procedure to solve bi-objective combinatorial optimization problems.* Foundations of Computing and Decision Sciences, vol. 20, no. 2, pages 149–165, 1995.

[Uster 08] G. Uster. Service de mobilité et d'information : innovation et recherche. La Documentation française, 2008.

[Vanderpooten 89a] D. Vanderpooten. *The interactive approach in MCDA : a technical framework and some basic conceptions.* Mathematical and Computer Modelling, vol. 12, pages 1213–1220, 1989.

[Vanderpooten 89b] D. Vanderpooten. *L'approche interactive dans l'aide multicritère à la décision : Aspects conceptuels, méthodologiques et informatiques.* Thèse de doctorat, Université Paris-Dauphine, 1989.

[Vincke 74] P. Vincke. *Problèmes multicritères.* Cahiers du CERO, vol. 16, 1974.

[Wachter 05] S. Wachter, J. Theys, Y. Crozet & J-P. Orfeuil. La mobilité urbaine en débat : cinq scénarios pour le futur ? CERTU, 2005.

[Whitney 32] H. Whitney. *Congruent graphs and the connectivity of graphs.* American Journal of Mathematics, vol. 54, pages 150–168, 1932.

[Wierzbicki 86] A.P. Wierzbicki. *On the completeness and constructiveness of parametric characterizations to vector optimization problems.* OR Spektrum, vol. 8, pages 73–87, 1986.

[Winter 01] S. Winter. *Weighting the path continuation in route planning.* In GIS '01 : Proceedings of the 9th ACM international symposium on Advances in geographic information systems, pages 173–176, New York, NY, USA, 2001. ACM.

[Winter 02] S. Winter. *Modeling Costs of Turns in Route Planning.* Geoinformatica, vol. 6, pages 345–361, December 2002.

[Yager 88] R.R. Yager. *On ordered weighted averaging aggregation operators in multicriteria decisionmaking.* IEEE Transactions on Systems, Man and Cybernetics, vol. 18, no. 1, pages 183–190, 1988.

[Yu 98] G. Yu & J. Yang. *On the robust shortest path problem.* Computers and Operations Research, vol. 25, pages 457–468, 1998.

[Zeleny 74] M. Zeleny. *A Concept of Compromise Solutions and the Method of the Displaced Ideal.* Computers and Operations Research, vol. 1, no. 3, pages 479–496, 1974.

[Zeleny 82] M. Zeleny. Multiple criteria decision making. McGraw-Hill, New York, 1982.

Gaël SAUVANET

Recherche de chemins multiobjectifs pour la conception et la réalisation d'une centrale de mobilité destinée aux cyclistes

Résumé :

Les travaux présentés dans cette thèse visent à proposer des méthodes de calcul d'itinéraires adaptés aux cyclistes à l'échelle d'une agglomération. Plusieurs critères sont considérés, comme la distance, la sécurité et l'effort. La difficulté est de calculer des chemins de compromis sous une contrainte de temps de quelques secondes pour pouvoir intégrer ce calculateur à un site web.
Deux approches ont été abordées pour résoudre ce problème. L'approche *a posteriori* dans laquelle l'ensemble des solutions de compromis est calculé et l'approche *a priori* dans laquelle les préférences de l'utilisateur sont prises en compte et permettent d'orienter la recherche pour privilégier les chemins les plus prometteurs. Enfin, nous proposons de modéliser le réseau routier sous la forme d'un graphe adjoint pour pouvoir prendre en compte de nouveaux critères nécessitant, par exemple, des coûts sur les enchaînements d'arcs.
L'ensemble de ce travail a permis de développer le service Géovélo qui est un calculateur d'itinéraires multiobjectif adaptés au vélo. Le service est disponible sous la forme d'un site web et d'applications mobiles.
Mots-clés : problème de plus court chemin, graphe, optimisation multiobjectif.

Abstract :

The work presented in this thesis aims at proposing methods for computing bicycle paths across a metropolitan. Several criteria such as distance, safety and effort must be considered in the path computation. The difficulty is to compute paths under a time constraint of a few seconds, in order to integrate the computation in the respond-time of a web page.
Two approaches were discussed to solve this problem. The first one is an *a posteriori* approach where all compromise solutions are computed and the second approach is an *a priori* method that takes user preferences into account to guide the search by the selection of the most promising sub-paths first. Finally, we propose to model the road network as a line graph to take into account new criteria, requiring costs on arc sequences for example.
All this work was necessary to develop the service Géovélo, which is a multiobjective route planner adapted to bicycle. The service is available on a website and as mobile applications.
Key words : shortest path problem, graph, multiobjective optimisation.