UNIVERSITÉ DU QUÉBEC À MONTRÉAL

PLANIFICATION D'ITINÉRAIRES POUR VÉHICULE ÉLECTRIQUE AVEC DISPONIBILITÉ INCERTAINE DES BORNES DE RECHARGE

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
JAËL CHAMPAGNE GAREAU

DÉCEMBRE 2019

REMERCIEMENTS

Je remercie d'abord tout particulièrement mes directeurs de recherche, les professeurs Éric Beaudry et Vladimir Makarenkov, pour l'encadrement et les judicieux conseils qu'ils m'ont donnés. Leurs encouragements à sans cesse me dépasser m'ont permis de rester motivé même pendant les périodes plus difficiles.

De plus, je tiens à remercier le Fonds de recherche du Québec — Nature et technologies (FRQNT), la fondation de l'UQAM, ainsi que mes directeurs de recherche pour le soutien financier qu'ils m'ont accordé. Sans celui-ci, je n'aurais indéniablement pas pu m'investir autant dans ce projet.

Les bons outils étant nécessaires à une bonne productivité, je remercie Bjarne Stroustrup, l'initiateur du langage C++, Donald Knuth et Leslie Lamport, les créateurs de TEX et LATEX, Linus Torvalds, le créateur de Linux et de Git, ainsi que Bill Joy et Bram Moolenaar, créateurs respectifs de Vi et de Vim.

Je remercie tous mes collègues étudiants de laboratoire (Éric, Gabrielle, Jean, Mathieu, Jonathan, Marc-André, Charline, Guillaume, Philippe, Arnaud, Erwan, Alexandre et Lise) qui sont devenus non seulement de bons collègues, mais également de bons amis. Je remercie particulièrement Lise Jolicœur qui a entre autres travaillé sur un site web permettant une utilisation simple de mon planificateur.

Sur un plan plus personnel, je souhaite finalement remercier mes parents, qui m'ont encouragé dans mes études et m'ont soutenu financièrement, mon frère, avec qui je parle régulièrement d'informatique, ainsi que mes amis. Un merci particulier à Bertrand, mon chat de 20 ans qui est toujours là pour me soutenir.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

RÉSUMÉ

Un planificateur automatique étant capable de donner aisément et rapidement des itinéraires pour véhicule électrique s'avère nécessaire considérant le nombre croissant de ces véhicules en circulation et le peu de bornes de recharge disponibles. Ce planificateur devrait être en mesure de tenir compte de plusieurs facteurs, tels que l'emplacement des bornes de recharge et leur probabilité d'occupation, le modèle stochastique de charge-décharge de la batterie ou encore, la topographie ou même la température. En plus de considérer les facteurs précédents, la rapidité de calcul et la simplicité d'utilisation sont primordiales pour un planificateur digne de ce nom.

Ce mémoire vise dans un premier temps à présenter les caractéristiques du problème et les difficultés reliées à ce problème. Dans un second temps, il vise à présenter les forces et les faiblesses des planificateurs existants pour véhicule électrique et à présenter en détail les algorithmes utilisés par ceux-ci. Dans un troisième temps, il présente deux contributions nouvelles. La première est une technique de regroupement de bornes permettant de diminuer le temps de calcul de chaque requête par le planificateur. Cette technique a permis de diviser par 8 le nombre de noeuds considérés dans le graphe des bornes et de diviser par 35 le temps de calcul de la recherche de chemins dans ce graphe. La seconde technique permet de considérer le temps espéré d'attente à chaque borne dans le planificateur, alors que les planificateurs existants ne prenaient en compte que le temps de déplacement et le temps de recharge. La considération du temps d'attente par le planificateur a permis de diminuer de $17,3$ minutes en moyenne le temps total d'exécution des itinéraires retournés. Cette technique utilise un modèle probabiliste permettant d'estimer l'occupation des bornes et le temps espéré d'attente, qui a été généré à partir de données réelles du circuit électrique, le réseau public de bornes de recharges au Québec.

MOTS-CLÉS : planification, itinéraires, véhicule électrique, ressources, incertitude, recharge, regroupement, attente, modèle probabiliste, chemins alternatifs.

INTRODUCTION

Dans les dernières années, la notion d'intelligence artificielle (IA) est devenue une notion commune et il n'est pas rare d'en entendre parler dans les journaux et les émissions télévisées grand public. On entend principalement parler de l'apprentissage (*learning*), une des branches de l'IA. Plusieurs techniques et sujets d'étude propres à l'IA ne font cependant pas partie de cette branche. Une autre grande branche, appelée **planification automatique** (*automated planning & scheduling*), quoique moins médiatisée, est également un domaine de recherche actif et permet, tout comme l'apprentissage, de résoudre plusieurs classes de problèmes variés. La planification vise à concevoir des algorithmes capables d'automatiser la prise de décisions complexes, comme la sélection et l'ordonnancement d'actions. Elle s'intègre dans une multitude de domaines que ce soit en robotique, en aérospatiale, dans les jeux vidéo ou encore dans les systèmes d'aide à la décision.

Parallèlement aux avancées technologiques, l'environnement est un autre sujet fortement discuté dans plusieurs médias et gouvernements. En 2015, les 195 délégations de pays présentes à la COP21 ont approuvé l'accord de Paris sur le climat. Cet accord a pour objectif de réduire les émissions de gaz à effet de serre, dont le CO_2, dans le but de freiner les changements climatiques causés par celles-ci. Plusieurs solutions ont été proposées pour atteindre les objectifs de réduction de l'accord. Une d'entres elles repose sur le développement de nouvelles technologies permettant des modes de transports utilisant des énergies renouvelables.

C'est dans ce contexte que les **véhicules électriques** (VÉ) sont de plus en plus prisés. Par exemple, on peut voir respectivement l'évolution du marché des VÉ

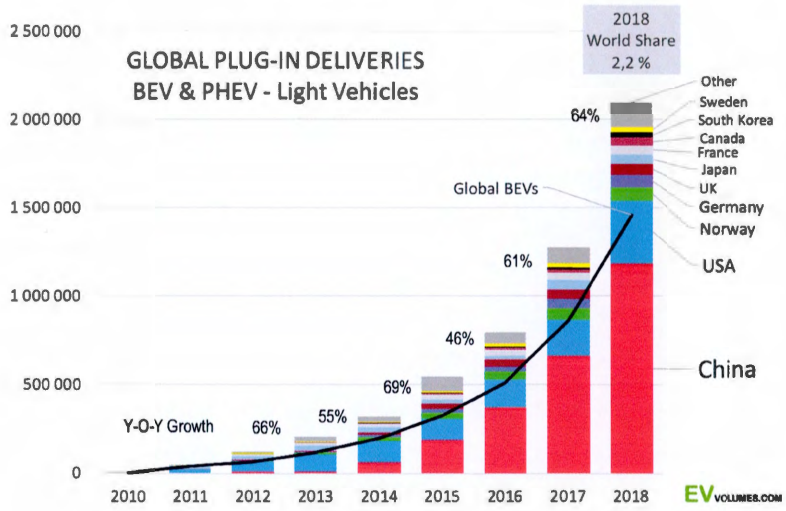


FIGURE 0.1: Marché mondial des VÉ 2010–2018

de 2010 à 2018 (Irle, 2018) et les prévisions de Bloomberg des ventes jusqu'en 2040 (Randall, 2016) aux figures 0.1 et 0.2. Le nombre de VÉ a crû très vite dans les dernières années, mais croîtra encore plus rapidement dans le futur. L'élaboration d'un bon planificateur pour VÉ est donc un problème de recherche pertinent et actuel d'un point de vue sociétal.

Au Québec, il y a actuellement plus de 40 000 VÉ immatriculés, soit près de 50 % des VÉ au Canada (Bonneau, 2019). Ils y sont plus populaires qu'ailleurs grâce à la subvention provinciale de 8 000$ à l'achat introduite en 2012 qui s'additionne à la nouvelle subvention fédérale de 5 000$ introduite en 2019. Le faible coût de l'électricité au Québec, l'une des moins chères en Amérique du Nord, contribue également à leur popularité. La nouvelle technologie de batterie, offrant une capacité plus grande et coûtant moins chère à produire, augmente quant à elle mondialement l'intérêt des individus envers les VÉ.

Malgré ces éléments encourageants, un frein à leur adoption à grande échelle sub-

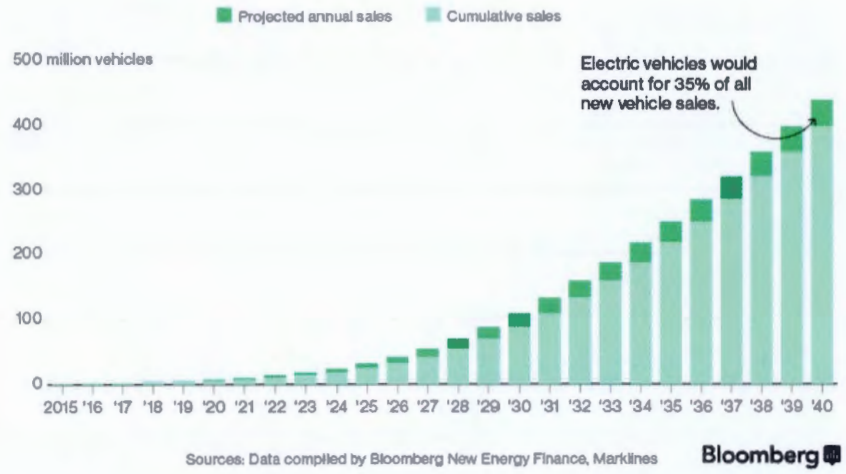


FIGURE 0.2: Projection de l'évolution des ventes de VÉ 2015–2040

siste : l'autonomie des VÉ reste moins élevée par rapport aux véhicules conventionnels de la même gamme. De plus, le nombre de stations de recharge électrique demeure faible par rapport au nombre de stations-service à essence. Cela implique que les utilisateurs de VÉ se doivent de planifier de façon efficace leurs itinéraires allant d'un point A vers un point B pour ne pas tomber en panne, particulièrement lors de longs trajets. Cette autonomie limitée et ce manque d'infrastructure dédié aux VÉ peut mener à un phénomène appelé *anxiété des conducteurs* qui a été étudiée dans quelques recherches, dont celle de Neubauer et Wood (2014). Pour combattre celle-ci et ainsi contribuer à la démocratisation des VÉ, ce mémoire développe et applique des méthodes issues de la planification automatique au problème de planification d'itinéraires pour VÉ. Ce problème n'est pas résoluble avec les méthodes classiques de planification et nécessite donc une analyse rigoureuse et l'utilisation de stratégies non triviales. En effet, il se place dans la classe des problèmes de planification nécessitant la gestion efficace de *ressources* (ex. : énergie) *sous incertitude* (ex. : la disponibilité des bornes), qui sont reconnus comme étant des problèmes difficiles à résoudre dans un temps raisonnable.

Un **planificateur d'itinéraires pour VÉ** (PIVÉ) doit être capable de recommander des itinéraires optimaux ou quasi optimaux pour VÉ en tenant compte de divers facteurs. Les caractéristiques des véhicules électriques, les variables à prendre en compte, les caractéristiques des bornes de recharge ainsi que les planificateurs grand public actuellement disponibles sont présentés au **chapitre 1**.

Avant de penser aux méthodes avancées pouvant être développées et incluses, il est utile d'avoir un planificateur de base permettant de résoudre une version simplifiée du problème à partir duquel on pourra construire les diverses améliorations, et qui pourra servir de référence (*baseline*) pour mesurer l'apport de ces améliorations. Le **chapitre 2** propose une définition formelle du problème de planification pour VÉ et présente un tel planificateur de base.

Dans les dernières années, certaines méthodes ont été développées pour permettre l'émergence de bons planificateurs pour VÉ. Le **chapitre 3** présente les principales et détaille deux d'entres elles : Energy-A*, permettant de considérer la recharge de batterie induite par un freinage ou une baisse d'altitude, et la contraction hiérarchique de graphes, permettant une accélération substantielle du temps de calcul pour obtenir un itinéraire complet.

Étant donné que plusieurs bornes de recharge sont près les unes des autres sur la carte (ex. : environ la moitié des bornes du Québec sont à Montréal), il semble raisonnable de regrouper certaines bornes ensemble pour permettre d'accélérer le temps de calcul tout en maintenant un faible écart avec la solution optimale. Le **chapitre 4** présente une technique d'accélération basée sur un regroupement de bornes de recharge, qui est la première contribution originale proposée.

Bien que cela puisse faire une grande différence, les PIVÉ actuellement disponibles ne tiennent malheureusement pas compte du temps espéré d'attente lorsqu'ils cherchent l'itinéraire optimal. Le **chapitre 5**, qui présente la seconde contri-

bution originale, propose une modification du planificateur pour y introduire la considération du temps espéré d'attente. Pour ce faire, un modèle probabiliste dépendant du temps (ex. : le jour et l'heure) est introduit. Celui-ci permet de prédire la probabilité d'occupation à chaque borne et, lorsqu'occupée, le temps espéré d'attente. Deux techniques utilisant ce modèle et permettant de minimiser les imprévus dus à une borne plus occupée que prévu sont par la suite introduites et comparées avec le planificateur de base.

Ces analyses théoriques ont mené à la création d'un planificateur, appelé VEPlan et disponible en ligne à l'adresse `https://gitlab.com/jaja360/Veplan` sous licence MIT, utilisable par les utilisateurs de VÉ. Ce planificateur comprend les méthodes présentées dans ce mémoire incluant les deux contributions. Différents outils permettant l'utilisation simple du planificateur ont été développés (application Java, site web et interface console). L'implémentation du planificateur ainsi que les manières de l'utiliser sont présentées à l'**annexe A**.

CHAPITRE I

PLANIFICATION POUR UN VÉHICULE ÉLECTRIQUE

Tout objectif sans plan n'est qu'un souhait.

— Antoine de Saint-Exupéry

Afin d'être adopté et utilisé massivement, un bon planificateur pour VÉ doit avoir certaines caractéristiques. Parmi les plus importantes, on peut penser à :

- **Convivialité** : Le système doit être simple et agréable à utiliser.
- **Rapidité** : Le système doit être en mesure de fournir un itinéraire à l'utilisateur dans un temps raisonnable.
- **Adaptabilité** : Le système calcule les plans en s'adaptant le plus possible aux diverses variables (probabilité d'occupation des bornes, vitesse de recharge, topographie de la région, probabilité de réussite, etc.).
- **Qualité des solutions** : Le système donne à l'utilisateur un plan optimal ou presque optimal, par rapport à la combinaison des divers facteurs considérés.

Pour construire un tel planificateur, la première étape est de bien définir le problème et les variables à considérer. Ce chapitre présente les informations pertinentes à connaître par rapport aux VÉ et aux bornes de recharge, respectivement aux sections 1.1 et 1.2. Il présente également à la section 1.3 les outils d'aide à la planification pour VÉ actuellement disponibles sur le marché. Une analyse des

forces et faiblesses de chacun d'eux est ensuite faite pour montrer leurs lacunes, et donc les opportunités d'amélioration.

1.1 Caractéristiques des VÉ

Il existe actuellement 4 types de véhicule électrique en circulation (CAA, 2019) :

- **VÉPC : Véhicules électriques à pile à combustion** : Ce sont des véhicules à hydrogène, comme la Toyota Mirai. Leur nombre et le nombre de stations à hydrogène sont pour l'instant négligeables (actuellement deux stations au Québec). La plupart des techniques vues dans ce mémoire s'appliquent aussi pour les VÉPC, mais l'enjeu est moins pertinent pour l'instant.
- **VH : Véhicules hybrides** : Ce sont des véhicules dont la batterie est chargée exclusivement par récupération d'énergie lors des freinages et par le moteur électrique (c.-à-d. qu'elles ne peuvent être chargées à des bornes de recharge).
- **VHR : Véhicules hybrides rechargeables (ou enfichables)** : Les VHR permettent d'être chargés manuellement grâce à des bornes. L'autonomie typique de ces véhicules (lorsque l'essence n'est pas utilisée) varie entre 20 et 80 km, dépendamment des modèles.
- **VÉB : Véhicules électriques à batterie** : Ces véhicules sont 100 % électriques et doivent être chargés à des bornes pour pouvoir se déplacer.

Les VHR et les VÉB peuvent, comme les VH, récupérer de l'énergie lors des freinages. Dans la suite du mémoire, seuls les VÉB sont considérés et le sigle VÉ sera utilisé comme synonyme de VÉB (mais certaines techniques qui seront vues s'appliquent aussi aux VHR et aux VH).

Les VÉ possèdent plusieurs caractéristiques qui les distinguent des véhicules conventionnels et qui doivent être prises en compte pour obtenir une solution adaptée. La plus importante d'entre elles est **l'autonomie du véhicule** (en km). Celle-ci, dépendamment du modèle, varie actuellement entre 92 km pour une Smart EQ Fortwo cabriolet et 595 km pour une Tesla modèle S Long Range (Ressources naturelles Canada, 2019). La consommation énergétique du véhicule (mesurée en joules/km), et donc l'autonomie, est aussi influencée par des facteurs externes (Wu **et al.**, 2013; Yuksel et Michalek, 2015; De Cauwer **et al.**, 2015). Par exemple, **la pente de la route** (l'angle d'inclinaison) qui affecte la consommation d'énergie (Liu **et al.**, 2017), et **la température ambiante** qui affecte linéairement, entre $-15°C$ et $20°C$, l'autonomie (Lindgren et Lund, 2016). Cela s'explique principalement par le fait que la performance des batteries varie grandement selon la température. Des modèles de simulation ont été développés pour estimer la consommation énergétique des VÉ dans le contexte de la planification de chemins (Genikomsakis et Mitrentsis, 2017).

Comme mentionné précédemment, une autre caractéristique distinctive des VÉ est de pouvoir se recharger grâce à une **récupération d'énergie** lors d'un freinage ou d'une perte d'énergie potentielle gravitationnelle (baisse d'altitude). Comme présenté au chapitre 3, ce simple fait empêche d'utiliser directement certaines techniques classiques de recherche de chemins dans un graphe et est donc une caractéristique primordiale à prendre en compte.

Les autres caractéristiques distinctives des VÉ sont en lien avec la batterie et le système de recharge. Par exemple, **la non-linéarité de la courbe de recharge** de la batterie, **le temps de recharge** (il est parfois mieux de ne pas recharger à 100 %) ainsi que **l'occupation des bornes** sont d'autres facteurs à prendre en compte qui ne sont pas importants pour les véhicules conventionnels (car on n'a généralement pas à attendre pour faire le plein d'essence à une station-service).

1.2 Caractéristiques des bornes de recharge

Un bon planificateur pour VÉ doit non seulement prendre en compte les caractéristiques propres aux VÉ, mais aussi les propriétés des bornes. Les informations qui suivent proviennent toutes d'un guide technique d'Hydro-Québec (2015).

Seuls trois niveaux de bornes sont disponibles en Amérique du Nord. Les caractéristiques de chacun d'eux sont présentées au tableau 1.1. Dans celui-ci, CA veut dire courant alternatif, tandis que CC veut dire courant continu. Le temps donné pour le niveau 3 est pour recharger la batterie à 80 % de sa capacité (car la recharge ralentit automatiquement après ce seuil pour ne pas endommager la batterie).

TABLE 1.1: Comparatif des trois niveaux de bornes de recharge

	Niveau 1	Niveau 2	Niveau 3
Tension	120 V	208 ou 240 V	200 à 450 V
Type de courant	CA	CA	CC
Puissance utile[1]	1,4 kW	7,2 kW	50 kW
Temps de recharge (15 kWh)	12 h	3 h	20 min
Connecteur	J1772	J1772	J1772 Combo, CHAdeMO, Supercharger

Tous les VÉ peuvent utiliser les bornes de niveau 1 qui s'utilisent avec une prise électrique de maison (120V en Amérique du Nord). Le niveau 2 correspond quant à lui à la majorité des bornes publiques pour VÉ. Tous les VÉ ont une prise pour les utiliser. Finalement, les bornes de niveau 3, à recharge rapide, aussi appelées **bornes de recharge à courant continu** (BRCC), sont les bornes les plus rapides actuellement disponibles. Deux normes existent pour les BRCC :

1. La puissance réelle consommée, telle que mesurée par un wattmètre.

CHAdeMO, une norme japonaise, et J1772 Combo, une norme nord-américaine. Les BRCC publiques disposent pour la plupart des deux types de connecteurs. Les VÉ disposent d'un de ces connecteurs ou à tout le moins d'un adaptateur. Au Québec, le réseau public de bornes pour VÉ s'appelle le **Circuit électrique** (Add Énergie, 2019). En juin 2019, le réseau comptait 1889 bornes (niveaux 2 et 3), dont 184 BRCC. Certaines bornes du réseau sont hors du Québec (ex. : 20 bornes dans l'est de l'Ontario), mais la grande majorité est au Québec.

