État de l'art : outils de recherche du plus court chemin pour les réseaux routiers

Auteurs : Alexis MARIE, David MARCHÈS, Luca BANKOFSKI, Martial BROSTIN, Theo HELLER, Thomas MABILLE

1 INTRODUCTION

Dans le contexte spécifique de notre projet, axé notamment sur l'optimisation des trajets, la recherche de chemin le plus court devient un enjeu stratégique. **L'efficacité** du réseau routier est essentielle pour optimiser le temps de trajet des véhicules, réduire les embouteillages, et favoriser une mobilité plus fluide.

Le réseau routier mondial, confronté à une expansion continue du nombre de véhicules en circulation, est devenu un écosystème complexe marqué par des défis perpétuels. Actuellement, plus de 1,4 milliard de véhicules sillonnent les routes à travers le monde, indiquant une croissance significative au cours des dernières décennies. Cette augmentation exponentielle du trafic routier est malheureusement accompagnée d'une augmentation proportionnelle du nombre d'accidents et de décès liés à la circulation **[1]**.

Notre état de l'art vise à éclairer les aspects essentiels de la recherche de chemin le plus court dans le contexte de la circulation routière, offrant ainsi un fondement solide pour le développement de solutions innovantes et adaptées à notre projet.

Mots-clés : Problème du chemin le plus court, Dijkstra, Bellman-Ford, A*, Floyd-Warshall, D* Lite.

2 ALGORITHMES

2.1 ALGORITHMES CLASSIQUES

Algorithme de Dijkstra

L'algorithme de Dijkstra, conçu par Edsger W. Dijkstra en 1956, est une méthode cruciale pour trouver le chemin le plus court dans un graphe pondéré. Il est largement utilisé dans la planification de trajets et la conception de réseaux de communication.

Fonctionnant sur des graphes orientés ou non orientés avec des poids d'arêtes non négatifs, l'algorithme explore itérativement les sommets du graphe depuis le sommet **source**, calculant les distances minimales à chaque étape. La sélection du prochain sommet à traiter est optimisée par l'utilisation d'une file de priorité, et les informations essentielles, comme les distances minimales et les prédécesseurs, sont maintenues.

L'algorithme de Dijkstra assure l'optimalité des sous-chemins, garantissant que les distances minimales sont définies lors du traitement d'un sommet. Sa complexité temporelle dépend du choix de la structure de données utilisée pour la file de priorité, généralement en $O((V + E) \times logV)$.

- V est le nombre de sommets (ou nœuds) dans le graphe.

- E est le nombre d'arêtes (respectivement arcs) dans le graphe (respectivement graphe orienté).

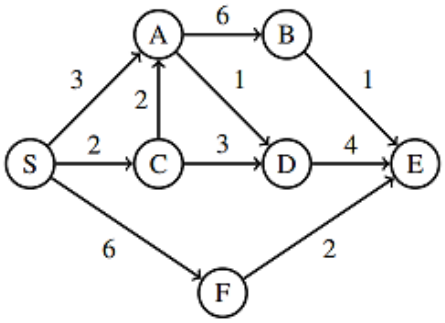


*Figure 1 - Graphe orienté à poids non négatifs **[4]**.*En résumé, l'algorithme de Dijkstra recherche le plus court chemin d'un nœud vers tous les nœuds.

Cependant, il ne fonctionne pas correctement avec des poids négatifs. Dans de tels cas, d'autres approches comme l'algorithme de Bellman-Ford sont préférables.

Algorithme de Bellman-Ford

L'algorithme de Bellman-Ford, développé par Alfonso Shimbel et Richard Bellman, est une méthode de recherche du plus court chemin dans un graphe pondéré depuis le sommet **source**. Sa particularité réside dans sa capacité à gérer des graphes avec des arêtes de poids **négatif**, ce qui le distingue de l'algorithme de Dijkstra.

