Guider un calcul d'itinéraire multi-critères

Arthur Finkelstein[1] et Jean-Charles Régin[2]

[1] Instant System, Garden Space B2, Rue Evariste Galois, 06410 Biot, France
arthur.finkelstein@instant-system.com
[2] Université Côte d'Azur, I3S, CNRS, France
jean-charles.regin@univ-cotedazur.fr

Mots-clés : *calcul d'itinéraire pour les transports en commun, plus court chemin, optimisation Pareto, précalcul.*

1 Introduction

Calculer un itinéraire est un problème fondamental dans notre société, et ce depuis l'aube des temps. Pouvoir aller de manière efficace d'un point A à un point B est un besoin de tous les jours, cependant trouver ce chemin est une tâche complexe. De plus, il faut trouver des itinéraires efficaces pour les utilisateurs, mais l'efficacité est difficile à définir. Un chemin efficace pour un automobiliste n'est pas le même que pour un cycliste ni le même que pour un autre automobiliste.

Le problème devient encore plus complexe en prenant en compte les transports en commun, qui introduisent la notion de temps, par exemple deux bus différents peuvent passer par les mêmes arrêts à des horaires différents ce qui implique de rajouter beaucoup d'informations dans le réseau. Calculer un trajet avec de multiples transferts va donc nécessiter d'accéder à de nombreuses informations, comme les lignes passant à un arrêt et leurs horaires, les arrêts desservis par une ligne, ce qui va demander du temps de calcul. De plus, les gens ne réussissent pas toujours à trouver le chemin le plus rapide car de nombreux biais existent comme par exemple la navigation par ligne de vue, où on se déplace principalement en direction du point d'arrivée même si le plus court chemin passe par une autre direction [2].

Pour pouvoir espérer répondre aux besoins précis d'un utilisateur sans aucune autre information que le point de départ, le point d'arrivée et l'heure de départ de la recherche d'itinéraire, il faut fournir de multiples résultats pour une même recherche d'itinéraire. Cela passe par l'utilisation d'un front de Pareto qui est une solution efficace pour fournir des résultats intéressants pour un utilisateur. Malheureusement, le coût d'un front de Pareto avec de multiples critères est conséquent, chaque critère augmentant le nombre de résultats et faisant diminuer le temps de réponse. Or, le temps de réponse ne peut pas être trop long pour une utilisation interactive sinon les utilisateurs abandonnent [4].

Le Connection Scan Algorithm (CSA) [3] est un algorithme de plus court chemin pour les transports en commun. Il met de côté la représentation du graphe pour se concentrer sur les connexions, le plus petit élément dans un transport en commun, c'est-à-dire un moyen de transport qui va d'un arrêt à un autre sans arrêts intermédiaire. Les connexions sont triées par heure de départ et vont être scannées pour savoir si elles sont accessibles ce qui va mettre à jour l'heure d'arrivée provisoire de l'arrêt d'arrivée de la connexion. Une fois que toutes les connexions sont scannées, l'heure d'arrivée provisoire pour l'arrêt d'arrivée est l'heure d'arrivée au plus tôt.

Le CSA est l'algorithme le plus rapide pour résoudre le problème du plus court chemin pour les transports en commun avec un front de Pareto [1]. Cependant, les temps de réponse sont trop longs pour une utilisation interactive, car dépassant souvent la seconde. Un point faible du CSA est qu'il n'utilise pas la spatialité du réseau, mais seulement sa temporalité, ce qui implique qu'une certaine partie des calculs ne serviront à rien. Par exemple, le CSA pour une

recherche d'itinéraire entre Lyon et Paris va regarder si des connexions sont atteignables dans toute la France plutôt que de se focaliser sur des régions intéressantes. Le but de nos travaux est d'ajouter cette logique spatiale grâce à une légère phase de précalcul afin de guider la recherche sur des régions intéressantes et d'éviter de chercher inutilement dans certaines régions.

2 Goal Directed Connection Scan Algorithm (GDCSA)

Nous proposons l'algorithme GDCSA (Goal Directed Connection Scan Algorithm). Cet algorithme se base sur un partitionnement du graphe en un ensemble de zones. Ce partitionnement est fait avec l'algorithme Inertial Flow [5], qui est un algorithme simple et efficace qui minimise la taille totale des arêtes coupées.

Pendant une recherche d'itinéraire, GDCSA utilise des bornes inférieures et supérieures sur la durée dans les transports en commun pour ouvrir toutes les zones intéressantes pour l'itinéraire et fermer toutes les zones inintéressantes pour l'itinéraire.

De manière intuitive, la borne inférieure de la durée entre deux arrêts s et t est la durée minimum entre tous les itinéraires d'une journée qui vont de la frontière de la zone contenant s à la frontière de la zone contenant t.

Tout est fait pour diminuer le nombre de zones ouvertes et donc le nombre de nœuds avec des résultats intermédiaires, sans toutefois perdre des résultats du front de Pareto à l'arrêt d'arrivée.

3 Expériences

Nos expériences sur 3 réseaux de transports en commun métropolitains de Paris, Berlin et Stockholm ainsi que sur les 2 réseaux ferroviaires de l'Allemagne et de la Suisse montrent un temps de réponse de 3 à 9 fois plus rapide du GDCSA comparé au CSA.

Cette diminution du temps de réponse s'explique par une diminution de l'espace de recherche. Ainsi le GDCSA diminue le nombre de connexions scannées d'un facteur de 5 à 7 et diminue le nombre d'arrêts avec au minimum un élément dans son front de Pareto d'un facteur de 1.5 à 2, grâce aux zones qui ne sont pas explorées. Le temps de réponse est réduit d'un facteur de 5 à 7 pour les réseaux métropolitains et de 3 à 4 pour les réseaux ferroviaires, cette différence s'explique par les caractéristiques des réseaux, denses pour le métropolitain et épars pour le ferroviaire.

Quant au temps de précalcul, il reste assez faible allant de 5 minutes à 3 heures, ainsi plus un réseau est grand plus le temps de précalcul est long. Cependant, tous les calculs sont séquentiels et le calcul des bornes inférieures peut être parallélisé, ce qui permet de gros gains de temps.

Références

[1] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer, 2016.

[2] Christian Bongiorno, Yulun Zhou, Marta Kryven, David Theurel, Alessandro Rizzo, Paolo Santi, Joshua Tenenbaum, and Carlo Ratti. Vector-based pedestrian navigation in cities. *arXiv preprint arXiv :2103.07104*, 2021.

[3] Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Connection scan algorithm. *Journal of Experimental Algorithmics (JEA)*, 23 :1–56, 2018.

[4] Google. Find out how you stack up to new industry benchmarks for mobile page speed, 2018.

[5] Aaron Schild and Christian Sommer. On balanced separators in road networks. In *International Symposium on Experimental Algorithms*, pages 286–297. Springer, 2015.