La différence de temps de recharge entre les VÉ et les véhicules conventionnels, pouvant atteindre quelques heures si la borne n'est pas de niveau 3, peut inciter des consommateurs à préférer un véhicule conventionnel. Par conséquent, les bornes de niveau 3 sont essentielles pour l'adoption massive des VÉ. Or, les 184 BRCC réparties sur un territoire aussi vaste que le Québec sont loin d'être suffisantes pour éviter l'anxiété des conducteurs. Leur nombre élevé, mais également leur bon positionnement, est essentiel pour permettre aux conducteurs d'avoir des itinéraires qui ne nécessitent pas de grands détours. Lors du positionnement des futures bornes, plusieurs facteurs sont à considérer, tels que l'accessibilité, la rentabilité, le temps d'attente et le coût d'installation.

Bien que ce mémoire vise d'abord la planification d'itinéraires sur une carte où les bornes sont déjà placées, plusieurs chercheurs se sont intéressés au problème du placement optimal des bornes, appelé dans la littérature le EVCSPP (*electric vehicle charging station placement problem*). Il est intéressant de survoler ce qui se fait dans ce domaine, car la connaissance des propriétés du positionnement des bornes peut avoir un impact sur l'algorithme de planification utilisé. Parmi ces chercheurs, Gopalakrishnan **et al.** (2016) tiennent compte des quatre facteurs mentionnés. Le temps d'attente à chaque borne est pris en compte à l'aide d'une file d'attente $M/M/N$ (Notation de Kendall (Gelenbe et Pujolle, 1982)) et d'un processus de Poisson. Ils utilisent une régression linéaire multivariée pour tenir

compte d'un grand nombre de facteurs dans le placement des bornes. Lam **et al.** (2014) proposent quant à eux de considérer les déplacements dans une ville, en tenant compte davantage des facteurs humains que des facteurs technologiques. Ils démontrent que le problème de placement est NP-difficile [2] et proposent quatre méthodes pour résoudre le problème. Deux d'entre elles sont basées sur l'optimisation d'un programme linéaire, tandis que les deux autres utilisent respectivement une technique gloutonne et une technique d'optimisation inspirée de la nature et basée sur les réactions chimiques (Chemical Reaction Optimization, CRO).

Funke **et al.** (2016) s'intéressent de leur côté à un placement de bornes garantissant le détour maximal nécessaire pour faire n'importe quel trajet. L'approche proposée s'appuie sur le problème du *hitting set* (le dual du problème de recouvrement minimal, un problème fondamental d'algorithmique). Cette approche permet de fixer un paramètre $D \in \mathbb{R}$, qui est la longueur du détour maximal tolérable pour se rendre à une borne à partir d'un chemin optimal sans recharge. Cela correspond à une approche mitoyenne par rapport à deux autres approches publiées (Funke **et al.**, 2015; Funke et Storandt, 2013) qui demandaient respectivement que l'on puisse se rendre sans détour de n'importe où vers n'importe où sur une carte ($D = 0$) et qu'on puisse faire de même, mais sans limite de détour ($D = \infty$). L'article propose également un algorithme de planification, mais celui-ci n'est valide que si les bornes ont été placées sur la carte par leur algorithme.

1.3 Outils existants d'aide à la planification

Plusieurs planificateurs publics sont disponibles actuellement sur le web. Cette section présente les avantages, mais surtout les lacunes, de chacun d'eux dans le but de montrer la nécessité de développer de nouveaux algorithmes.

---

2. Pour un survol des classes de complexité, voir le chapitre 7 du livre de Sipser (2013)

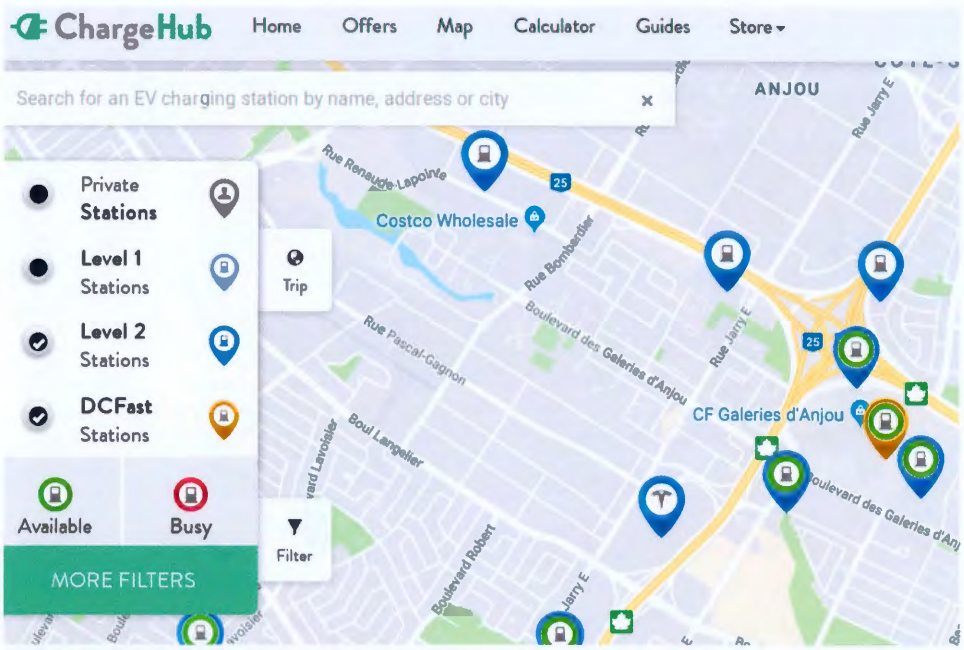


FIGURE 1.1: Site de ChargeHub

1.3.1 ChargeHub

Le site `https://chargehub.com` est opéré par la compagnie MogileTech. Il propose des outils, comme une calculatrice d'économie d'essence et des informations générales sur les VÉ. Une carte est disponible (présentée à la figure 1.1) et permet de chercher des bornes à proximité d'une adresse donnée. Une fois une borne sélectionnée, des informations la concernant s'affichent, comme le tarif et la disponibilité en temps réel. Par contre, le calculateur de trajets n'est pas intuitivement accessible (il a fallu aller dans l'aide du site pour savoir qu'il y en avait un). De plus, le site n'affiche que le plus court chemin (sans recharge), comme le ferait Google Maps. C'est ensuite à l'utilisateur de choisir manuellement sur la carte les bornes qui lui semblent être appropriées.

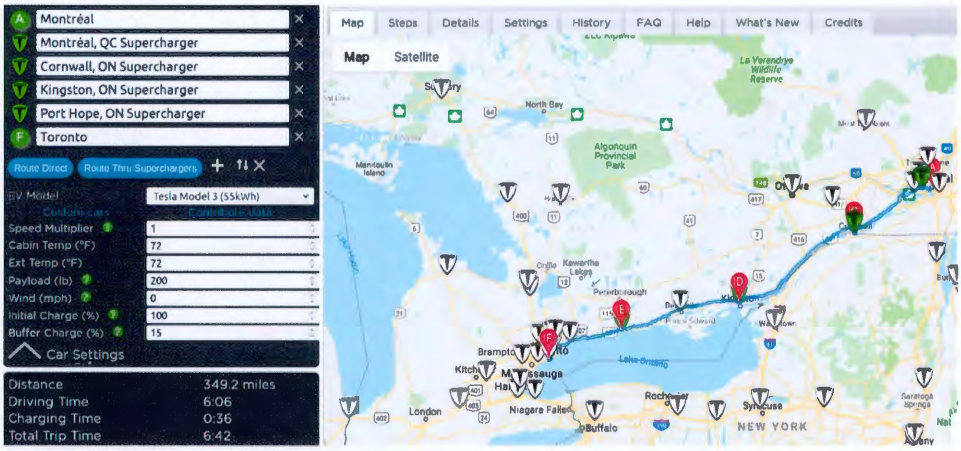


FIGURE 1.2: Site de EV Trip Planner

1.3.2 EV Trip Planner

Le planificateur disponible sur le site `https://evtripplanner.com/planner` permet d'afficher le chemin optimal entre deux points avec une route directe (sans recharge) ou bien avec une route passant par des stations de recharge. On peut aussi spécifier des destinations intermédiaires (*waypoints*). Il permet de spécifier le vent moyen, la température de la cabine et de l'extérieur ou encore la charge utile (poids du conducteur, du passager, des bagages, etc.) mais rien sur le site n'indique l'effet de ces paramètres sur le résultat obtenu. Il serait intéressant que le vent et la température soient considérés automatiquement (avec des données en temps réel) sans que l'utilisateur n'ait à spécifier ces paramètres manuellement. Il est possible de choisir le véhicule dans une liste prédéfinie, mais à quelques exceptions près, seuls les modèles Tesla sont dans la liste. Les plus grandes lacunes de ce planificateur sont qu'il n'inclut que les bornes *Tesla Superchargers* et qu'il n'a pas été mis à jour depuis quelques années. Le calcul de l'itinéraire en tant que tel (en anglais, *routing*) est délégué à l'API de Google Maps et de MapQuest. La figure 1.2 présente l'interface.

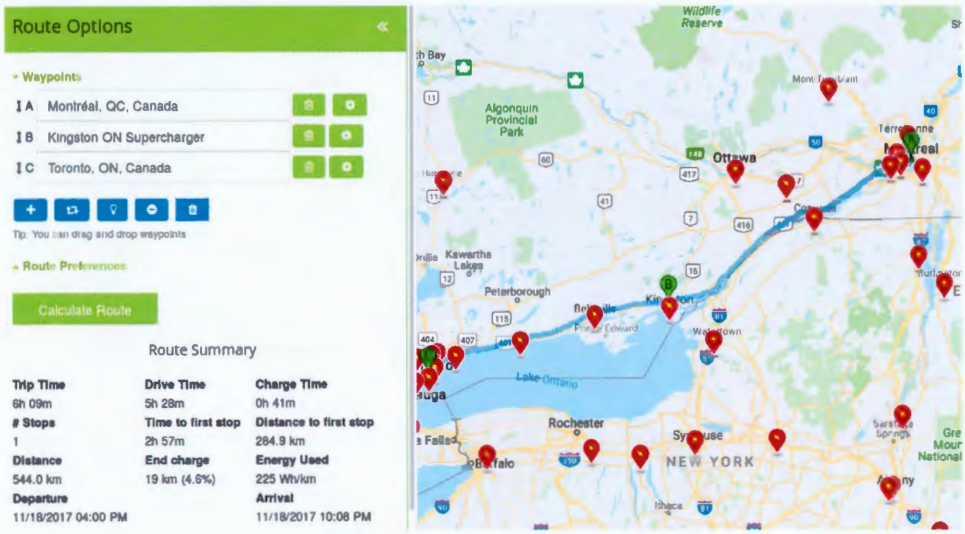


FIGURE 1.3: Site de EV Tripping

1.3.3 EV Tripping

Le site EV Tripping (à l'adresse `https://evtripping.com/route`) a quelques points communs avec EV Trip Planner (il affirme même s'en être inspiré). Il permet de spécifier des destinations intermédiaires et affiche le chemin optimal avec recharge passant par ces destinations. Par contre, il faut créer un compte (gratuit) pour pouvoir utiliser le planificateur. De l'aveu des créateurs du site, l'inscription obligatoire est pour diminuer le nombre d'usagers pour empêcher le serveur de se faire surcharger de requêtes. Malheureusement, même en sachant cela, les quelques requêtes testées sur ce site (qui ne nécessitaient pas un énorme trajet, tel que Montréal-Toronto) ont pris près de 30 secondes avant d'afficher une solution, ce qui laissait penser que le site ou l'ordinateur avait planté. Le problème principal de ce site est donc la vitesse de calcul trop lente pour plusieurs conducteurs. De plus, tout comme EV Trip Planner, seules les bornes *Superchargers* sont prises en compte. L'interface est présentée à la figure 1.3.

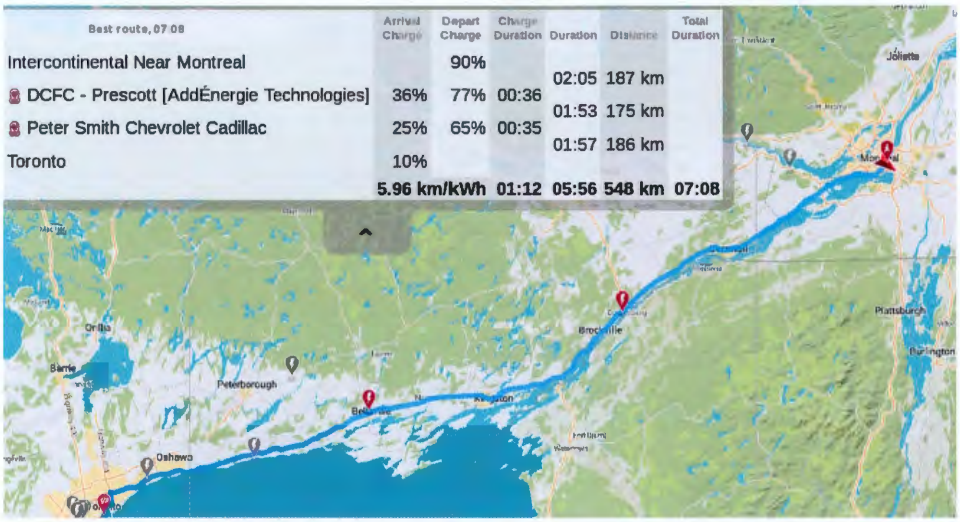


FIGURE 1.4: Site de A Better Route Planner

1.3.4 A Better Route Planner

A Better Route Planner (`https://abetterrouteplanner.com`), qu'on peut voir à la figure 1.4, est le planificateur qui semble le plus récent, le plus accessible (c'est le premier résultat dans Google), le plus visuellement clair, et le plus complet. Tous les principaux modèles et marques de VÉ sont sélectionnables (pas seulement des Tesla) et les bornes du réseau Flo et du Circuit électrique sont incluses. On peut spécifier tous les paramètres qui étaient aussi dans EV Trip Planner et dans EV Tripping. Une amélioration par rapport à ces derniers est que pour chaque borne faisant partie de l'itinéraire retourné, quelques choix alternatifs sont proposés à l'utilisateur (pour lesquels le temps et la distance supplémentaire induite par ces choix alternatifs sont spécifiés). Le temps de calcul est meilleur que pour les planificateurs précédents, mais il laisse néanmoins place à l'amélioration. Par exemple, l'itinéraire Montréal-Vancouver a pris 22 secondes à calculer, ce qui reste assez long pour un utilisateur lambda qui veut tester quelques chemins.

Somme toute, les quatre principaux planificateurs d'itinéraires pour VÉ dispo-

nibles au grand public ont tous des lacunes, dont deux qui sont présentes dans chacun d'eux. La première est le temps de calcul : soit les planificateurs ne sélectionnent pas eux-mêmes les bornes à utiliser et laissent la sélection à l'utilisateur (ce qui est un problème beaucoup plus simple), soit la sélection est faite, mais le temps de réponse du site est élevé. La seconde lacune est la non-considération, par tous les planificateurs disponibles, du temps espéré d'attente à chaque borne dans le choix du chemin optimal. Ces quelques observations justifient la pertinence des deux contributions de ce mémoire.

CHAPITRE II

FORMALISME ET PLANIFICATEUR DE BASE

Simplicity is prerequisite for reliability.

— Edsger Dijkstra

2.1 Formulation du problème

La théorie des graphes est utilisée pour représenter le problème de planification pour VÉ. Les définitions qui suivent présentent les différents aspects du problème et sont autosuffisantes. Pour une introduction à la théorie des graphes, le livre *A first course in graph theory* de Chartrand et Zhang (2012) est suggéré, principalement les chapitres 1, 2, 6, 7 et 12.

Les définitions 2.1 à 2.4 formalisent respectivement le réseau de transport, le problème de planification pour véhicules électriques, les solutions à ce problème et ce qu'on entend par solution optimale.

Définition 2.1. Un **réseau de transport** M est un quintuplet (V, E, λ, μ, S), où (V, E) est un graphe orienté et λ, μ sont des étiquetages des arcs. Plus précisément :

- V est l'ensemble des points (latitude, longitude) sur la carte (nœuds) ;
- E est l'ensemble des segments de route (arcs) ;
- $\lambda\colon E \to \mathbb{R}^+$ donne la longueur (en mètres) de chaque arc ;

— μ: $E \to \mathbb{R}^+$ donne la vitesse espérée de parcours (en m/s) sur chaque arc ;

— S est l'ensemble des stations (bornes) de recharge.

Chaque station $s \in S$ est associée au nœud $v_s \in V$ le plus près. Avec cette association, on peut supposer que $S \subseteq V$.

Remarque. Il est possible d'utiliser les étiquetages λ et μ pour en définir un troisième, donnant pour chaque arc le temps espéré de parcours. Cela donne une formulation du réseau de transport sous forme de temps plutôt que de distance. Les deux formulations sont équivalentes.

L'étiquetage μ peut se baser sur des données empiriques donnant la vitesse moyenne de parcours des véhicules sur chaque segment de route, ou simplement sur la vitesse maximum légale permise sur le segment.

Définition 2.2. Une instance d'un **problème de planification pour VÉ** (PPVÉ) est défini par un quadruplet $(M, \rho, \alpha, \omega)$, où

— M est le réseau de transport ;

— $\rho \in \mathbb{R}^+$ est l'autonomie du VÉ ;

— $\alpha, \omega \in V$ sont les points de départ et d'arrivée.

Remarque. Il est possible de généraliser au cas où $\alpha, \omega \notin V$ en utilisant un Arbre-KD (Bentley, 1975) pour trouver le nœud le plus près de α et de ω.

Définition 2.3. Une **solution** d'un PPVÉ $(M, \rho, \alpha, \omega)$ est un doublet (P, Q), où

— P est une suite $\langle P_0, P_1, \ldots, P_k \rangle$, où $P_i \in V$ pour $i = \{0, 1, \ldots, k\}$;

— Q est une sous-suite $\langle P_{i_0}, \ldots, P_{i_{b+1}} \rangle \subseteq P$ qui contient les b stations de recharge utilisées par la solution, et qui contient α et ω ;

— $P_{i_0} = P_0 = \alpha$;

— $P_{i_{b+1}} = P_k = \omega$;

— pour tout $j \in \{0, 1, \ldots, b\}$, $\texttt{dist}(Q_j, Q_{j+1}) \leq \rho$ (où `dist` donne la distance dans le graphe entre deux nœuds), c.-à-d. que deux nœuds consécutifs dans Q sont toujours à une distance d'au plus l'autonomie.

En d'autres mots, P est la suite de nœuds que le VÉ doit parcourir, et Q est une sous-suite contenant les stations où le VÉ doit s'arrêter pour se recharger (ainsi que α et ω). L'objectif est de trouver une solution au PPVÉ qui minimise le temps total de l'itinéraire, incluant le temps de déplacement, le temps de recharge et le temps d'attente. La prochaine définition formalise ce qu'on entend par une solution optimale.

Définition 2.4. Une **solution optimale** à un PPVÉ est une solution (P, Q), telle que définie à la définition 2.3, qui minimise la fonction objectif

$$\mathrm{Z}(P, Q) = \mathrm{TD}(P) + \mathrm{TR}(Q) + \mathrm{TA}(Q),$$

où TD, TR et TA sont des fonctions donnant respectivement le temps espéré de déplacement, de recharge et d'attente de la solution. Elles sont données par :

$$\mathrm{TD}(P) = \sum_{i=0}^{k-1} \frac{\lambda(P_i, P_{i+1})}{\mu(P_i, P_{i+1})},$$
$$\mathrm{TR}(Q) = \sum_{i=1}^{b} \mathrm{TRB}(Q_i),$$

où $\mathrm{TRB}(Q_i)$ est le temps de recharge espéré à la borne Q_i lorsqu'on considère l'état de charge de la batterie et le niveau de la station. La fonction TA est quant à elle définie au chapitre 5 lorsque nous considérons le temps d'attente.

La figure 2.1 présente un exemple qui illustre les différents concepts qui viennent d'être introduits. Celle-ci montre un réseau de transport M, où $V = \{A, \ldots, I\}$ et

$S = \{E, F\}$ (en bleu). Pour simplifier la représentation, les nombres sur les arcs représentent le temps espéré de parcours (donné par la fonction $\frac{\lambda}{\mu}$).

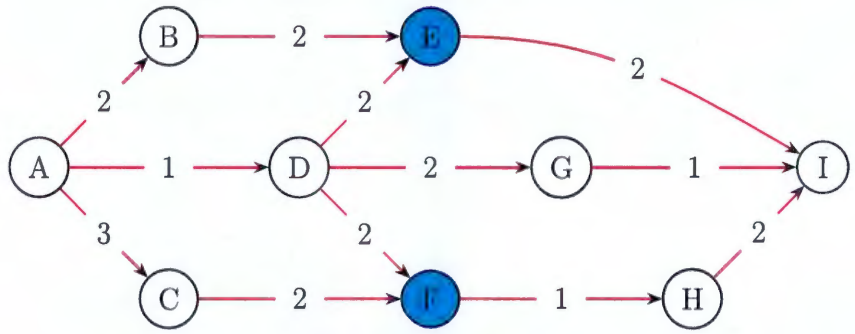


FIGURE 2.1: Exemple d'un réseau de transport

Le quadruplet $(M, 3, A, I)$ représente un PPVÉ où le point de départ est $\alpha = A$, celui d'arrivée est $\omega = I$ et l'autonomie du véhicule est de $\rho = 3$ unités (ex. : km). Dans cet exemple, le quadruplet

$$(P = \langle A, D, G, I \rangle, Q = \langle A, I \rangle)$$

n'est pas une solution, car $\texttt{dist}(A, I) = 4 > \rho$. Par contre, le quadruplet

$$S_1 = (P_1 = \langle A, D, F, H, I \rangle, Q_1 = \langle A, F, I \rangle)$$

en est une, car $\texttt{dist}(A, F) = 3 \leq \rho$ et $\texttt{dist}(F, I) = 3 \leq \rho$.

Si on suppose, par souci de concision, que TR $\equiv 2$ et que TA $\equiv 1$, alors le coût de la solution S_1 est

$$\begin{aligned} Z(P_1, Q_1) &= \text{TD}(P_1) + \text{TR}(Q_1) + \text{TA}(Q_1) \\ &= (1 + 2 + 1 + 2) + 2 + 1 \\ &= 9. \end{aligned}$$

Par contre, S_1 n'est pas une solution optimale. En effet, la solution

$$S_2 = (P_2 = \langle A, D, E, I \rangle, Q_2 = \langle A, E, I \rangle)$$

a un coût de $Z(P_2, Q_2) = (1 + 2 + 2) + 2 + 1 = 8 < Z(P_1, Q_1)$. Il est possible de montrer que S_2 est une solution optimale (dans l'exemple, elle est même unique).

2.2 Planificateur de base

Cette section présente un algorithme de base faisant office de planificateur qui sera utilisé comme référentiel dans les tests pour les différentes techniques proposées. Il ne considère pas le temps d'attente, et considère un temps de recharge constant (ce dernier dépend en pratique du niveau de charge restant, de la température, du niveau de la borne, etc.).

Tout d'abord, le planificateur précalcule la distance entre chaque paire de bornes de recharge et stocke les valeurs dans une matrice $D = (D_{ij})$ où $D_{ij} = \texttt{dist}(s_i, s_j)$. Le chemin optimal permettant de joindre chacune de ces paires de bornes est également stocké en mémoire.

Ensuite, un graphe (V', E') (qu'on appellera le **s-graphe**) est dérivé de (V,E). Celui-ci contient les nœuds qui sont des bornes (les éléments de S). Les éléments de E' sont les arcs joignants chaque paire de bornes (dont on connait la longueur en utilisant la matrice D). En supposant que le réseau routier est connexe, le s-graphe est à ce moment-ci un graphe complet.

Le planificateur est alors prêt à traiter les requêtes. Pour chaque requête (ρ, α, ω) reçue par le planificateur, les points de départ et d'arrivée (α, ω) sont temporairement ajoutés au s-graphe. L'algorithme de Dijkstra (Dijkstra, 1959) est alors utilisé deux fois : une fois en partant du départ, et une autre fois en partant de

l'arrivée sur le graphe transposé. Cela permet de trouver les distances du départ jusqu'à chaque borne, et de chaque borne vers l'arrivée. Pour chacune des distances trouvées ayant une longueur d'au plus ρ, un arc est ajouté temporairement au s-graphe. Pour ce qui est des arcs borne-borne, ceux ayant une autonomie de plus de ρ ne doivent pas être considérés (par exemple, en les supprimant du s-graphe et en les remettant après la requête, ou en modifiant l'algorithme de recherche pour ne pas parcourir les arcs dépassant l'autonomie).

À ce stade, il suffit de chercher le plus court chemin dans le s-graphe (de α à ω) en utilisant l'algorithme de Dijkstra ou, encore mieux, l'algorithme A* (Hart **et al.**, 1968) avec la distance du grand cercle comme heuristique[1] (qui est avantageux si le s-graphe a une assez grande taille), pour obtenir une suite Q respectant la dernière partie de la définition 2.3 (la contrainte d'autonomie). Cette contrainte est satisfaite, car chaque nœud intermédiaire dans le s-graphe est une borne (donc il est possible de se recharger à chaque nœud traversé) et les arcs ont une distance d'au plus l'autonomie, par construction. La suite P peut ensuite être trouvée en utilisant Q et les chemins déjà calculés (par les deux appels à `Dijkstra`, et par les précalculs qui ont permis de trouver la matrice D).

Le pseudocode résumant la discussion précédente est présenté à l'algorithme 2.1. Le théorème 2.5 assure que le planificateur donne bien le résultat attendu. L'exécution d'une requête en utilisant ce planificateur a une complexité temporelle de $\mathcal{O}(n \log n + m)$[2], c.-à-d. la même complexité que l'algorithme de Dijkstra, lorsqu'implémenté avec un monceau de Fibonacci (Fredman et Tarjan, 1987). Dans un réseau de transport réaliste, le degré maximal d'un nœud est borné par un petit nombre (basé sur le plus grand nombre d'intersections de routes en un seul

1. C'est la plus petite distance entre deux points sur une sphère.

2. Dans ce mémoire, on utilise $n = |V|$ et $m = |E|$ à moins d'avis contraire.

nœud). Par conséquent, la complexité temporelle peut être simplifiée à $\mathcal{O}(n \log n)$.

Algorithme 2.1 Planificateur de base

1: Calculer la matrice D et le chemin le plus court reliant chaque paire de bornes
2: Construire le s-graphe contenant chaque borne de recharge
3: **pour tout** requête (ρ, α, ω) **faire** ▷ Tant qu'il y a des requêtes à traiter
4: Exécuter `Dijkstra` partant de α sur le graphe original
5: Exécuter `Dijkstra` partant de ω sur le graphe transposé
6: Ajouter α et ω au s-graphe et ajouter les arcs de longueur $\leq \rho$
7: Exécuter l'algorithme A* de α à ω sur le s-graphe pour trouver la suite Q
8: Trouver la suite P à l'aide de Q en utilisant les chemins déjà calculés
9: Retirer α et ω du s-graphe et réajuster les arcs
10: Envoyer la solution (P, Q)
11: **fin pour**

Théorème 2.5. *L'algorithme 2.1 donne une solution optimale (au sens de la définition 2.4) lorsqu'on ne considère pas le temps d'attente.*

Preuve. Voir l'annexe B. □

Remarque. Plusieurs travaux considèrent que $\alpha, \omega \in S$ et supposent que le s-graphe est donné en entrée. Lorsque c'est le cas, alors les deux exécutions de l'algorithme de Dijkstra ne sont pas nécessaires et la complexité temporelle de l'algorithme 2.1 devient simplement $\mathcal{O}(|S| \log |S|)$, soit beaucoup plus rapide (car $|S| \ll |V|$ en général).

CHAPITRE III

TECHNIQUES EXISTANTES

If I have seen further than others, it is by standing upon the shoulders of giants.

— Isaac Newton

Le planificateur présenté au chapitre précédent permet de trouver des itinéraires pour VÉ ayant un temps total minimal, mais a néanmoins plusieurs limitations. Parmi celles-ci :

1. la batterie est supposée initialement à 100 % de charge ;
2. la recharge se fait toujours jusqu'à 100 % alors que moins de charge aurait peut-être suffi ;
3. le temps de recharge est supposé constant, alors qu'il dépend de l'autonomie restante lors de l'arrivée à la borne et qu'il n'est pas linéaire ;
4. l'effet de la vitesse sur la consommation d'énergie n'est pas considéré ;
5. l'effet du relief de la carte n'est pas considéré ;
6. la vitesse de calcul n'est pas optimisée autant qu'elle pourrait l'être ;
7. le temps d'attente aux bornes n'est pas considéré.

Ce chapitre présente les techniques existantes qui permettent d'améliorer le planificateur en éliminant certaines de ces limitations. Un survol de quelques techniques est d'abord fait à la section 3.1. Les sections 3.2 et 3.3 présentent ensuite en détail

deux méthodes, Energy-A* et les contractions de bornes, qui ont chacune à leur façon grandement influencé le domaine de la planification pour VÉ.

3.1 Survol de diverses techniques

Les premiers travaux en lien avec la planification pour VÉ ne considéraient pas l'existence de bornes de recharge. Ils tenaient compte d'autres facteurs spécifiques aux VÉ comme la possibilité pour la batterie de récupérer de l'énergie pendant le déplacement. Par exemple, Artmeier **et al.** (2010) sont les premiers à avoir considéré cela, et Eisner **et al.** (2011) ont proposé une approche similaire, mais avec une petite modification qui permet une complexité de $\mathcal{O}(n \log n + m)$ plutôt que $\mathcal{O}(n^3)$. La principale méthode permettant cela sera présentée à la section 3.2.

C'est quelques années plus tard que le problème a été généralisé pour permettre l'utilisation de bornes de recharge. Sweda et Klabjan (2012) sont les premiers à avoir considéré cette variante, qui ressemble à celle résolue par le planificateur présenté à l'algorithme 2.1. La différence est que la méthode utilisée permet de considérer les niveaux de charge (ce qui règle les trois premiers problèmes de la liste ci-haut). Pour considérer les niveaux de charge (une variable continue), une discrétisation est faite, de sorte que la solution obtenue reste optimale. En effet, les auteurs ont prouvé que toute solution optimale possède une politique de recharge où, à chaque borne, le conducteur doit soit ne pas recharger, recharger au maximum, ou recharger la quantité exacte nécessaire pour se rendre à la prochaine borne. Le désavantage de leur méthode par rapport à l'algorithme 2.1 est que le graphe considéré ne contient que des bornes, ce qui fait que le départ et l'arrivée doivent être des bornes.

Baum **et al.** (2017) considèrent également la possibilité de ne pas recharger entièrement le VÉ à chaque borne rencontrée. Ils utilisent pour ce faire le concept

d'état de charge (*State of Charge*, SoC). Ils abordent aussi le problème qui consiste à trouver, pour chaque état de charge initiale possible (dans $[0, \rho]$), le chemin optimal entre deux points. L'état de charge initiale est importante à considérer, car cela peut changer significativement le chemin optimal, par exemple si la charge initiale ne permet pas de se rendre à la première station prévue lorsqu'on n'en tient pas compte. Ils incluent également la contraction de graphes (voir section 3.3) au planificateur.

Adler **et al.** (2016) considèrent quant à eux aussi le problème de planification d'itinéraire pour VÉ avec recharges intermédiaires, et le solutionnent en utilisant un formalisme de théorie des graphes très similaire à celui utilisé au chapitre 2 (en faisant plusieurs fois l'algorithme de Dijkstra pour trouver les distances du départ aux bornes et des bornes à l'arrivée). Ils étendent le problème en proposant deux variantes : la première ajoute un nombre maximal de recharges, tandis que la seconde vise à minimiser l'anxiété maximale des conducteurs (définie comme une fonction monotone qui croit à mesure que la batterie se vide, et qui est à son minimum lorsque la batterie est pleine) dans l'itinéraire retourné.

Plusieurs travaux en lien avec la planification pour VÉ traitent d'un problème différent, mais connexe, soit celui de devoir parcourir avec un VÉ une liste d'endroits, dans n'importe quel ordre (par exemple pour livrer des colis à des clients), de façon à minimiser le temps total. Ce problème est une généralisation du problème du voyageur de commerce qui considère les contraintes d'autonomie du VÉ et les recharges intermédiaires. Roberti et Wen (2016) ont étudié ce problème, qu'ils appellent le « problème du voyageur de commerce électrique avec fenêtre de temps » (en anglais, *electric traveling salesman problem with time windows*, E-TSPTW). La technique proposée pour résoudre le problème utilise un formalisme et des algorithmes propres au domaine de la **recherche opérationnelle** (RO), c'est-à-dire que le problème est modélisé par une fonction linéaire de coût (qu'on

souhaite minimiser) sujette à des contraintes linéaires qui modélisent les différents aspects du problème.

D'autres travaux étendent ce problème à une flotte entière de VÉ qui doivent se coordonner pour visiter plusieurs nœuds. Tous les résultats récents par rapport à ce problème spécifique sont présentés dans l'article synthèse (*survey*) de Erdelić et Carić (2019). Parmi les autres approches, mentionnons celle de Yang **et al.** (2015) qui ajoute comme contrainte la considération du coût de recharge dans la fonction objectif. Ils supposent que le prix varie en fonction de la borne et de l'heure de la journée (par exemple pour inciter les conducteurs à ne pas se recharger pendant les heures de forte demande à la borne ou sur le réseau électrique). Un algorithme d'apprentissage parthéno-génétique (*learnable partheno-genetic algorithm*) est introduit (une variante d'algorithme génétique d'optimisation) puisque le problème est discret, non linéaire et non convexe, ce qui fait que les techniques classiques ne sont pas adaptées. Mentionnons également la technique de Montoya **et al.** (2017) qui étend le concept d'état de charge (SoC) au problème du voyageur de commerce électrique à plusieurs véhicules et celle de Lin **et al.** (2016), qui est la première recherche à prendre en compte l'effet de la masse de la cargaison et des passagers sur la consommation énergétique.

3.2 Energy-A*

La possibilité qu'ont les VÉ de se recharger grâce à des baisses d'altitude et à des freinements fait en sorte que les méthodes classiques de détermination du plus court chemin ne sont plus suffisantes. En effet, sur une route inclinée où le VÉ descend, la récupération d'énergie pourrait être plus grande que l'énergie dépensée pour faire le parcours. Par conséquent, lorsqu'on modélise les contraintes énergétiques en ajoutant au graphe des arcs donnant le coût énergétique de parcours,

il peut y avoir des arcs négatifs. Malheureusement, l'algorithme de Dijkstra et l'algorithme A* nécessitent l'hypothèse que les coûts associés aux arcs du graphe soient tous positifs. Des modifications non triviales sont donc requises.

L'algorithme Energy-A* (Sachenbacher **et al.**, 2011) permet de considérer la récupération énergétique et de modéliser les contraintes énergétiques spécifiques aux VÉ sus-mentionnées. Une technique similaire est utilisée par toutes les autres approches qui considèrent également la récupération énergétique. Energy-A* est donc un algorithme fondamental pour la planification pour VÉ (dans cette section, on ne considère pas la présence de bornes de recharge).

D'abord, appelons $z\colon V \to \mathbb{R}^+$ la fonction donnant l'altitude à chaque sommet du graphe (qui est supposée connue). La définition 3.1 présente l'étiquetage des arcs utilisé par Energy-A* qui est basé sur l'énergie plutôt que sur le temps ou la distance comme c'était le cas au chapitre 2.

Définition 3.1. Le **poids** de chaque arc $e = (a, b) \in E$ est défini par

$$C(e) = C_L(e) + C_P(e)$$

où

- $C_L(e) = E_L(\lambda(e), \sigma(e), \dots)$ modélise la perte énergétique due à l'environnement, où C_L est linéairement croissante par rapport à λ, et est monotone croissante par rapport à σ. La fonction E_L peut optionnellement prendre plus de paramètres, par exemple pour modéliser l'impact de la température sur la consommation énergétique.
- $C_P(e) = \pi(b) - \pi(a)$ (où $\pi(u) = mgz(u)$) donne la différence d'énergie potentielle gravitationnelle entre les deux nœuds joints par l'arc (où m est la masse du VÉ, et $g \approx 9,8$ N/kg est l'accélération gravitationnelle terrestre).

Remarque. Puisque $C_P(e) < 0$ lors d'une baisse d'altitude, $C(e)$ peut être négatif

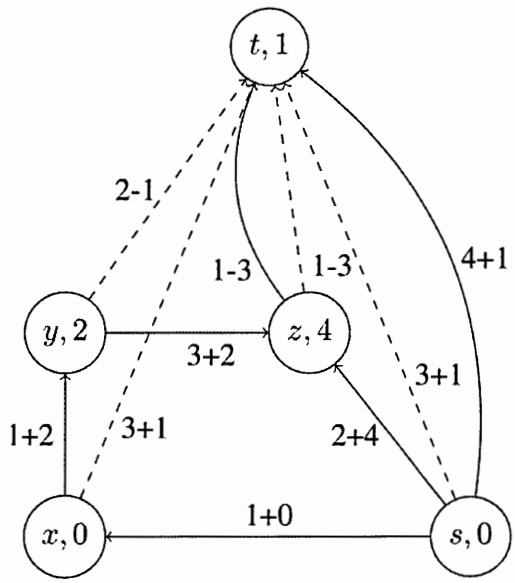


FIGURE 3.1: Graphe présentant la formulation énergétique des arcs

pour certains arcs. Par contre, il ne peut jamais y avoir de cycle négatif (à cause de la première loi de la thermodynamique, sur la conservation de l'énergie).

La figure 3.1 présente un graphe où on peut voir pour chaque nœud n l'altitude $z(n)$, et pour chaque arc e les deux valeurs $C_L(e)$ et $C_P(e)$.

Définition 3.2. Soient $P = \langle v_1, v_2, \ldots, v_k \rangle$ un chemin dans le graphe, $P_i = \langle v_1, \ldots, v_i \rangle$ le chemin préfixe passant par $i \leq k$ nœuds et $J \leq \rho$, la charge initiale de la batterie. Alors le **coût du chemin** P_k est donné par la fonction

$$c(P_k) = \begin{cases} \rho - J & \text{si } k = 1 \\ 0 & \text{si } k > 1 \wedge \Delta_k < 0 \\ \Delta_k & \text{si } k > 1 \wedge \Delta_k \in [0, \rho] \\ \infty & \text{si } k > 1 \wedge \Delta_k > \rho \end{cases}$$

où $\Delta_k = c(P_{k-1}) + C(v_{k-1}, v_k)$. Le premier cas permet de tenir compte de la charge

initiale (si le VÉ a une charge initiale de J, cela correspond à un *coût virtuel* initial de $\rho - J$ par rapport à une charge pleine). Le deuxième cas indique simplement qu'on peut *ignorer* le coût de tout préfixe ayant un coût net négatif. Le dernier cas indique que si le coût dépasse l'autonomie, le chemin est impossible (donc il a un coût infini). Dans tous les autres cas, le coût est simplement la somme des coûts de chaque arc traversé.

Maintenant que la formulation énergétique du problème de planification pour VÉ est présentée, il reste à voir comment trouver le chemin optimal. Puisque le graphe peut contenir des arcs négatifs, `Dijkstra` (de complexité $\mathcal{O}(n \log n + m)$) ne peut pas être utilisé directement. L'algorithme normalement utilisé lorsqu'il y a des arcs négatifs, l'algorithme de Bellman-Ford, a malheureusement une complexité de $\mathcal{O}(nm)$, soit beaucoup plus grande que celle de `Dijkstra`.

Il est connu que l'ajout d'un **potentiel** donné par une fonction de potentiel Π à l'énergie potentielle de chaque nœud ne modifie pas le chemin optimal retourné par `Dijkstra`, où Π est une fonction qui respecte la propriété

$$\Pi(v) - \Pi(u) \leq c(u, v) \quad \forall (u, v) \in E$$

(voir par exemple la section 10.6 du livre de Mehlhorn et Sanders (2008)).

Il suffit donc de trouver une telle fonction, ce qui permettra de se ramener à des arcs positifs et d'utiliser l'algorithme de Dijkstra. Normalement, l'algorithme de Johnson est utilisé pour trouver la fonction de potentiel. Celui-ci consiste simplement à exécuter l'algorithme de Bellman-Ford pour trouver une telle fonction, à modifier les arcs en conséquence, puis à lancer `Dijkstra` sur le graphe modifié. Cependant, dans notre cas, la fonction $\pi(u) = mgz(u)$ est déjà une fonction de potentiel, donc on n'a pas à en trouver une. Avec ce choix, chaque arc est maintenant

positif. En effet :

$$\begin{aligned} c_\pi(u,v) &= C_L(u,v) + C_P(u,v) + \pi(u) - \pi(v) \\ &= C_L(u,v) + (\pi(v) - \pi(u)) + \pi(u) - \pi(v) \\ &= C_L(u,v) \geq 0 \end{aligned}$$

L'algorithme de Dijkstra est donc maintenant utilisable pour trouver une solution. Comme pour le planificateur de base présenté à l'algorithme 2.1, on peut utiliser A* plutôt que `Dijkstra`. L'heuristique utilisée par Energy-A* est donnée par

$$h_L(u,v) = E_L(\ell = \|(u,v)\|, s = s_{\min})$$

où $\|(u,v)\|$ est la distance euclidienne L_2, tandis que $s_{\min}$ est la vitesse minimale possible sur le segment de route (u,v).

Cette heuristique est consistante, car

$$\begin{aligned} h_L(u,v) &\leq h_L(u,t) + h_L(t,v) \quad (E_L \text{ est lin. croiss. en fct. de } \ell) \\ &\leq C_L(u,t) + h_L(t,v) \quad (E_L \text{ est croiss. en fct. de } s \text{ et } \ell) \end{aligned}$$

Il reste un dernier problème à régler. Bien que l'algorithme retourne maintenant des chemins de coût total respectant l'autonomie, certains chemins retournés peuvent être impossibles, car la contrainte de batterie n'est pas vérifiée pour chaque préfixe du chemin. Par exemple, le chemin $\langle s, z, t \rangle$ à la figure 3.1 a un coût total de $(2+4)+(1-3) = 4$, ce qui respecte une autonomie de $\rho = 4$. Par contre, l'arc (s,z) n'est pas traversable, car son coût de $2+4$ dépasse l'autonomie. Pour

pallier ce problème, il faut ajouter une nouvelle fonction de coût, qui tient compte dynamiquement des contraintes énergétiques.

Définition 3.3. La fonction de **coût dynamique** est donnée par

$$\tilde{c}\colon V \times V \times \mathbb{R} \to \mathbb{R}$$

$$(u,v,k) \mapsto \begin{cases} -\Delta(u,v,k) & \text{si } \Delta(u,v,k) < 0 \\ 0 & \text{si } \Delta(u,v,k) \in [0,\rho] \\ \infty & \text{si } \Delta(u,v,k) > \rho \end{cases}$$

où $\Delta(u,v.k) = k + c(u,v)$ et k est l'autonomie actuelle restante. Le premier cas est celui qui permet à l'algorithme de considérer une recharge de la batterie lors d'une descente.

En résumé, l'algorithme Energy-A* est donc une modification de A* qui parcourt des arcs représentant un coût ou un gain énergétique et qui utilise $c + \tilde{c}$ comme fonction de coût et h_L comme heuristique. Les deux avantages par rapport au planificateur de base sont la possibilité d'avoir une batterie initialement partiellement chargée, et la considération de la recharge de la batterie par récupération d'énergie due à une baisse d'énergie potentielle. Par contre, Energy-A* ne considère pas la possibilité de se recharger à des bornes. La complexité théorique est $\mathcal{O}(n \log n + m)$ (comme `Dijkstra`), mais A* explore en pratique beaucoup moins de nœuds que `Dijkstra`.

3.3 Contraction hiérarchique de graphe

Cette section présente la méthode assez récente de la contraction de graphe, proposée initialement par Geisberger **et al.** (2008) et améliorée subséquemment (Geisberger **et al.**, 2012; Delling et Werneck, 2013). Celle-ci a révolutionné le domaine

de planification d'itinéraires en général, et d'itinéraires pour VÉ en particulier, car elle permet de diminuer significativement le nombre d'arcs qui doivent être traversés par `Dijkstra` avant de trouver un chemin optimal entre deux points, ce qui accélère le calcul en conséquence. Dans un graphe comme celui du Québec, le nombre de nœuds dépasse 5 millions. Il suffit d'imaginer la taille du graphe de l'Amérique du Nord pour voir qu'une méthode d'accélération comme la contraction de graphe est essentielle pour obtenir une solution dans un temps raisonnable.

Cette technique se base sur une idée assez intuitive : la majorité des chemins dans un réseau routier représentent des petites routes secondaires. Or, la grande majorité des chemins vont passer par des autoroutes et ce n'est que près du départ et de l'arrivée que le véhicule risque de devoir prendre des plus petites routes. L'intuition est donc de modifier le graphe en ajoutant des *raccourcis* (de nouveaux arcs) dans le graphe. Par exemple, on pourrait ajouter un arc allant directement de Montréal jusqu'à Québec en fusionnant tous les arcs de l'autoroute 20 entre Montréal et Québec.

Plus formellement, on crée des raccourcis en contractant un nœud. Les définitions 3.4 et 3.5 présentent le concept de contraction.

Définition 3.4. Soit un nœud N ayant des arêtes entrantes partant des nœuds $\{u_1, u_2, \ldots, u_k\}$ et ayant des arêtes sortantes allant aux nœuds $\{v_1, v_2, \ldots, v_p\}$. La **contraction du nœud** N se fait en :

- ajoutant une arête entre chaque u_i et v_j tels que $< u_i, N, v_j >$ est l'unique plus court chemin allant de u_i à v_j (sur laquelle on note le nœud N ayant amené à la création du raccourci) ;
- retirant N et ses arêtes incidentes.

Définition 3.5. Soit $S = (s_1, s_2, \ldots, s_n)$ un ordonnancement des nœuds du graphe. La **contraction de graphe** consiste à contracter chacun des nœuds du

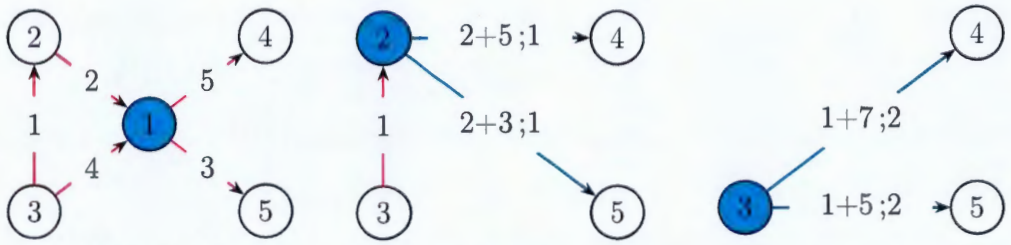


FIGURE 3.2: Exemple de contraction d'un graphe

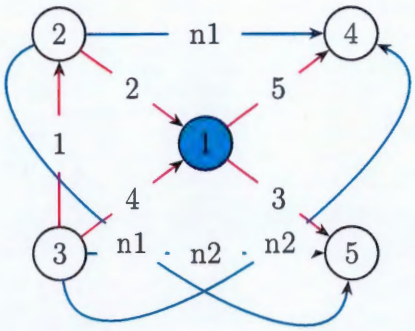


FIGURE 3.3: Graphe G^*

graphe dans l'ordre donné par S. Le graphe G auquel on ajoute les raccourcis créés par les contractions est appelé G^* et est le résultat final de la contraction.

La figure 3.2 montre un exemple des différentes étapes de la contraction d'un graphe. Même si en théorie, chacune des figures montre un graphe différent, ce qui donne au final une suite de plusieurs graphes $G = G_0, G_1, \ldots, G_n$, on peut en pratique tout faire sur le même graphe, ce qui donne G^* comme résultat après contraction de chacun des nœuds (qu'on peut voir à la figure 3.3).

L'algorithme 3.1 donne un aperçu des différentes étapes de la technique de contraction hiérarchique d'un graphe. D'abord, il faut choisir un ordonnancement des nœuds. Si les nœuds étaient numérotés de 1 à n, alors choisir un ordonnancement consiste à choisir une permutation des nœuds. Bien que la contraction est assu-

rée d'être valide pour n'importe quel ordonnancement, le nombre de raccourcis ajoutés peut varier et affecter la performance. Intuitivement, puisqu'en contractant un nœud, on enlève les arêtes incidentes et on ajoute des raccourcis, on veut commencer par contracter les nœuds qui ont le plus d'arêtes incidentes, mais qui vont créer le moins de raccourcis. Dans le pire cas, en contractant un nœud ayant i arcs entrants et j arcs sortants, on peut avoir jusqu'à ij raccourcis créés, ce qui est le même nombre d'arêtes que la contraction enlève, ce qui n'amène aucun gain. La définition 3.6 présente une heuristique permettant de déterminer un choix d'ordonnancement des nœuds donnant de bons résultats empiriques.

Algorithme 3.1 Contraction hiérarchique d'un graphe

fonction CONTRACTIONGRAPHE($G = (V, E, c)$)
  Choisir $\sigma \in S_n$ (S_n est l'ensemble des permutations de n éléments)
  $A \leftarrow \sigma(V)$
  pour tout $a \in A$ **faire**
    ContracterNoeud(G, a)
  fin pour
fin fonction

fonction CONTRACTERNOEUD($G = (V, E, c), a$)
  $V_e(a) \leftarrow \{u_1, \ldots, u_k\}$, les prédécesseurs de a
  $V_s(a) \leftarrow \{v_1, \ldots, v_p\}$, les successeurs de a
  pour tout $u \in V_e(a)$ **faire**
    Soit $G' \subset G$, le sous-graphe induit ne contenant pas a
    $\ell \leftarrow c(u, a) + \max_{v \in V_s(a)} c(a, v)$
    $d \leftarrow$ `Dijkstra`(u, G') ▷ Arrêt lorsque coût dépasse ℓ
    pour tout $v \in V_s(a)$ **faire**
      si $d(u, v) > c(u, a) + c(a, v)$ **alors**
        $E \leftarrow E \cup \{(u, v)\}$ ▷ Ajout raccourci
      fin si
      $E \leftarrow E \setminus \{(a, v)\}$ ▷ Retrait arête sortante
    fin pour
    $E \leftarrow E \setminus \{(u, a)\}$ ▷ Retrait arête entrante
  fin pour
fin fonction

Définition 3.6. Soit $\tau(u) = R(u) - i(u)$, la **différence d'arête** de u, où

— $R(u)$ est le nombre de raccourcis qui seraient créés si u était contracté ;

— $i(u)$ est le nombre d'arêtes incidentes à u.

Soit T, la suite ordonnée des u telle que $\forall i \in \{1, 2, \ldots, n\}, \tau(T[i]) \leq \tau(T[i+1])$. Alors la permutation

$$\sigma = \begin{pmatrix} T[1] & T[2] & \cdots & T[n] \\ 1 & 2 & \cdots & n \end{pmatrix}.$$

donne un ordonnancement donnant de bons résultats empiriques.

Remarque. Puisque $\tau(u)$ va changer au fur et à mesure que d'autres nœuds sont contractés, des techniques de mise à jour dynamique existent pour empêcher d'avoir à recalculer $\tau(u)$ pour tous les nœuds après chaque contraction.

Ensuite, il suffit de contracter chaque nœud dans l'ordre pour trouver le graphe G^*. Pour faire la contraction de nœud telle que décrite à la définition 3.4, la difficulté est de déterminer pour quelle paire (voisin entrant, voisin sortant) il faut ajouter un raccourci. L'algorithme 3.1 présente une façon simple permettant de prendre cette décision (en faisant des appels à `Dijkstra` partant des nœuds entrants du nœud à contracter).

Une fois le graphe $G = (V, E, c)$ contracté et le graphe $G^* = (V, E^*, c')$ trouvé, une version modifiée de l'algorithme de Dijkstra bidirectionnel est utilisée pour trouver le plus court chemin entre deux nœuds. L'algorithme 3.2 en présente les grandes lignes. *Grosso modo*, un `Dijkstra` partant du départ et ne parcourant que des arêtes allant vers des nœuds plus grands (par rapport à l'ordonnancement) et un `Dijkstra` partant de l'arrivée et ne parcourant en sens inverse que des arêtes allant vers des nœuds plus petits sont faits. Nécessairement, ces appels à `Dijkstra` se croisent en un ensemble de points, sur lequel on prend le minimum de la somme des distances (ligne 7 de l'algorithme 3.2).

Une autre *heuristique* peut être proposée pour choisir l'ordonnancement des nœuds.

Algorithme 3.2 Requête dans un graphe contracté

1: **fonction** REQUETECONTRACTION($G^* = (V, E^*, c'), s, t$)
2: $G^*_{\uparrow} \leftarrow (V, \{(u, v) \in E^* : v > u\})$
3: $G^*_{\downarrow} \leftarrow (V, \{(u, v) \in E^* : v < u\})$
4: $d_s \leftarrow$ `Dijkstra`$(G^*_{\uparrow}, s)$
5: $d_t \leftarrow$ `Dijkstra`$(\overline{G}^*_{\downarrow}, t)$ ▷ $\overline{G}$ est le graphe transposé de G
6: $I \leftarrow \{u \in V \mid d_s(u) \neq \infty \wedge d_t(v) \neq \infty\}$ ▷ Nœuds explorés par les 2 dijk.
7: $d(s, t) = \min_{v \in I}(d_s(v) + d_t(v))$
8: **fin fonction**

Elle est basée sur le fait qu'il n'est pas intéressant d'avoir plusieurs nœuds rapprochés qui sont contractés les uns à la suite des autres, car alors le temps d'exécution des deux appels à `Dijkstra` ne sera pas équilibré et on perdra un peu en rapidité. Pour pallier ce problème, on peut garder un compteur dans chaque nœud qui indique le nombre de voisins qui ont été contractés pour l'instant. Ce compteur peut être combiné à la fonction τ de la définition 3.6 pour permettre une contraction des nœuds qui est un peu plus uniforme dans le graphe.

La technique de contraction de graphe permet une accélération substantielle de la recherche de plus courts chemins dans un graphe. Pour donner une idée de l'amélioration, les résultats dans l'article de Geisberger **et al.** (2012) sont les suivants : la moyenne de temps (sur 100 000 requêtes) était de 5591 ms par requête pour un `Dijkstra` unidirectionnel standard, tandis qu'elle était de 0.21 ms par requête avec la contraction hiérarchique (nécessitant un prétraitement de 8 minutes pour construire le graphe contracté).

CHAPITRE IV

REGROUPEMENT DE BORNES

Truth is ever to be found in the simplicity, and not in the multiplicity and confusion of things.

— Isaac Newton

Comme vu aux chapitres précédents, la planification pour VÉ est un problème complexe nécessitant l'utilisation de diverses méthodes pour la prise en compte de plusieurs facteurs. Par exemple, la considération de la recharge de la batterie par récupération énergétique due à un freinage ou à une baisse d'altitude a été résolue par l'algorithme Energy-A*. De même, une réduction du temps de calcul peut être réalisée grâce à la contraction hiérarchique de graphe.

Dans ce chapitre, nous présentons notre première contribution originale dans le domaine, qui a mené à un article et une présentation dans le cadre de la $XXV^{ième}$ *rencontre de la Société Francophone de Classification* (Champagne Gareau **et al.**, 2018) ainsi qu'à un autre article et une autre présentation à la conférence de la *International Federation of Classification Societies* (Champagne Gareau **et al.**, 2019). Cette contribution est une méthode permettant l'accélération des calculs, tout comme la contraction hiérarchique, mais se basant plutôt sur des regroupements de bornes. En examinant sur une carte le positionnement des bornes actuellement en service, on peut voir qu'elles ne sont pas dispersées uniformément. En effet, elles sont pour la plupart situées aux alentours des grandes villes

et des grandes autoroutes. On pourrait donc s'attendre à pouvoir regrouper les bornes qui sont près les unes des autres (donc celles qui ne sont pas isolées) pour diminuer le nombre de bornes à considérer, et ainsi accélérer les calculs dans le s-graphe. La figure 4.1 présente le résultat d'un script qui génère une carte et affiche les bornes sur celle-ci en coloriant celles qui sont rapprochées (ex. les bornes des Laurentides ont été mises en bleu par le script, les bornes de Lanaudière en turquoise et les bornes de la région de la Capitale-Nationale en violet).

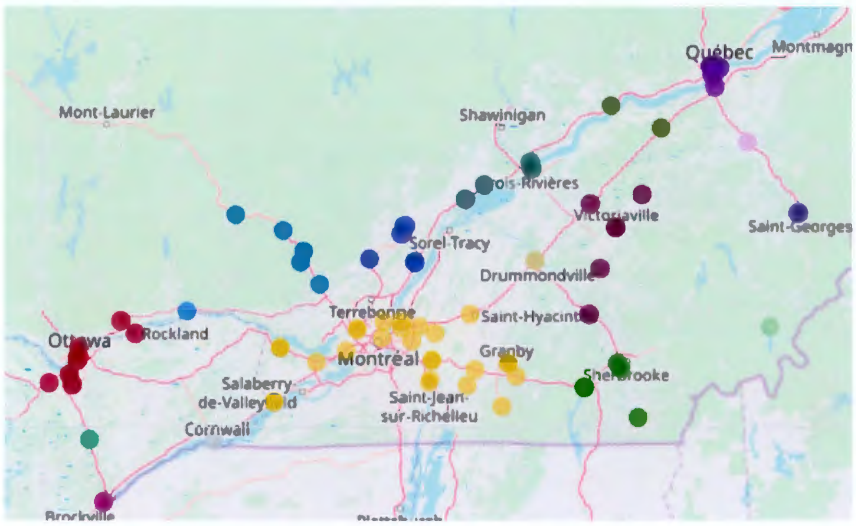


FIGURE 4.1: Bornes regroupées

4.1 Algorithme de regroupement

Dans cette section, l'algorithme qui crée les regroupements est introduit et on montre comment modifier le planificateur de base pour tenir compte de ceux-ci. La figure 4.2 présente une idée du graphe qu'on cherche à obtenir.

Pour pouvoir créer des regroupements de différentes tailles, un paramètre $d_{\max} \in \mathbb{R}^+$ correspondant à la distance maximale entre le centre d'un regroupement et une borne à l'intérieur de celui-ci est nécessaire. L'algorithme 4.1 présente la méthode

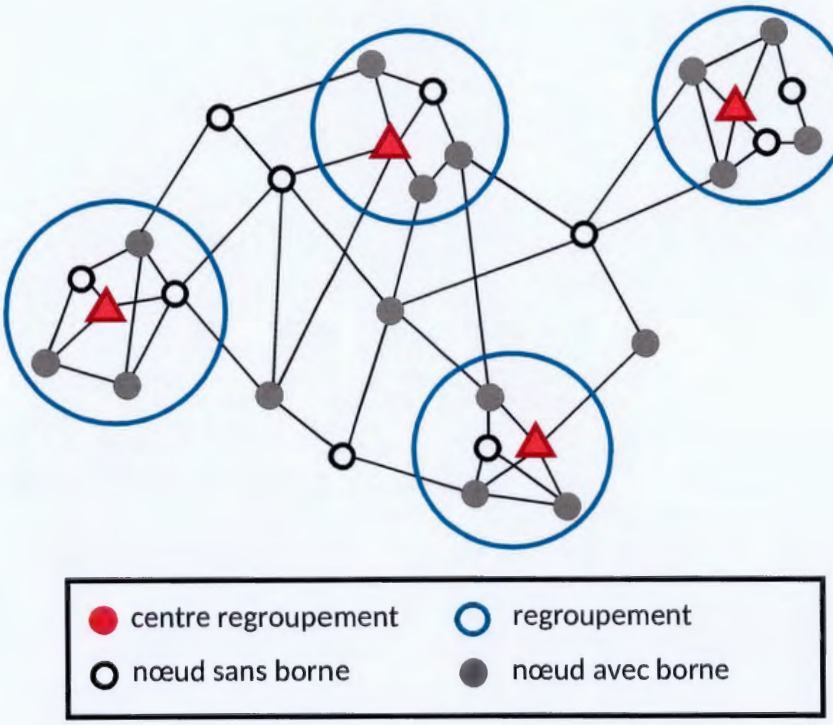


FIGURE 4.2: Graphe après construction des regroupements

de regroupement. Dans celui-ci, trouver les bornes les plus rapprochées est simple : il suffit de trouver le plus petit nombre dans la matrice de distance D. Ensuite, pour trouver le nœud à mi-chemin, on peut faire une exécution de `Dijkstra` et parcourir les nœuds le long du chemin optimal jusqu'à ce qu'on dépasse la moitié de la distance. On enlève ensuite de l'ensemble des bornes celles qui sont dans le regroupement, et on ajoute le centre comme nouvelle borne. En assignant ensuite des 0 aux colonnes et lignes qui correspondent aux bornes faisant partie du nouveau regroupement, celles-ci seront ignorés pour la prochaine itération.

Pour ce qui est de la complexité temporelle, trouver les bornes les plus rapprochées se trouve dans $\mathcal{O}(|S|^2)$. Les autres opérations sont négligeables, donc la complexité totale de la boucle est $\mathcal{O}(K|S|^2)$ où K est le nombre de regroupements créés (le nombre d'itérations de la boucle). K est normalement plus petit que S (sinon les

Algorithme 4.1 Construction des regroupements

Trouver à l'aide de la matrice D les bornes $a, b \in S$ les plus rapprochées [1]
tant que $\texttt{dist}(a, b) \leq d_{\max}$ **faire**
    Trouver le nœud $m \in V$ à mi-chemin entre a et b
    $d_m \leftarrow \texttt{Dijkstra}_{d_{\max}}(m)$
    $C \leftarrow \{s \in S \mid d_m(s) \leq d_{\max}\}$
    $S \leftarrow (S \setminus C) \cup \{m\}$
    Mettre 0 aux lignes et colonnes de D correspondant aux bornes dans C
    Trouver à l'aide de D les deux bornes $a, b \in S$ les plus rapprochées
fin tant que
Recalculer D avec les bornes représentant des regroupements

regroupements seraient petits, voire n'auraient qu'une borne), mais asymptotiquement, on ne peut pas dire mieux que $K \in \mathcal{O}(|S|)$. Finalement, le recalcul de D nécessite de lancer K fois `Dijkstra`, ce qui donne une complexité de $\mathcal{O}(K(n \log n))$. Somme toute, la complexité globale est donc de $\mathcal{O}(K(|S|^2 + n \log n))$.

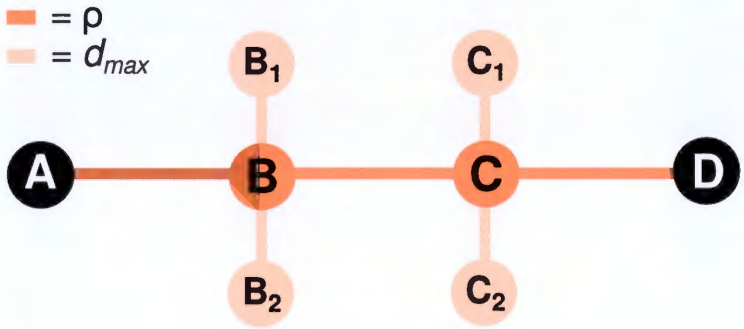


FIGURE 4.3: Illustration du problème de l'autonomie avec regroupement

Le planificateur ne peut malheureusement pas directement fonctionner avec les regroupements. En effet, sans changement, le planificateur calculera des chemins en considérant la distance allant d'un centre de regroupement à un autre, alors qu'en pratique, le VÉ doit se rendre à une borne à partir du centre, ce qui augmente l'autonomie nécessaire par rapport à ce que le planificateur considère. Si

1. On rappelle (du chapitre 2) que S est l'ensemble des bornes et que D est la matrice des distances entre chaque paire de bornes.

l'on ne veut pas obtenir d'itinéraires non réalisables, l'autonomie considérée par le planificateur doit donc être modifiée en conséquence. La figure 4.3 illustre le problème lorsqu'on ne modifie pas le planificateur. Dans cet exemple, la distance entre les regroupements B et C est ρ, l'autonomie du véhicule. Or, pour atteindre une des stations du regroupement D à partir d'une station du regroupement C, il faut parcourir une distance de $\rho + 2d_{\max}$. Le chemin retourné par le planificateur, $\langle A, B, C, D \rangle$, n'est donc pas possible dans la réalité. La proposition 4.1 présente la modification nécessaire pour que tous les chemins soient valides.

Proposition 4.1. *Soit ρ l'autonomie réelle du VÉ et ρ' l'autonomie considérée par le planificateur. Lorsqu'on crée des regroupements de taille $\leq d_{\max}$, alors ρ' doit être égal à $\rho - d_{\max}$ entre le départ et le premier regroupement, et entre le dernier regroupement et l'arrivée, et doit être égal à $\rho - 2d_{\max}$ entre deux regroupements.*

Preuve. Soient $\{C_1, \ldots, C_k\}$, l'ensemble des regroupements traversés dans le chemin de α à ω retourné par le planificateur et $\{c_1, \ldots, c_k\}$, l'ensemble contenant le nœud central de chacun de ceux-ci. Le chemin de α à c_1 a une distance d'au plus ρ'. Puisque c_1 est le centre de C_1 (pas nécessairement une borne) et que les bornes de C_1 sont à une distance d'au plus $d_{\max}$ de c_1, alors le VÉ doit avoir une autonomie de $\rho = \rho' + d_{\max}$, et donc $\rho' = \rho - d_{\max}$. Le même argument permet de montrer le même résultat pour le segment du plan allant du dernier regroupement à l'arrivée. Pour ce qui est du chemin entre $s_i \in C_i$ et $s_{i+1} \in C_{i+1}$:

$$\begin{aligned} \texttt{dist}(s_i, s_{i+1}) &\leq \texttt{dist}(s_i, c_i) + \texttt{dist}(c_i, c_{i+1}) + \texttt{dist}(c_{i+1}, s_{i+1}) && \text{inégalité du } \Delta \\ &\leq d_{\max} + \rho' + d_{\max} && \text{par définition.} \end{aligned}$$

Puisque dans certains cas (ex. : figure 4.3) l'inégalité précédente est une égalité, alors le majorant est un suprémum et on doit donc avoir $\rho' = \rho - 2d_{\max}$. □

La proposition 4.1 présente le pire cas théorique. Par contre, dans l'implémentation de l'algorithme de planification, on peut considérer un rayon r_i pour chaque regroupement c_i (correspondant à la distance maximale entre le centre du regroupement et une borne en faisant partie). L'autonomie considérée entre deux regroupements (c_i, c_j) peut donc être de $\rho' = \rho - r_i - r_j \geq \rho - 2d_{\max}$.

Le fait que la génération de regroupements puisse produire des chemins impossibles (un cas extrême est lorsque $d_{\max} = \rho$; l'autonomie considérée est alors de 0) peut sembler inacceptable. Une possibilité pour surmonter ce désagrément est la suivante : on calcule d'abord le trajet avec les regroupements. Si le planificateur ne trouve pas de chemin possible dû à la contrainte provenant du $d_{\max}$, on recalcule un trajet, mais sans considérer les regroupements. Cette solution, qu'on appelle la *stratégie amortie*, n'élimine qu'une petite partie des gains de temps obtenus par les regroupements, tout en éliminant le problème des chemins impossibles.

Avant de présenter la méthodologie et les résultats des tests, un résumé du planificateur complet, incluant les regroupements et la stratégie amortie, est présenté à l'algorithme 4.2 qui est une amélioration de l'algorithme 2.1.

Le temps de construction du s-graphe (incluant l'ajout de α et de ω) n'est pas influencé par la méthode présentée, car les appels à `Dijkstra` sont faits sur le graphe original. Cette étape peut par contre être accélérée avec la contraction hiérarchique de graphe. Les lignes 12 à 14 ne sont pas modifiées non plus, mais ont un temps négligeable. Par conséquent, les tests évaluant le gain de performance des regroupements se concentrent sur le temps d'exécution des lignes 8 à 11.

Lorsqu'on utilise des regroupements (et donc que $d_{\max} > 0$), Q contient des centres de regroupements, et non plus de vraies bornes où on peut se recharger. Le choix de la borne à utiliser à l'intérieur du regroupement doit donc être fait séparément, et peut être fait pendant l'exécution du plan (la conduite du VÉ) plutôt qu'à

Algorithme 4.2 Planificateur incluant les regroupements de bornes

1: Générer les regroupements avec l'algorithme 4.1
2: Calculer la matrice D et le chemin le plus court reliant chaque paire de regroupements
3: Construire le s-graphe contenant chaque regroupement
4: **pour tout** requête (ρ, α, ω) **faire**
5: Exécuter `Dijkstra` partant de α sur le graphe original
6: Exécuter `Dijkstra` partant de ω sur le graphe transposé
7: Ajouter α et ω au s-graphe et ajouter les arcs de longueur $\leq \rho$
8: Exécuter l'algorithme A* de α à ω sur le s-graphe pour trouver la suite Q
9: **si** le chemin est impossible **alors** ▷ Stratégie amortie
10: Exécuter A* sur le s-graphe (sans regroupements) pour trouver Q
11: **fin si**
12: Trouver la suite P à l'aide de Q en utilisant les chemins déjà calculés
13: Retirer α et ω du s-graphe et réajuster les arcs
14: Retourner la solution (P, Q)
15: **fin pour**

l'étape de planification. Pour ce faire, on peut placer dans le graphe un *nœud de décision* de sorte qu'il soit atteint un certain temps avant d'arriver à un regroupement prévu. C'est en arrivant à celui-ci que le choix de la borne à utiliser dans le regroupement est fait. Cela amène plusieurs avantages. En effet, cela permet de considérer des données disponibles en temps réel (ex. : les conditions routières, l'état d'occupation des bornes, le trafic, etc.) pour choisir la borne la plus prometteuse à l'intérieur du prochain regroupement sur le chemin. Le chapitre 5 présente une façon de choisir le nœud de décision optimal lorsqu'on considère l'état d'occupation des bornes.

4.2 Évaluation

Le gain de performance amené par la technique de regroupement est mesuré en utilisant des données réelles du Québec. La carte (les nœuds et les segments de routes) provient du projet OpenStreetMap (Weber et Haklay, 2008). Le Québec

est un bon choix de territoire pour mesurer la technique, car il est vaste, le réseau de bornes de recharge est relativement bien développé et les itinéraires entre deux villes peuvent être très longs et nécessiter plusieurs recharges. Le graphe généré à partir de ces données a 2 923 013 sommets et 5 907 672 arêtes.

Les bornes de recharge considérées sont des bornes réelles faisant partie du réseau du Circuit électrique. L'ensemble de données contient 1162 bornes (incluant les bornes de niveau 1, 2 et 3). Elles ont toutes été considérées par le planificateur comme étant des bornes de niveau 3, car il n'y a pas assez de bornes réelles de niveau 3 actuellement disponibles (122 dans l'ensemble de données) pour permettre de tester adéquatement la méthode si l'on ne considère qu'elles.

Pour évaluer la méthode, 1000 requêtes consistant en un triplet (α, ω, ρ) ont été générées. Les nœuds de départ et d'arrivée ont été choisis uniformément parmi tous les nœuds du graphe. Ainsi, puisqu'il y a par exemple plus de nœuds à Montréal qu'à Chibougamau, les requêtes ont plus de chance de commencer ou de terminer à Montréal qu'à Chibougamau. C'est souhaitable, car ce sera le cas aussi pour les requêtes des utilisateurs. Pour sa part, l'autonomie a été générée uniformément entre 90 et 550 km, soit environ la limite minimale et maximale des VÉ actuellement en vente. Les solutions optimales des requêtes ont une grande variabilité, allant de 150 à 1000 km. De plus, chacune des requêtes générées nécessite au moins un arrêt à une borne de recharge pour que la destination soit atteignable.

La méthode a été évaluée en exécutant ces 1000 requêtes par le planificateur. Les tests ont été exécutés sur un Intel Core i5 7600k surcadencé à 5.1 GHz. Les résultats sont comparés en faisant varier deux paramètres : le nombre de bornes (en prenant un pourcentage de l'ensemble des données) et $d_{\max}$. Cela permet de mesurer l'effet du regroupement par rapport à la densité de bornes sur la carte et à la taille des regroupements.

Pour des raisons de simplicité, on suppose dans les tests que :

$$\sigma(e) = 90 \text{ km/h} \quad \text{pour tout } e \in E$$
$$\text{TRB}(s) = 30 \text{ min} \quad \text{pour tout } s \in S$$
$$\text{TA} \equiv 0$$

c'est-à-dire que la vitesse de la voiture est constante à 90 km/h, que le temps de recharge à chaque borne est constant (30 min) et qu'on néglige le temps d'attente aux bornes (c'est le sujet du chapitre 5). Ces trois facteurs n'ont pas d'impact sur ce qu'on souhaite montrer. De plus, on a déjà vu à la section 3.1 des techniques permettant de gérer des vitesses et des temps de recharge variables.

Le tableau 4.1 présente les résultats obtenus par les tests décrits ci-haut. Les colonnes représentent respectivement le nombre de bornes (B), le paramètre $d_{\max}$, le nombre de regroupements (R), le taux de rallonge du trajet (TR, augmentation de la durée des trajets) et finalement le taux d'échec (TE, pourcentage des requêtes qui étaient faisables sans regroupements et qui ne le sont plus) et le temps d'exécution, avec ou sans la stratégie amortie. $d_{\max} = 0$ implique qu'on ne fait pas de regroupements et donc que chaque borne est en soit son propre regroupement.

En considérant les colonnes $d_{\max}$ et R, on observe que l'augmentation de la taille des regroupements fait décroître leur nombre (car des bornes qui étaient isolées sont incluses dans un plus grand regroupement). Par exemple, lorsqu'on considère toutes les bornes (B = 1662) avec $d_{\max} = 2.5$, le nombre de regroupements passe de 1162 à 487, soit 41.9 % du nombre initial. Avec la plus grande valeur de $d_{\max}$ testée (40), on passe à 7.49 % du nombre de regroupements initial.

Comme le montre la colonne TE (version de base), le nombre de requêtes devenant impossibles avec les regroupements augmente assez vite lorsque le paramètre $d_{\max}$

TABLE 4.1: Résultats empiriques des regroupements

Paramètres		Regroupements		Version de base		Version amortie	
B	$d_{\max}$	R	TR	TE	CPU	TE	CPU
#	km	#	%	%	ms	%	ms
230	0,0	230	0,0	0,0	0,908	0	0,908
230	2,5	164	0,0	0,0	0,556	0	0,556
230	5,0	137	0,2	0,3	0,397	0	0,400
230	10,0	107	0,2	0,5	0,301	0	0,305
230	15,0	94	0,6	0,5	0,268	0	0,272
230	20,0	87	0,7	0,5	0,252	0	0,256
230	30,0	71	1,1	2,2	0,219	0	0,239
230	40,0	62	1,6	5,5	0,208	0	0,258
508	0,0	508	0,0	0,0	4,451	0	4,451
508	2,5	280	0,1	0,0	1,077	0	1,077
508	5,0	213	0,0	0,3	0,540	0	0,553
508	10,0	158	0,1	0,8	0,459	0	0,494
508	15,0	128	0,3	1,2	0,340	0	0,394
508	20,0	115	0,2	2,0	0,298	0	0,387
508	30,0	84	1,2	2,7	0,231	0	0,351
508	40,0	71	1,9	6,5	0,213	0	0,503
744	0,0	744	0,0	0,0	10,218	0	10,218
744	2,5	351	0,0	0,0	1,827	0	1,827
744	5,0	252	0,1	0,9	0,775	0	0,867
744	10,0	179	0,1	1,3	0,509	0	0,642

B : Bornes ; **R** : Regroupements ; **TR** : Taux de rallonge du trajet ;

TE : Taux d'échec ; **CPU** : Temps d'exécution

TABLE 4.1: Résultats empiriques des regroupements (suite)

Paramètres		Regroupements		Version de base		Version amortie	
B	$d_{\max}$	R	TR	TE	CPU	TE	CPU
#	km	#	%	%	ms	%	ms
744	15,0	147	0,2	1,7	0,391	0	0,565
744	20,0	126	0,1	2,5	0,319	0	0,574
744	30,0	91	1,0	2,8	0,242	0	0,528
744	40,0	74	2,2	8,1	0,217	0	1,044
970	0,0	970	0,0	0,0	17,246	0	17,246
970	2,5	428	0,0	0,0	2,701	0	2,701
970	5,0	307	0,1	0,4	1,258	0	1,327
970	10,0	215	0,2	0,7	0,501	0	0,621
970	15,0	165	0,6	0,8	0,434	0	0,572
970	20,0	138	0,9	1,4	0,341	0	0,582
970	30,0	100	2,7	1,8	0,249	0	0,560
970	40,0	83	3,0	8,1	0,221	0	1,618
1162	0,0	1162	0,0	0,0	26,563	0	26,563
1162	2,5	487	0,0	0,0	3,385	0	3,385
1162	5,0	342	0,2	0,4	1,541	0	1,647
1162	10,0	236	0,2	0,8	0,588	0	0,801
1162	15,0	188	0,6	0,9	0,523	0	0,762
1162	20,0	150	1,0	1,4	0,382	0	0,754
1162	30,0	111	2,3	2,0	0,265	0	0,796
1162	40,0	87	2,8	8,2	0,226	0	2,404

B : Bornes ; **R** : Regroupements ; **TR** : Taux de rallonge du trajet ;

TE : Taux d'échec ; **CPU** : Temps d'exécution

augmente, ce qui est inacceptable. En comparant les deux colonnes de la version de base avec celles de la version amortie, on voit que la version amortie élimine la perte de requêtes possibles, tout en préservant la majorité du gain de performance. La figure 4.4 montre la diminution du temps de calcul obtenue pour la version amortie en fonction du paramètre $d_{\max}$ pour les différentes densités de bornes testées. La colonne TR montre que la technique de regroupement de bornes peut augmenter légèrement la durée totale des itinéraires retournés par le planificateur. Le pourcentage d'augmentation est directement proportionnel à la valeur de $d_{\max}$.

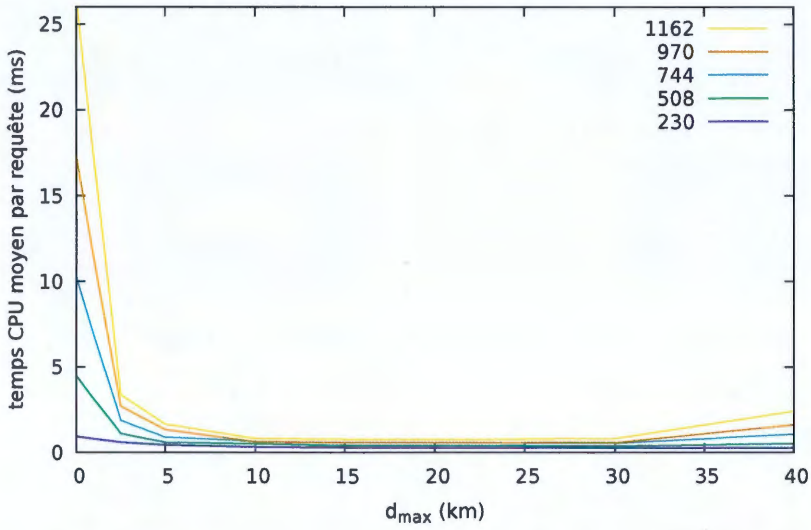


FIGURE 4.4: Temps CPU moyen pour différentes densités avec stratégie amortie

En considérant la figure 4.4 et le tableau 4.1, on constate deux choses. Premièrement, plus le nombre de stations est élevé, plus le temps de calcul est grand. Deuxièmement, plus le nombre de stations est élevé, plus la diminution du temps de calcul due à la création des regroupements est grande (en valeur, mais de façon plus surprenante, également en pourcentage). Cela s'explique par le fait que lorsqu'on ajoute de nouvelles bornes sur la carte, il y a plus d'opportunités de

regroupements. Ainsi, l'utilité de la méthode de regroupement ira en augmentant au fur et à mesure que le nombre de bornes disponibles sur la carte augmente.

Finalement, en considérant les colonnes TR et CPU (amortie), on peut voir que lorsqu'on se fixe une limite de 1 % de rallonge, la valeur $d_{\max}$ optimale semble être autour de 20 km. Une valeur plus grande implique ou bien un dépassement de la limite de 1 %, ou bien une augmentation du temps de calcul due au fait que le taux d'échec devient trop grand et donc que la stratégie amortie doit lancer trop de calculs pour éviter les chemins impossibles. Ce dernier phénomène peut être vu à la figure 4.4 où on voit que lorsque le paramètre $d_{\max}$ passe un certain seuil, le temps de calcul remonte et on perd tous les gains obtenus.

Somme toute, les résultats montrent que les regroupements ont réduit d'un facteur 8 le nombre de bornes considérées par le planificateur, et d'un facteur 35 le temps de calcul dans le s-graphe sans nécessiter de compromis sur la faisabilité des chemins retournés (avec la stratégie amortie) et en ayant moins de 1 % d'écart entre l'itinéraire optimal et celui retourné. Le gain de performance est proportionnel à la densité de bornes sur la carte, ce qui suggère que l'utilité de celle-ci ira en augmentant (car de plus en plus de bornes sont actuellement installées).

CHAPITRE V

TEMPS D'ATTENTE

On veut pouvoir faire une recharge d'appoint où que l'on soit et sans avoir à attendre qu'une borne se libère. Sans ces conditions, il est difficile de susciter une véritable percée du VÉ.

— Renaud Cloutier, délégué principal en développement des affaires, Hydro-Québec [1]

Ce chapitre présente notre seconde contribution au problème de planification pour VÉ. Celle-ci a mené à la publication d'un article qui a été présenté à l'*International Conference on Advances in Geographic Information Systems* (SIGSPATIAL 2019) de l'*Association for Computing Machinery* (ACM) en novembre 2019 à Chicago (Champagne Gareau **et al.**, 2019). Nous proposons de considérer le temps d'attente dans la fonction à minimiser dans le planificateur, permettant ainsi d'éviter les chemins passant par des bornes ayant une grande espérance d'attente. C'est un élément de plus en plus important à prendre en compte. En effet, dans la grande majorité des pays, le nombre de VÉ en circulation augmente plus rapidement que le nombre de bornes de recharge, ce qui amènera à une hausse du temps d'attente à court terme (Irle, 2018; Stubbe, 2018; Eckhouse **et al.**, 2019).

1. Bonneau (2019)

Le temps d'attente, bien qu'important à considérer, n'a commencé à être pris en compte que récemment. Kullman **et al.** (2017) se concentrent sur le problème de planification pour une flotte de VÉ qui doit livrer plusieurs colis dans un ordre optimal, en débutant et en terminant au même nœud (similaire au problème du voyageur de commerce, mais en considérant la présence de bornes de recharge et le temps d'attente). Ils appellent ce problème le EVRP-MRUA (*EV routing problem with mid-route recharging and uncertain availability*). Ils modélisent le problème en utilisant un **processus décisionnel de Markov** (*Markov decision process*, MDP) qui vise à minimiser le coût pour l'opérateur de la flotte de VÉ et le temps global de la livraison des colis.

Sweda **et al.** (2017) sont pour leur part les seuls autres chercheurs à avoir considéré le temps d'attente pour les VÉ. Ils se concentrent sur le même problème que celui traité dans ce chapitre, soit celui de déterminer le chemin optimal entre deux points tout en respectant l'autonomie du VÉ et en considérant le temps d'attente et l'incertitude sur la disponibilité des bornes. Les auteurs décrivent d'abord des algorithmes permettant de trouver une politique de rechargement optimale, c'est-à-dire une politique donnant le temps optimal que le VÉ devrait rester à chaque borne pour se recharger, étant donné un chemin fixé d'avance auquel le VÉ ne peut déroger. Ils étendent ensuite cet algorithme pour trouver le chemin optimal à priori, incluant la politique de recharge optimale (utilisant également des MDP). Le graphe considéré dans ce cas possède une borne de recharge à chaque nœud ; le départ et l'arrivée doivent donc être des bornes. La politique optimale (incluant le choix du chemin et la politique de recharge) est calculée en $\mathcal{O}(n^4)$, où n est le nombre de nœuds (donc de bornes) dans le graphe. Des heuristiques sont utilisées pour réduire la taille du problème. Dans le modèle utilisé, la disponibilité d'une borne de recharge est seulement connue lorsque le VÉ arrive à celle-ci. L'évaluation de la technique proposée est faite sur une grille

de 500 × 500 nœuds, avec une distance commune de 5 milles entre chaque paire de nœuds adjacents. Les probabilités d'occupation et les temps d'attentes sont générés aléatoirement en utilisant des distributions uniformes.

Les nouveautés amenées par les méthodes proposées dans ce chapitre permettent de combler certaines failles des méthodes mentionnées plus haut. Parmi les améliorations, notons :

— la construction et l'utilisation d'un modèle stochastique d'attente ;

— la prise en compte de la possibilité d'avoir accès aux données d'occupation en temps réel avant l'arrivée à une borne et de pouvoir ainsi changer dynamiquement l'itinéraire ;

— l'évaluation des algorithmes proposés avec des données réelles (carte, disposition des bornes, occupation) plutôt qu'avec des données artificielles (bornes disposées en grille, espacées uniformément, etc.).

Le chapitre est organisé comme suit. La section 5.1 montre comment utiliser les données réelles d'occupation pour développer un modèle stochastique à chaque borne donnant la probabilité d'occupation et, lorsqu'occupée, le temps d'attente espéré. Elle montre aussi comment modifier le planificateur de base pour tenir compte de ce modèle et considérer le temps *apriori* d'attente à chaque borne lors du calcul initial de l'itinéraire. La section 5.2 présente quant à elle une méthode de génération de chemins alternatifs qui peuvent être utilisés lorsque l'attente est pire qu'attendue à la borne prévue.

5.1 Modèle stochastique de l'attente aux bornes

Puisque la minimisation du temps d'attente se fait en allant à une borne qui nécessite peut-être un détour par rapport au plus court chemin géographique

(augmentant ainsi le temps de déplacement), il faut trouver un compromis optimal entre le temps d'attente et le temps de déplacement. Cela se fait assez simplement avec le formalisme défini au chapitre 2 une fois que la fonction TA qui avait été mentionnée est bien définie. Pour définir celle-ci, on a besoin d'informations sur la probabilité d'occupation et le temps d'attente espéré à chaque borne (qui dépendent du jour de la semaine, de l'heure, etc.). Ces données sont utilisées pour générer une famille de fonctions donnant ces informations pour chacune des bornes sur la carte.

Définition 5.1. La **probabilité à priori d'occupation** de chaque borne $s \in S$ est donnée par

$$f_s \colon \{Lundi, \ldots, Dimanche\} \times \{0..23\} \to [0,1]$$
$$(j,h) \mapsto \mathbb{P}(s \text{ est occupée} \mid \text{Jour} = j \wedge \text{Heure} = h)$$

et le **temps espéré d'attente** de la même borne est donnée par

$$g_s \colon \{Lundi, \ldots, Dimanche\} \times \{0..23\} \to \mathbb{R}^+.$$

Remarque. Le modèle peut être raffiné en considérant de plus petits intervalles de temps. Des intervalles d'une heure ont été choisis, car il n'y avait pas assez de données réelles à disposition pour pouvoir avoir de plus petits intervalles tout en gardant une représentativité statistique du modèle.

En utilisant le vocabulaire de la théorie des files d'attente (Gelenbe et Pujolle, 1982), les nombres $g_s(j,h)$ correspondent à des temps de service moyens (habituellement notés $\frac{1}{\mu}$) dans une file d'attente non homogène (les temps de service dépendent du jour et de l'heure).

Pour ce qui suit, on suppose que les familles de fonctions $\{f_s\}_{s \in S}$ et $\{g_s\}_{s \in S}$ sont

connus. La section sur l'évaluation de la technique à la fin du chapitre montre comment les fonctions ont été construites à l'aide des données réelles. On montre maintenant comment modifier le planificateur de base du chapitre 2 pour y inclure la considération de ces fonctions et ainsi avoir un planificateur qui considère le temps d'attente espéré dans le calcul de l'itinéraire optimal. La première étape est de définir un étiquetage dynamique ξ des arêtes qui étend l'étiquetage λ en y ajoutant un coût pour l'attente espérée (dépendant du temps). La définition 5.2 présente ce nouvel étiquetage.

Définition 5.2. Soit l'arête $e = (u, v) \in E$. L'étiquetage

$$\xi \colon E \times \{Lundi, \ldots, Dimanche\} \times \{0..23\} \to \mathbb{R}^+$$

est donné par

$$\xi(e, j, h) = \begin{cases} \lambda(e) & \text{si } u \notin S \\ \lambda(e) + f_u(j, h) \cdot g_u(j, h) \cdot \mu(e) & \text{si } u \in S. \end{cases}$$

ξ correspond donc à λ auquel on ajoute une distance *virtuelle* qui donne le coût correspondant au temps d'attente moyen lorsqu'on arrive à un moment spécifique.

Remarque. De façon équivalente, on aurait pu choisir d'ajouter un coût sur les nœuds et de modifier A* en conséquence plutôt que d'ajouter une valeur proportionnelle au temps d'attente à chaque arête sortante des bornes. Le choix a été fait d'inclure le coût sur les arêtes plutôt que sur les nœuds, car les arêtes dépendantes du temps semblaient plus fréquentes dans la littérature, et donc les algorithmes pour en tenir compte sont mieux compris. De surcroît, le formalisme utilisé pour les arêtes dépendantes du temps peut aussi servir à d'autres considérations reliées à la planification, comme le trafic ou la météo, tandis qu'un coût sur les nœuds ne peut pas servir à beaucoup d'autres choses que le temps d'attente.

Puisque nous avons maintenant un étiquetage dépendant du temps et que le temps (j, h) lors de l'arrivée à une borne $s \in S$ est inconnu avant de démarrer l'algorithme de recherche dans le graphe, le planificateur de base (algorithme 2.1) doit être modifié en adaptant l'appel à l'algorithme A* de la ligne 7. Un survol de la littérature (*survey*) en ce qui a trait aux modifications des principales méthodes de recherche de chemins dans un graphe permettant la considération des arêtes variant dans le temps est présenté par Delling et Wagner (2009).

L'appel à A* dans l'algorithme de base est modifié de la même manière que `Dijkstra` est modifié dans cet article. En résumé, lorsqu'un nœud $s \in S$ du s-graphe est extrait de la file prioritaire, le plus court chemin jusqu'à s a été trouvé, et donc le jour et l'heure estimés d'arrivée à s sont connus (le temps au départ, plus le temps pour se rendre jusqu'à s, incluant les temps d'attente et de recharge). Ainsi, l'étiquetage $\xi(e, j, h)$ de chaque arête sortante $e = (s, v)$ de s peut être déterminé lors de l'extraction de s de la file, et A* peut donc ajouter les successeurs de s dans la file avec les bons coûts, et continuer comme normalement.

5.2 Génération de chemins alternatifs

Comme présenté à la section 5.3, la modification du planificateur proposée à la section précédente permet de diminuer énormément le temps moyen d'attente (et le temps total). Par contre, la technique n'est basée que sur les probabilités d'occupation et les temps d'attentes disponibles à priori. Si, par malchance, l'occupation réelle est pire que prévue pendant que le conducteur de VÉ effectue le plan, il serait avantageux d'avoir un planificateur en ligne (*online planner*) capable de prendre en compte cette information en temps réel et d'adapter le chemin proposé au fur et à mesure de l'évolution de l'état des bornes. Dans cette section, on propose d'ajouter cette fonctionnalité au planificateur en précalculant, pour

chaque borne de la suite Q (les bornes faisant partie de la solution optimale à priori), un chemin alternatif passant par une (ou des) borne(s) de recharge alternative(s). Ces chemins peuvent ensuite être choisis par le planificateur pendant l'exécution du plan (*runtime*) lorsqu'on arrive à un nœud spécial (qu'on appelle un nœud de décision) et que la prochaine borne prévue par la solution à priori est occupée. Le nœud de décision est choisi comme étant le dernier nœud commun entre le chemin de la solution à priori (qu'on appelle le chemin de base), et le chemin alternatif.

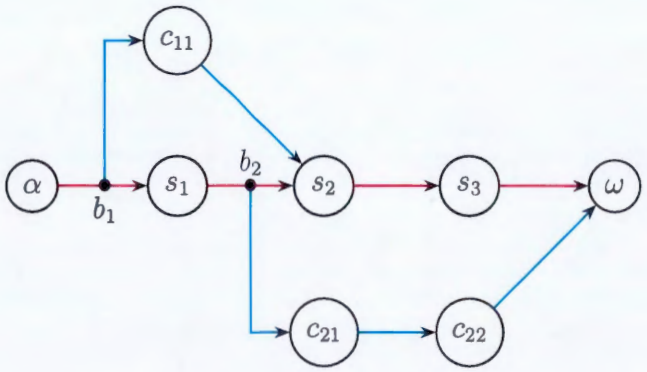


FIGURE 5.1: Génération de chemins alternatifs

La situation est illustrée graphiquement à la figure 5.1. Dans cette figure, le chemin milieu $Q = (\alpha, s_1, s_2, s_3, \omega)$ est le chemin obtenu par la méthode considérant l'occupation à priori de la section précédente, tandis que le chemin du haut (respectivement du bas) est un chemin alternatif qui peut être utilisé si s_1 (respectivement s_2) est occupé lorsque le VÉ arrive à b_1 (respectivement b_2). Le nœud c_{ij} représente la $j^{\text{ième}}$ borne de recharge sur le $i^{\text{ième}}$ chemin alternatif (le chemin utilisé lorsque la borne s_i est occupée) et le nœud b_i est le nœud de décision où on peut choisir de prendre le $i^{\text{ième}}$ chemin alternatif.

Pour trouver des chemins alternatifs pour chaque borne $s \in Q$ tels que décrits ci-haut, on exécute l'algorithme 5.1. Comme résultat, on obtient une application

$\pi\colon V \to V^2$ définie par

$$\pi(x) = \begin{cases} (s_{i+1}, -) & \text{si } x = s_i \wedge \neg\texttt{Alt}(i+1) \\ (b_i, -) & \text{si } x = s_i \wedge \texttt{Alt}(i+1) \\ (s_i, c_{i1}) & \text{si } x = b_i \\ (c_{i,j+1}, -) & \text{si } x = c_{ij}, \end{cases}$$

où $\texttt{Alt}(i)$ est un prédicat qui est vrai si et seulement s'il existe un chemin alternatif pour la borne s_i. On suppose également que pour tout i, il existe k et p tels que $c_{ik} = s_p$ (c'est-à-dire que le dernier nœud d'un chemin alternatif est un nœud du chemin de base).

Algorithme 5.1 Génération de chemins alternatifs

```
1: chemin de base ← planificateur de la section 5.1
2: pour tout s_i ∈ Q faire
3:     supposer temporairement que f_{s_i} ≡ 1          ▷ Simule que s_i est occupée
4:     nouveau chemin ← planificateur de la section 5.1
5:     si nouveau chemin = ancien chemin alors
6:         continue                                     ▷ Pas de chemin alternatif pour cette borne
7:     fin si
8:     b_i ← dernier nœud en commun du préfixe des deux chemins
9:     considérer le nouveau chemin comme alternative sur le nœud b_i
10: fin pour
```

Remarque. Ce ne sont pas toutes les bornes s qui mènent à la génération d'un chemin alternatif, car il est parfois moins long d'attendre que d'aller ailleurs.

Une fois les chemins alternatifs créés, l'exécution du plan a lieu comme suit. Une fois le VÉ arrivé à un nœud de décision, une requête à un serveur contenant les données d'occupation en temps réel de chaque borne s (qu'on notera **otr(s)**) est lancée. Si la prochaine borne prévue sur le chemin de base est occupée, le planificateur en ligne envoie le VÉ sur le chemin alternatif. La stratégie d'exécution de ce planificateur en ligne est présentée à l'algorithme 5.2.

Algorithme 5.2 Exécution en ligne du plan

```
1: procédure EXÉCUTERPLAN(π)
2:     n ← α
3:     tant que n ≠ ω faire
4:         (x, y) ← π(n)
5:         si y = − ∨ ¬otr(x) alors
6:             n ← x
7:         sinon
8:             n ← y
9:         fin si
10:        Faire rouler le VÉ jusqu'au nœud n
11:    fin tant que
12: fin procédure
```

Remarque. On suppose que l'occupation de chaque borne (borne occupée ou non) peut être connue pendant l'exécution du plan, mais que le temps d'attente ne peut pas l'être (c'est-à-dire que l'information sur l'existence éventuelle d'une file d'attente à la borne n'est pas disponible pour le planificateur). Cette supposition est réaliste par rapport au manque actuel de données disponibles vis-à-vis des réseaux publics de bornes de recharge.

La méthode présentée dans cette section est simple, amène à une augmentation du temps de précalculs négligeable, et réduit significativement le temps moyen de voyagement, lorsque comparée à la technique de réétiquetage sans chemins alternatifs (section 5.1).

On pourrait penser qu'il est possible de simplement recalculer un nouveau chemin à chaque nœud du chemin de base parcouru plutôt que de calculer des chemins alternatifs avant le début du trajet, tout en ayant des résultats similaires. Or, cela amènerait différents problèmes. En effet, le nombre d'appels au serveur qui contient les données d'occupation en temps réel serait de $\mathcal{O}(|P|)$, tandis que notre technique ne nécessite que $\mathcal{O}(|Q|)$ appels au serveur, donnant ainsi des résultats similaires à un recalcul systématique (à cause du positionnement ingénieux du

nœud de décision), tout en induisant une utilisation des ressources beaucoup plus faible du côté du serveur.

5.3 Évaluation

Pour évaluer les deux méthodes proposées, on utilise la même carte et les mêmes 1000 requêtes qu'à la section 4.2. Les bornes sont distribuées dans le même réseau (le Circuit Électrique), mais des données plus récentes ont été utilisées. Il y avait dans cette version 140 bornes de niveau 3 et 1318 bornes (tous les niveaux confondus). Les résultats de l'algorithme de base (algorithme 2.1) sont comparés avec les deux méthodes proposées (l'utilisation de l'étiquetage ξ ainsi que la génération de chemins alternatifs). De plus, pour mesurer l'effet de l'emplacement et de la densité des bornes sur la carte sur les deux méthodes proposées, des bornes artificielles ont été générées. Elles sont distribuées uniformément parmi tous les nœuds sur la carte. Puisque les grandes villes possèdent plus d'embranchements (et donc de nœuds) que les petits villages, plus de bornes y seront générées, ce qui est réaliste (il y a par exemple plus de bornes à Montréal et à Québec qu'à Tadoussac ou à Sainte-Marguerite)[2]. Les données de bornes artificielles ont été testées avec respectivement $250, 500, 1000$ et 2000 bornes.

Les familles de fonctions f_s et g_s (les données réelles pour l'occupation et le temps d'attente à chaque heure et à chaque jour) ont été construites à partir de données extraites du Circuit Électrique entre décembre 2017 et juin 2019. Pour simuler l'augmentation de l'occupation des bornes dans les prochaines années, nous avons ajouté un facteur multiplicatif dans les fonctions f_s qui est utilisé dans les tests (1 pour la valeur réelle, et 2 et 3 pour simuler une plus grande occupation). Pour ce

2. Un modèle encore plus précis pourrait utiliser des données donnant la densité de circulation pour faire le placement des bornes

TABLE 5.1: Résultats pour 1000 requêtes en considérant l'occupation

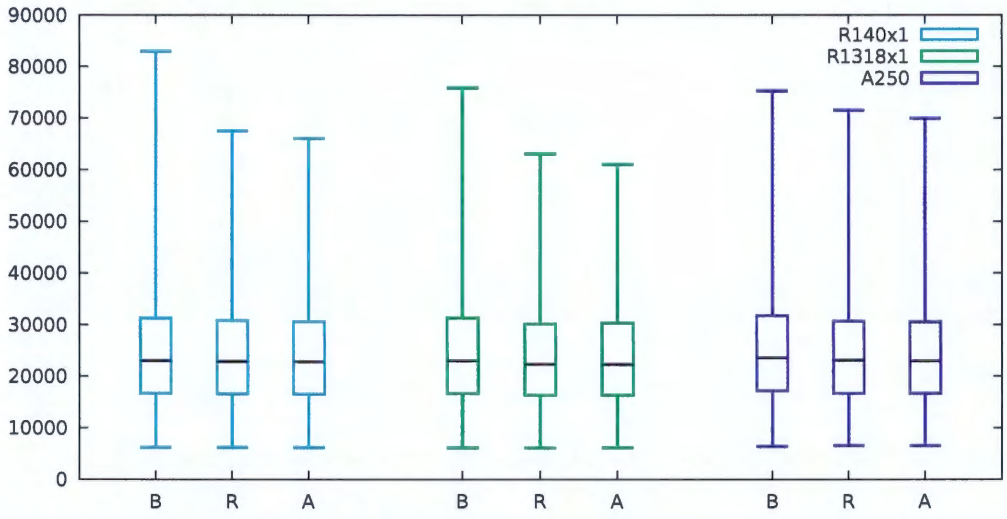
Paramètres		Base		Réétiquetage				Chemins alternatifs			
RB	MP	A	AD	A	AD	RA	RT	A	AD	RA	RT
		min	min	min	min	%	min	min	min	%	min
R140	× 1	10,2	350,1	3,3	343,9	-67,5	-6,2	2,2	342,8	-78,8	-7,2
R140	× 2	22,7	362,5	6,0	347,0	-73,6	-15,5	4,1	345,3	-82,0	-17,2
R140	× 3	37,5	377,3	7,7	349,0	-79,3	-28,3	7,2	348,5	-80,9	-28,7
R140	Aléa	51,3	391,2	10,9	352,6	-78,7	-38,5	9,8	351,6	-80,9	-39,5
R1318	× 1	19,4	356,0	2,2	339,2	-88,6	-16,7	1,7	338,7	-91,5	-17,3
R1318	× 2	37,5	374,0	4,0	341,2	-89,3	-32,8	3,0	340,3	-92,0	-33,7
R1318	× 3	50,5	387,1	6,5	343,9	-87,1	-43,1	4,7	342,3	-90,6	-44,8
R1318	Aléa	62,0	398,6	6,9	345,3	-88,8	-53,3	6,0	344,4	-90,4	-54,2
A250	Aléa	29,3	368,2	12,0	354,2	-59,0	-13,9	10,3	353,1	-64,8	-15,1
A500	Aléa	28,9	363,4	9,4	346,9	-67,6	-16,5	8,3	346,1	-71,2	-17,3
A1000	Aléa	28,7	362,5	7,4	343,8	-74,2	-18,7	6,5	343,1	-77,3	-19,4
A2000	Aléa	27,1	359,9	4,9	340,0	-81,9	-19,9	3,8	339,0	-86,0	-20,9

RB : Réseau de bornes (où R = réelles et A = artificielles) ;
MP : Modificateur des probabilités ; **A** : Temps d'attente moyen ;
AD : Temps total moyen (Attente + Déplacement) ;
RA : Réduction de l'attente (vs Base) ; **RT** : Réduction du total (vs Base).

qui est des bornes artificielles, les valeurs $f_s(j,h)$ ont été générées uniformément entre 0 et 1, tandis que les valeurs $g_s(j,h)$ ont été générées en utilisant une distribution de Kumaraswamy (Jones, 2009) (avec les paramètres 2 et 100, mise à l'échelle entre 1 minute et 6 heures), ce qui donne une moyenne de 31,6 minutes. Cette distribution a été choisie, car il a été mesuré en pratique qu'un conducteur de VÉ reste en moyenne 30 minutes à une borne de recharge, mais plusieurs d'entre eux restent beaucoup plus longtemps (ex. : toute la nuit à un hôtel, ou pendant 2 heures au restaurant) ce qui fait qu'une distribution asymétrique positive a plus de sens qu'une distribution symétrique.

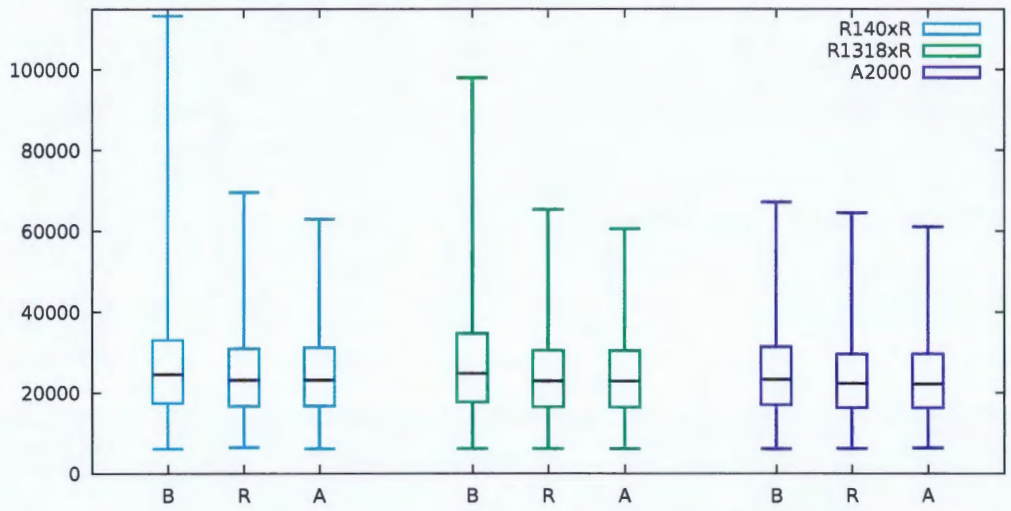
Pour les mêmes raisons que dans l'évaluation des regroupements à la section 4.2, on suppose que :

$$\sigma(e) = 90 \text{ km/h} \quad \text{pour tout } e \in E$$
$$\text{TRB}(s) = 30 \text{ min} \quad \text{pour tout } s \in S.$$



B : Base ; **R** : Réétiquettage ; **A** : Chemins alternatifs ;

FIGURE 5.2: Boîtes à moustaches montrant le résumé en 5 nombres du temps total



B : Base ; **R** : Réétiquettage ; **A** : Chemins alternatifs ;

FIGURE 5.3: Boîtes à moustaches montrant le résumé en 5 nombres du temps total

La tableau 5.1 montre les résultats obtenus en exécutant les tests décrits ci-haut. Les deux premières colonnes présentent les variables testées : le réseau de bornes RB utilisé (où R et A veulent dire respectivement qu'on utilise les données réelles et artificielles) et le modificateur MP de probabilité (où *Aléa* veut dire qu'on utilise des probabilités générées aléatoirement plutôt que les données réelles). Les deux prochaines colonnes présentent le temps d'attente moyen (A) et le temps moyen total (AD) de l'algorithme de base. Finalement, les deux groupes de 4 colonnes restants présentent respectivement les résultats pour les deux techniques proposées. Les colonnes A et AD ont la même signification que précédemment, tandis que les colonnes RA et RT comparent la technique avec l'algorithme de base en montrant la différence de temps d'attente (en pourcentage) et la différence de temps total (en minutes). Chacune des valeurs dans la table a été arrondie à une décimale.

En considérant la colonne RB avec les colonnes RA et RT, on observe que l'augmentation du nombre de bornes sur la carte (par exemple en passant de R140 à

R1318, ou de A250 à A2000) mène à une diminution du temps d'attente et du temps total (pour les deux techniques proposées). À titre d'exemple, lorsque le nombre de bornes augmente de 140 à 1318, le temps d'attente diminue de 67,5 % à 88,6 % pour la technique de réétiquetage, et de 78,8 % à 91,5 % pour la technique de génération de chemins alternatifs. Le même phénomène s'observe avec les données de bornes artificielles.

L'augmentation des probabilités d'occupation (en augmentant le multiplicateur des probabilités MP) augmente évidemment le temps d'attente moyen, permettant ainsi une plus grande réduction du temps total des trajets (ce qu'on peut constater en comparant les valeurs des deux dernières colonnes pour des lignes ayant des valeurs MP différentes).

Les figures 5.2 et 5.3 montrent des boîtes à moustaches pour chacune des trois techniques (base, réétiquetage et chemins alternatifs), respectivement pour les réseaux de bornes R140 × 1, R1318 × 1 et A250 (pour la première figure), et R140 × R, R1318 × R et A2000 (pour la seconde figure). Elles montrent respectivement le minimum, le premier quartile, la médiane, le troisième quartile et le maximum du temps moyen total des requêtes (en secondes). Bien que la technique des chemins alternatifs ne réduise pas significativement le temps moyen, elle réduit le temps maximum, permettant ainsi de diminuer la variance.

Somme toute, la diminution du temps d'attente, couplée avec une augmentation légère du temps de déplacement (pour les chemins alternatifs) semble réduire significativement le temps de trajet total dans les deux cas. Ces résultats s'appliquent non seulement aux données réelles testées, mais aussi aux données artificielles (positionnement, probabilités d'occupation et temps d'attente aléatoires), ce qui laisse penser que les résultats observés s'appliquent à une grande variété de situations, et pas seulement au cas spécifique du Circuit Électrique.

En conclusion, les résultats montrent que la première méthode a diminué de plus de $\frac{3}{4}$ le temps moyen d'attente tout en ayant un impact négligeable sur le temps de déplacement. Le temps total moyen de chaque requête a quant à lui diminué de $17,3$ minutes pour ce qui est des bornes réelles avec l'occupation réelle. La deuxième méthode, qui s'ajoute à la première, améliore davantage les résultats obtenus (quoique dans une moindre mesure). Elle diminue le temps d'attente de certaines requêtes, diminuant ainsi le maximum d'attente et la variance de l'attente.

La diminution du temps d'attente combinée en utilisant les deux méthodes est correlée à la densité des bornes sur la carte (plus il y avait de bornes dans les données, plus le temps d'attente était réduit par les techniques) et à la probabilité d'occupation, deux éléments qui vont en augmentant (Stubbe, 2018; Irle, 2018) à mesure que des gens achètent des VÉ et que les gouvernements installent de nouvelles bornes. Cela indique que l'efficacité des deux techniques ira en augmentant au fur et à mesure que le temps avance.

CONCLUSION

Ce mémoire a abordé le problème de planification d'itinéraires pour VÉ en considérant les contraintes d'autonomie, les spécificités des VÉ ainsi que l'incertitude reliée à la disponibilité des bornes de recharge.

Dans un premier temps, un algorithme de base permettant de résoudre le problème a été décrit. Une étude de la littérature récente portant sur le problème de planification pour VÉ a ensuite été présentée pour voir s'il existait des algorithmes de pointe permettant d'ajouter certaines fonctionnalités et de relaxer certaines suppositions de l'algorithme de base. Cela a permis de découvrir certains algorithmes tels que Energy-A* qui permet au planificateur de prendre en compte la recharge de batterie due à une baisse d'altitude ou à un freinage, et la contraction hiérarchique de graphe qui permet de diminuer significativement le temps de calcul en faisant des prétraitements dans le graphe représentant le réseau de transport.

Dans un deuxième temps, une première contribution originale a été proposée : le regroupement de bornes de recharge. Cette méthode a permis de rendre en moyenne 35 fois plus rapide le temps de calcul du plus court chemin dans le graphe des bornes avec pour seul compromis une possible augmentation de moins d'un pourcent la longueur du trajet retourné par le planificateur. Bien que la différence pourrait ne pas être perceptible pour les utilisateurs du planificateur (car c'est dans l'ordre de la milliseconde), le gain est important du point de vue du serveur qui aura à traiter beaucoup de requêtes dans un court intervalle de temps. Cette contribution a mené à un article de conférence ainsi qu'à une présentation à Paris dans le cadre des 25$^{\text{ième}}$ rencontres de la Société Francophone de Classifica-

tion (Champagne Gareau **et al.**, 2018) ainsi qu'à une présentation à Thessaloniki en Grèce dans le cadre de la 16$^{\text{ième}}$ conférence de la Fédération internationale des Sociétés de Classification (Champagne Gareau **et al.**, 2019).

Dans un troisième temps, une seconde contribution originale a été proposée : la prise en compte du temps d'attente aux bornes dans la fonction à minimiser. Pour ce faire, deux méthodes ont été proposées. La première repose sur l'utilisation de plusieurs années de données réelles d'occupation des bornes pour construire un modèle probabiliste qui prédit l'occupation en fonction du jour et de l'heure et qui utilise ce modèle dans l'algorithme de planification pour éviter les bornes ayant une grande espérance d'attente. La seconde méthode génère une politique d'exécution qui contient des chemins alternatifs qui peuvent être utilisés dans le cas où l'occupation est pire que prévu quelques instants avant d'arriver à la prochaine borne prévue par le plan initial. Ces méthodes ont permis de diviser par 4 le temps d'attente ainsi qu'une diminution moyenne de 17.3 minutes du temps total des itinéraires (sur des itinéraires moyens d'environ 6 h initialement). Cette contribution a mené à une publication dans une conférence internationale de rang A (selon le classement CORE), ACM SIGSPATIAL (Champagne Gareau **et al.**, 2019).

Dans un dernier temps, l'algorithme de base, les contributions proposées ainsi que des améliorations provenant de l'état de l'art ont été implémentés. Un site web a aussi été conçu pour rendre accessible le planificateur au grand public. Il y a encore plusieurs éléments qui peuvent être améliorés par les futurs planificateurs pour VÉ. Une piste de recherche consiste à considérer la possibilité que le planificateur soit lié à un système de réservation de bornes. Ce faisant, il pourrait réserver la borne à l'instant d'arrivée prévu. Le système devrait considérer l'optimalité globale en fonction de tous les utilisateurs du système, et pas seulement l'optimalité d'un utilisateur spécifique comme c'est présentement le cas.

ANNEXE A

PLANIFICATEUR VEPLAN

> *C++ is designed to allow you to express ideas, but if you don't have ideas or don't have any clue about how to express them, C++ doesn't offer much help.*
>
> — Bjarne Stroustrop

Cette annexe présente les détails d'implémentation et d'utilisation du planificateur développé dans cette maîtrise, qu'on appelle VEPlan.

A.1 Implémentation

VEPlan est écrit en C++ (standard 2017) et compilé avec g++ du projet GNU Compiler Collection 8.3.0. L'ensemble du projet (l'implémentation de base et les contributions des chapitres 4 et 5, de même que les articles, affiches et outils mentionnés dans ce mémoire) est disponible à l'adresse `https://gitlab.com/jaja360/Veplan`. Les modules qui composent le projet sont présentés dans un diagramme à la figure A.1. Les classes ne dépendent que de la bibliothèque standard de C++17 (incluant les structures de données), à l'exception de la classe *kdtree*, qui provient de GitHub [1].

1. `https://github.com/gishi523/kd-tree`

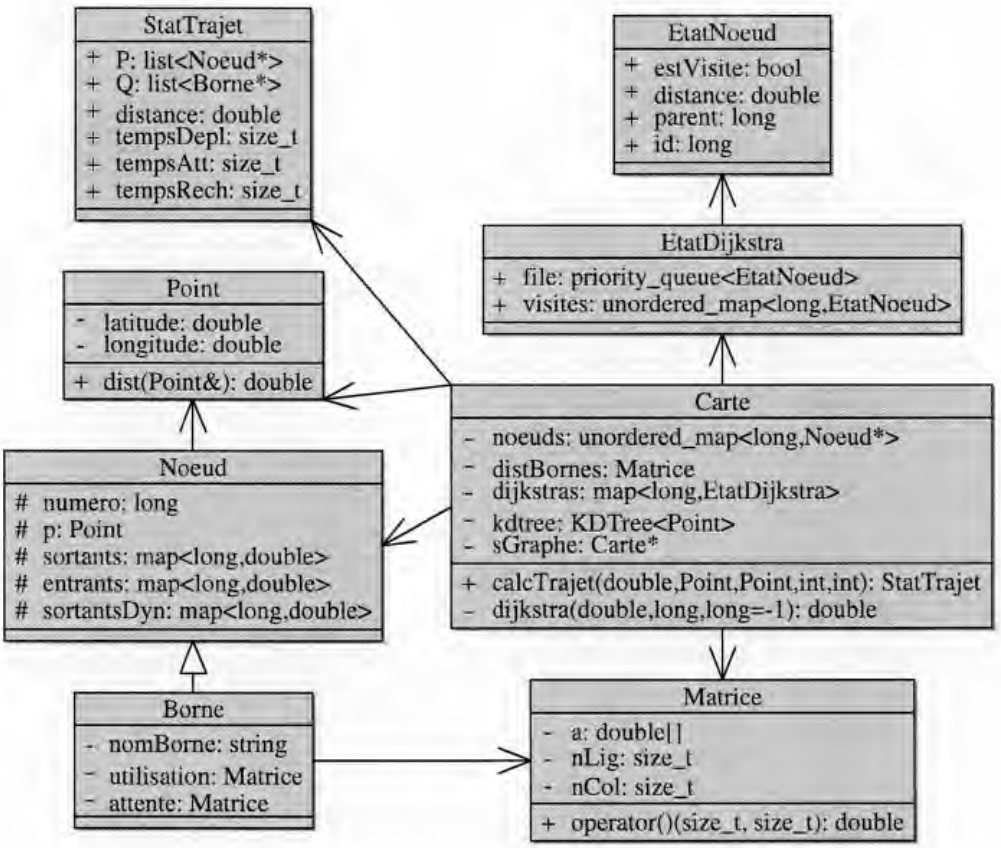


FIGURE A.1: Diagramme UML de l'implémentation

La fonction principale de VEPlan est la fonction *calcTrajet*, qui prend en paramètre une requête $(\rho, \alpha, \omega, \text{jour, heure})$ et retourne une structure contenant les informations utiles trouvées pour l'itinéraire, incluant la solution (P, Q), la distance, et les temps de déplacement, de recharge et d'attente (les deux derniers sont simulés de façon probabiliste). La fonction est une implémentation modifiée de l'algorithme 2.1. Pour ne pas avoir à itérer sur chaque arête du s-graphe pour adapter celui-ci lorsque l'autonomie des requêtes change, on distingue dans chaque nœud les arêtes sortantes et entrantes (qui ne dépendent pas de la requête) et les arêtes sortantes dynamiques. Ces dernières sont celles qui sont modifiées à la ligne 6 et sont réinitialisées à la ligne 9 de l'algorithme 2.1.

Dans notre implémentation, la fonction *dijkstra* choisit automatiquement entre l'algorithme de Dijkstra et l'algorithme A* selon qu'un point d'arrivée a été passé en paramètre ou non. L'état final de l'exécution de cette fonction est sauvegardé dans un tableau associatif appelé *dijkstras* permettant de récupérer facilement les distances minimales et les chemins optimaux entre le départ spécifié et tous les nœuds explorés. La sauvegarde permet aussi de poursuivre une exécution de `Dijkstra` où elle était rendue lorsqu'elle a été interrompue. Par exemple, cela peut être utile si une requête part d'un même nœud de départ qu'une précédente requête, mais avec une autonomie plus grande.

La matrice *distBornes* correspond à la matrice D de l'algorithme 2.1. Puisque celle-ci contient toutes les distances précalculées au lancement du programme et que ces précalculs sont généralement assez long à faire (pour les grandes cartes lorsqu'il y a beaucoup de bornes), le programme sauvegarde la matrice dans un fichier la première fois que les précalculs sont faits, et le fichier est simplement lu lors des prochains lancements du programme lorsqu'il est présent.

A.2 Utilisation

```
Usage: ./veplan [-h] [-v] [-s] [-o] [-f N] [-g N] [-G N] [-d D] [-r REQS] -c CARTE -b BORNES
Calcul du plus court chemin pour un véhicule électrique (VÉ)

Description des options:
  -c --carte CARTE            Le chemin de la carte à utiliser
  -b --bornes BORNES          Le chemin du fichier de bornes à utiliser
  -r --requetes REQS[=stdin]  Le chemin du fichier de requêtes à utiliser
  -d --dmax D[=0]             Le paramètre de clustering (en km)
  -g --genRequetes N          Génère N requêtes sur stdout
  -G --genBornes N            Génère N bornes sur la carte
  -f --facteur N[=5]          Multiplie la probabilité d'occupation par N
                              -1 : génération aléatoire des probabilités
  -t --contingences T         Génère des contingences de type T
  -o --occupation             Considérer l'occupation lors de la planif.
  -s --sous_graphe            Écrit sur stdout le graphe des bornes
  -v --verbose                Affiche les stats et le temps d'exécution
  -h --help                   Affiche ce message
```

FIGURE A.2: Options du planificateur VEPlan

```
n5989664859 (45.625506,-73.83739);
n5989664860 (45.624802,-73.836374);
---
Autoroute_du_Souvenir: n17074254 n35835087;
Avenue_du_Mont-Royal_Est: n26232411 n5022472271 n26232412;
```

FIGURE A.3: Extrait d'un fichier de carte

Le planificateur VEPlan s'utilise directement en ligne de commande et plusieurs options sont offertes. La description de celles-ci s'affiche lorsqu'on utilise l'option **-h** et est présentée à la figure A.2. Comme affiché au synopsis à la première ligne sur la figure, les options **-c** et **-b** sont obligatoires et permettent de spécifier le fichier contenant la carte et le fichier contenant les bornes à utiliser. Les figures A.3 et A.4 présentent respectivement un exemple minimal de fichier de carte et de bornes, pour illustrer le format.

Les fichiers de carte sont construits en utilisant les données d'OpenStreetMap (OSM) (Weber et Haklay, 2008), qui sont librement accessibles sous licence Open-Data (ODbL) [2]. La première partie du fichier contient la liste des nœuds, composée du numéro d'identification OSM et des coordonnées (latitude, longitude). L'autre partie représente les routes. Une route est une suite de segments droits (arêtes).

Quant à elles, les bornes sont composées d'un nom, d'un niveau (L1, L2 ou L3), de coordonnées et, optionnellement, d'une matrice donnant les probabilités d'occupation et d'une autre donnant les temps d'attente, lorsqu'occupée. On peut multiplier par un facteur N les probabilités lues dans le fichier à l'aide de l'option **-f N**. Les fichiers de bornes sont générés à partir des données réelles du Circuit électrique. Il est aussi possible de générer N bornes artificielles avec l'option **-G N**.

Le planificateur peut lire des requêtes provenant de l'entrée standard (*stdin*) ou

2. `https://www.openstreetmap.org/copyright/fr`

```
BRCC_-_Sainte-Jeanne-d'Arc_-_Crevier L3 (48.857444,-72.099139)
Mon 0 0 0 0 0 4 0 0 0 0 0 4 1 0 0 1 5 1 2 0 0 0 2 0
Tue 0 0 0 0 0 1 4 4 0 0 1 1 1 0 0 2 2 5 7 3 0 0 1 0
Wed 0 1 0 0 0 0 0 0 1 2 0 0 1 1 5 3 0 3 1 2 2 2 5 0
Thu 0 0 0 0 0 0 0 0 0 1 0 0 2 7 4 2 1 0 2 3 2 0 0 1 0
Fri 0 1 0 0 0 0 0 1 5 2 0 2 2 3 4 0 1 0 1 0 0 0 0 1
Sat 1 0 0 0 0 0 1 1 0 4 1 1 1 1 2 2 1 2 0 1 1 4 5 0
Sun 0 0 0 0 0 0 1 3 2 5 3 3 3 0 0 0 4 0 0 1 0 1 1 0
-
Mon 0 0 0 0 0 16 0 0 0 0 0 0 17 13 0 0 13 17 8 20 0 0 0 13 0
Tue 0 0 0 0 0 10 23 18 0 0 10 28 13 0 0 13 17 80 25 24 0 0 10 0
Wed 0 5 0 0 0 0 0 0 10 18 0 0 13 58 24 15 0 19 13 18 47 20 19 0
Thu 0 0 0 0 0 0 0 0 5 0 0 65 22 18 10 5 0 20 21 20 0 0 25 0
Fri 0 10 0 0 0 0 0 85 50 13 0 13 13 39 17 0 10 0 10 0 0 0 0 18
Sat 5 0 0 0 0 0 28 13 0 27 13 18 5 10 12 12 10 13 0 28 13 16 35 0
Sun 0 0 0 0 0 0 40 17 72 65 20 19 13 0 0 0 18 0 0 13 0 13 13 0
```

FIGURE A.4: Extrait d'un fichier de bornes

d'un fichier. Dans ce second cas, on spécifie le fichier de requêtes avec l'option **-r**. On peut générer un fichier de N requêtes aléatoires avec l'option **-g N**. Une requête ressemble à :

```
(45.473614,-72.295532) (45.598666,-74.196167) 85500 0 13,
```

incluant respectivement les coordonnées du départ et de l'arrivée, l'autonomie ainsi que le jour et l'heure de départ.

L'option **-d δ** permet de spécifier le paramètre $d_{\max} = \delta$ pour choisir le rayon des regroupements de bornes présenté au chapitre 4. L'option **-o** permet au planificateur d'utiliser un réétiquetage des arêtes pour considérer l'occupation, tandis que l'option **-t** permet de générer des chemins alternatifs.

Finalement, les options **-s** et **-v** permettent respectivement d'afficher le s-graphe sur la sortie standard (*stdout*) et d'activer le mode verbeux, pour afficher plus

d'informations, telles que le temps d'exécution de chaque étape du calcul.

FIGURE A.5: Visionneuse Java

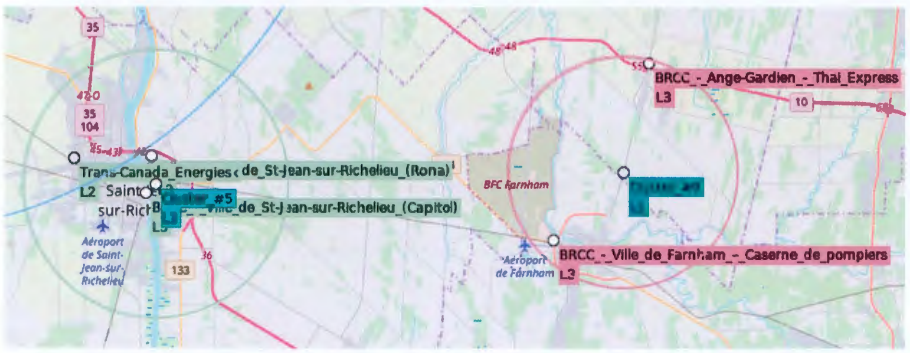


FIGURE A.6: Illustration des regroupements avec la visionneuse

Pour chaque requête, VEPlan écrit dans la sortie standard (*stdout*) les éléments de la structure *StatTrajet*. Plutôt que d'utiliser VEPlan directement en ligne de commande, le projet inclut aussi une visionneuse faite en Java utilisant JMap-Viewer du projet JOSM permettant de visualiser les solutions et les bornes sur la carte. Celle-ci communique avec VEPlan et affiche l'itinéraire optimal pour chaque requête. La figure A.5 montre l'affichage par la visionneuse d'un chemin retourné par VEPlan entre l'Outaouais et la Gaspésie. Il est également possible de voir les regroupements de bornes générés par VEPlan, incluant le centre du

regroupement et un cercle de rayon $d_{\max}$ autour du centre. La figure A.6 montre comment s'affichent les regroupements.

Pour que le planificateur puisse être utilisé par les conducteurs de VÉ et puisse contribuer à la démocratisation des VÉ, un site Web simple d'utilisation et faisant appel à VEPlan a aussi été développé. Celui-ci utilise Leaflet, une bibliothèque JavaScript, pour afficher la carte et les marqueurs. Les requêtes sont transmises au serveur, qui utilise la bibliothèque Flask en Python pour le *back-end* et s'occupe d'appeler VEPlan et de retransmettre au client les résultats, incluant le trajet, les prévisions de temps (déplacement, attente et recharge) et l'heure prévue d'arrivée.

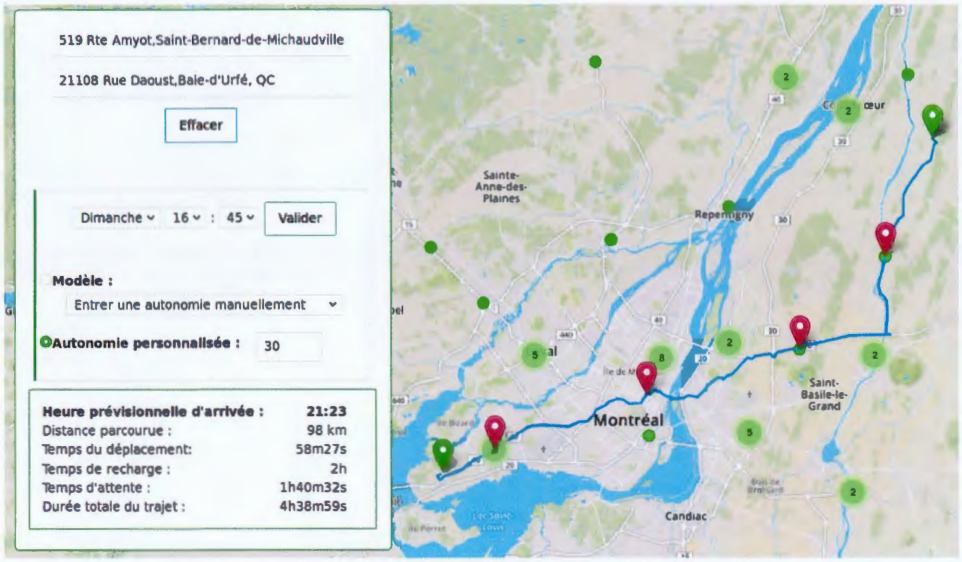


FIGURE A.7: Site Web faisant appel à VEPlan

ANNEXE B

DÉMONSTRATION DU THÉORÈME 2.5

Pour démontrer le théorème 2.5, il faut d'abord démontrer deux résultats intermédiaires. Les lemmes B.1 et B.2 prouvent respectivement ces deux éléments. Des propriétés de A* utilisées dans la preuve du lemme B.1 sont présentées dans le livre de Russell et Norvig (2009).

Lemme B.1. *L'heuristique* $h(n) := d_{GC}(n, \omega)$ *est monotone (aussi appelée consistante), où* $d_{GC}(x, y)$ *est la distance du grand cercle de* x *à* y *;*

Preuve. Soient $m = (s_1, t_1)$ et $n = (s_2, t_2)$ (où s_i, t_i donnent respectivement la latitude et la longitude du point). Par définition,

$$d_{GC}(m, n) = r \arccos(\sin(s_1)\sin(s_2) + \cos(s_1)\cos(s_2)\cos(t_2 - t_1)).$$

La distance du grand cercle est une métrique au sens mathématique. Elle a donc les trois propriétés suivantes [1] :

1. $d_{GC}(m, n) = 0$ ssi $m = n$ (identité)
2. $d_{GC}(m, n) = d_{GC}(n, m)$ (symétrie)
3. $d_{GC}(m, n) \leq d_{GC}(m, p) + d_{GC}(p, n)$ (inégalité triangulaire)

1. Il est possible de les démontrer, mais c'est orthogonal à l'objectif de ce mémoire

Sachant cela, on a que $\forall (n, m) \in E'$,

$$\begin{aligned} h(n) &= d_{GC}(n, \omega) && \text{(par définition)} \\ &\leq d_{GC}(n, m) + d_{GC}(m, \omega) && \text{(propriété 3)} \\ &= \lambda(n, m) + h(m) && \text{(par définition)} \end{aligned}$$

Par conséquent, h est consistante (donc admissible), et A* retourne donc une solution de coût minimal. □

Lemme B.2. *Le coût considéré par A* dans le s-graphe représente la fonction Z' où $Z'(P, Q) = \mathrm{TD}(P) + \mathrm{TR}(Q)$.*

Preuve. On veut montrer que le coût considéré par A* correspond à la fonction objectif Z' (c.-à-d. qu'il tient compte du temps de déplacement et de recharge). Par construction, il est assez évident que le temps de déplacement est considéré. Il reste à voir si le temps de recharge l'est (ou s'il doit l'être explicitement).

Notons d'abord que s'il existe un chemin sans recharge entre deux nœuds n et m, alors il y a un arc $(n, m) \in E'$. En ne considérant pas le temps de recharge dans le coût, le pire cas qui peut survenir est donc d'avoir des nœuds *en trop* dans Q. Par contre, la suite P obtenue sera correcte (c.-à-d. de temps de déplacement minimal). Dépendamment de l'implémentation, ça peut être ou ne pas être important, mais la considération explicite a comme avantage de permettre de distinguer la vitesse de différents modèles de borne (ex. : niveau 1, 2 ou 3).

La figure B.1 montre un exemple où la suite Q retournée n'est pas toujours la meilleure lorsque le temps de recharge n'est pas explicitement ajouté au coût. En effet, dans l'exemple, le coût est le même avec les deux chemins possibles, mais le chemin du haut ajoutera la borne N à Q, laissant penser au conducteur qu'il

doit se recharger à N même si ce n'est pas nécessaire. L'algorithme a autant de chance de retourner l'un ou l'autre de ces chemins, car l'ordre d'exploration de voisins ayant la même valeur heuristique n'est pas déterministe pour A*.

Pour pallier ce problème, il suffit d'ajouter une pénalité (correspondant au temps de recharge moyen) pour chaque nœud traversé par A*. L'algorithme choisira donc la solution donnant la suite Q de taille la plus petite possible lorsqu'il aura le choix entre deux solutions de même coût de déplacement. Nous verrons au chapitre 5 comment généraliser à des temps de recharge variables. □

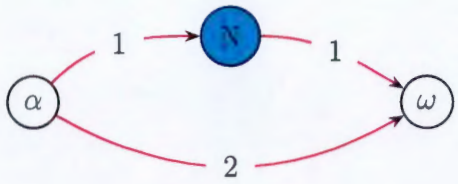


FIGURE B.1: Exemple sans considération du temps de recharge

Avec les deux lemmes précédents, nous sommes maintenant prêts à prouver le théorème 2.5.

Théorème. *L'algorithme 2.1 donne une solution optimale (au sens de la définition 2.4) lorsqu'on ne considère pas le temps d'attente.*

Preuve. On veut montrer que le doublet (P, Q) calculé par l'algorithme est un minimum global de la fonction $Z'(P, Q) = \mathrm{TD}(P) + \mathrm{TR}(Q)$.

Par le lemme B.1, A* retourne une solution de coût minimale. Par le lemme B.2, le coût considéré correspond à la fonction objectif Z' lorsqu'une pénalité sur le nombre de nœuds traversé est ajoutée à l'algorithme A* (ligne 7 de l'algorithme 2.1). Avec cette modification, le doublet (P, Q) retourné par l'algorithme est bien une solution optimale au sens de la définition 2.4. □

BIBLIOGRAPHIE

Add Énergie (2019). Le circuit électrique. Récupéré le 2019-06-17 de `https://lecircuitelectrique.com/`

Adler, J. D., Mirchandani, P. B., Xue, G. et Xia, M. (2016). The electric vehicle shortest-walk problem with battery exchanges. *Networks and Spatial Economics*, *16*(1), 155–173.

Artmeier, A., Haselmayr, J., Leucker, M. et Sachenbacher, M. (2010). The shortest path problem revisited : Optimal routing for electric vehicles. Dans *Proceedings of the 33rd Annual German Conference on Artificial Intelligence (KI-2010)*.

Baum, M., Sauer, J., Wagner, D. et Zündorf, T. (2017). Consumption profiles in route planning for electric vehicles : Theory and applications. Dans *International Symposium on Experimental Algorithms (SEA)*, volume 75, 19 :1–19 :18. Dagstuhl Publishing.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517.

Bonneau, S. (2019). Nos stratégies de recharge, capitales ! *Hydro Presse*, *Été 2019*, 42–47. ISSN 0707-9605.

CAA (2019). Types de véhicules électriques. Récupéré le 2019-06-15 de `https://www.caa.ca/fr/vehicules-electriques/types-de-vehicules-electriques/`

Champagne Gareau, J., Beaudry, É. et Makarenkov, V. (2018). Planification d'itinéraires quasi-optimaux pour un véhicule électrique en considérant le regroupement de bornes de recharge et leur probabilité d' occupation. Dans *XXV èmes Rencontres de la Société Francophone de Classification (SFC2018)*, 5–8., Paris, France.

Champagne Gareau, J., Beaudry, E. et Makarenkov, V. (2019). An efficient electric vehicle path-planner that considers the waiting time. Dans *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '19, 389–397., New York, NY, USA. ACM.

Champagne Gareau, J., Beaudry, É. et Makarenkov, V. (2019). A fast electric vehicle planner using clustering. Dans *16th Conference of the International Federation of Classification Societies (IFCS2019)*, Thessaloniki, Grèce.

Chartrand, G. et Zhang, P. (2012). *A first course in graph theory.* Dover Publications. ISBN 0486483681.

De Cauwer, C., Van Mierlo, J. et Coosemans, T. (2015). Energy consumption prediction for electric vehicles based on real-world data. *Energies*, *8*(8), 8573–8593.

Delling, D. et Wagner, D. (2009). Time-dependent route planning. Dans *Lecture Notes in Computer Science*, volume 5868 LNCS, 207–230.

Delling, D. et Werneck, R. F. (2013). Faster customization of road networks. Dans *Lecture Notes in Computer Science*, volume 7933 LNCS, 30–42. Springer.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271.

Eckhouse, B., Stringer, D. et Hodges, J. (2019). The world still doesn't have enough places to plug in cars - bloomberg. Récupéré le 2019-06-02 de `https://www.bloomberg.com/news/features/2019-02-14/the-world-still-doesn-t-have-enough-places-to-plug-in-cars`

Eisner, J., Funke, S. et Storandt, S. (2011). Optimal route planning for electric vehicles in large networks. Dans *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 1108–1113.

Erdelić, T. et Carić, T. (2019). A survey on the electric vehicle routing problem : Variants and solution approaches. *Journal of Advanced Transportation*, *2019*, 1–48.

Fredman, M. L. et Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, *34*(3), 596–615.

Funke, S., Nusser, A. et Storandt, S. (2015). Placement of loading stations for electric vehicles : No detours necessary ! *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, *53*, 633–658.

Funke, S., Nusser, A. et Storandt, S. (2016). Placement of loading stations for electric vehicles : Allowing small detours. Dans *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 131–139.

Funke, S. et Storandt, S. (2013). Enabling e-mobility : Facility location for battery loading stations. Dans *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 1341–1347.

Geisberger, R., Sanders, P., Schultes, D. et Delling, D. (2008). Contraction hierarchies : Faster and simpler hierarchical routing in road networks. Dans *Experimental Algorithms*, 319–333., Berlin, Heidelberg. Springer.

Geisberger, R., Sanders, P., Schultes, D. et Vetter, C. (2012). Exact routing in large road networks using contraction hierarchies. *Transportation Science*, *46*(3), 388–404.

Gelenbe, E. et Pujolle, G. (1982). *Introduction aux réseaux de files d'attente.* Eyrolles.

Genikomsakis, K. N. et Mitrentsis, G. (2017). A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications. *Transportation Research Part D : Transport and Environment*, *50*, 98–118.

Gopalakrishnan, R., Biswas, A., Lightwala, A., Vasudevan, S., Dutta, P. et Tripathi, A. (2016). Demand prediction and placement optimization for electric vehicle charging stations. Dans *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 3117–3123.

Hart, P., Nilsson, N. et Bertram, R. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Man and Cybernetics, Part A : Systems and Humans - TSMCA*, *4*(2), 100–107.

Hydro-Québec (2015). *Bornes de recharge pour véhicules électriques - Guide technique d'installation.* Rapport technique, Hydro-Québec.

Irle, R. (2018). The Electric Vehicle World Sales Database. Récupéré le 2019-08-17 de `http://www.ev-volumes.com/country/total-world-plug-in-vehicle-volumes/`

Jones, M. C. (2009). Kumaraswamy's distribution : A beta-type distribution with some tractability advantages. *Statistical Methodology*, *6*(1), 70–81.

Kullman, N., Goodson, J., Mendoza, J. E., Kullman, N., Goodson, J., Mendoza, J. E., Vehicle, E., Goodson, J. C. et Mendoza, J. E. (2017). *Electric vehicle routing with uncertain charging station availability dynamic decision making.* Rapport technique.

Lam, A. Y. S., Leung, Y. W. et Chu, X. (2014). Electric vehicle charging station placement : Formulation, complexity, and solutions. *IEEE Transactions on Smart Grid*, *5*(6), 2846–2856.

Lin, J., Zhou, W. et Wolfson, O. (2016). Electric vehicle routing problem. Dans *Transportation Research Procedia*, volume 12, 508–521. Elsevier.

Lindgren, J. et Lund, P. D. (2016). Effect of extreme temperatures on battery charging and performance of electric vehicles. *Journal of Power Sources*, *328*, 37–45.

Liu, K., Yamamoto, T. et Morikawa, T. (2017). Impact of road gradient on energy consumption of electric vehicles. *Transportation Research Part D : Transport and Environment*, *54*, 74–81.

Mehlhorn, K. et Sanders, P. (2008). *Algorithms and data structures : The basic toolbox*. Springer.

Montoya, A., Guéret, C., Mendoza, J. E. et Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B : Methodological*, *103*, 87–110.

Neubauer, J. et Wood, E. (2014). The impact of range anxiety and home, workplace, and public charging infrastructure on simulated battery electric vehicle lifetime utility. *Journal of Power Sources*, *257*, 12–20. `http://dx.doi.org/10.1016/j.jpowsour.2014.01.075`

Randall, T. (2016). Here's how electric cars will cause the next oil crisis. Récupéré le 2019-06-17 de `https://www.bloomberg.com/features/2016-ev-oil-crisis/`

Ressources naturelles Canada (2019). Outil de recherche pour les cotes de consommation de carburant. Récupéré le 2019-06-15 de `https://fcr-ccc.nrcan-rncan.gc.ca/fr`

Roberti, R. et Wen, M. (2016). The electric traveling salesman problem with time windows. *Transportation Research Part E : Logistics and Transportation Review*, *89*, 32–52.

Russell, S. et Norvig, P. (2009). *Artificial Intelligence : A Modern Approach* (3rd éd.). Upper Saddle River, NJ, USA : Prentice Hall Press.

Sachenbacher, M., Leucker, M., Artmeier, A. et Haselmayr, J. (2011). Efficient energy-optimal routing for electric vehicles. Dans *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 1402–1407.

Sipser, M. (2013). *Introduction to the theory of computation* (third éd.). Cengage Learning.

Stubbe, R. (2018). Charging stations keep pace with electric-vehicle growth - bloomberg. Récupéré le 2019-05-01 de `https://www.bloomberg.com/news/articles/2018-05-30/charging-stations-keep-pace-with-electric-vehicle-growth`

Sweda, T. M., Dolinskaya, I. S. et Klabjan, D. (2017). Adaptive routing and recharging policies for electric vehicles. *Transportation Science*, *51*(4), 1326–1348.

Sweda, T. M. et Klabjan, D. (2012). Finding minimum-cost paths for electric vehicles. Dans *IEEE International Electric Vehicle Conference (IEVC)*, 1–4. IEEE.

Weber, P. et Haklay, M. (2008). Openstreetmap : user-generated street maps. *IEEE Pervasive Computing*, *7*(4), 12–18.

Wu, X., Professor, A., Cabrera, A., Freese, D., Kitch, W. A. et Jia, X. (2013). Electric vehicles' energy consumption measurement and estimation. *Transportation Research*, 1–17.

Yang, H., Yang, S., Xu, Y., Cao, E., Lai, M. et Dong, Z. (2015). Electric vehicle route optimization considering time-of-use electricity price by learnable partheno-genetic algorithm. *IEEE Transactions on Smart Grid*, *6*(2), 657–666.

Yuksel, T. et Michalek, J. J. (2015). Effects of regional temperature on electric vehicle efficiency, range, and emissions in the united states. *Environmental Science and Technology*, *49*(6), 3974–3980.