À la différence de l'Algorithme de Dijkstra, qui trie les sommets par ordre de priorité, l'algorithme de Bellman-Ford procède étape par étape en examinant toutes les arêtes du graphe à chaque itération. À chaque étape, l'algorithme essaie de réduire les distances actuelles entre les sommets pour améliorer ses estimations des chemins les plus courts. Cette approche itérative permet à Bellman-Ford de repérer les cycles négatifs dans le graphe, ce qui est utile pour des applications comme la planification de trajets dans des réseaux complexes.

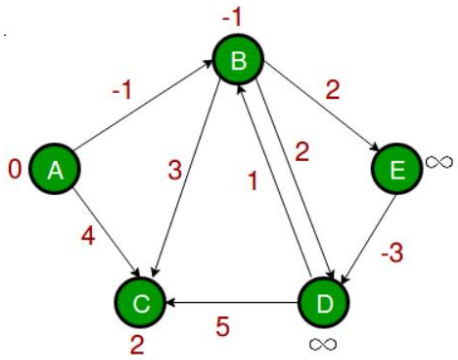


Figure 2 – Graphe orienté à poids positifs et négatifs.

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1

Figure 3 - Solution de sortie en utilisant l'algorithme de Bellman-Ford.

La complexité temporelle de l'algorithme est de l'ordre de $O(VE)$. Bien que cette complexité soit plus élevée que celle de l'Algorithme de Dijkstra dans des graphes sans poids négatifs, Bellman-Ford demeure une solution adaptée pour des contextes spécifiques.

En pratique, Bellman-Ford est souvent préféré lorsque la présence de poids négatifs est anticipée ou lorsque la détection de cycles négatifs est cruciale. Dans des graphes sans poids négatifs, l'Algorithme de Dijkstra conserve généralement une préférence en raison de sa complexité temporelle plus faible.

En résumé, l'algorithme de Bellman-Ford offre une robustesse particulière pour la recherche du plus court chemin, avec une flexibilité face aux poids négatifs et la capacité de détecter les cycles négatifs, en faisant un choix approprié dans des situations spécifiques.

Algorithme A*(A star)

A* est un algorithme de recherche du « meilleur premier », ce qui signifie qu'il

permet de trouver très rapidement un plus court chemin à partir d'un sommet de départ **[5][6]**. Outre sa vitesse, cet algorithme est réputé pour garantir une solution en sortie mais ne garantit pas que cette solution soit la plus optimale.

Ce que fait l'algorithme de recherche A*, c'est qu'à chaque étape, il sélectionne le sommet en fonction d'une valeur « f » qui est égale à la somme de « g » et « h ». À chaque étape, il sélectionne le nœud ayant le « f » le plus bas et le traite.

$$f(n) = g(n) + h(n)$$

- n est le nœud suivant sur le chemin.
- g(n) est le coût du chemin du nœud de départ jusqu'à n.
- h(n) est une fonction heuristique.

La fonction heuristique est spécifique au problème. Elle estime le coût du chemin le moins cher de n jusqu'à l'objectif.

Si la fonction heuristique ne surestime jamais le coût réel pour atteindre l'objectif, A* est assuré de renvoyer le chemin le moins coûteux du début à l'objectif.

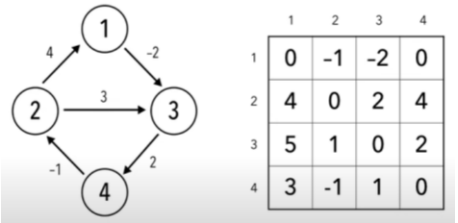
Algorithme de Floyd-Warshall

L'algorithme de Floyd-Warshall trouve tous les plus courts chemins entre **toutes** les paires de sommets dans un graphe orienté **[7]**.

Son objectif est de déterminer la distance minimale entre chaque paire de sommets dans le graphe, même en présence d'arcs négativement pondérés. L'algorithme fonctionne en initialisant une matrice des distances entre les paires de sommets, puis en explorant tous les sommets intermédiaires possibles pour ajuster les distances.

Pour chaque paire de sommets, l'algorithme compare la distance directe à la distance via un sommet intermédiaire, et s'il trouve un chemin plus court, il met à jour la distance. Cela se répète pour tous les sommets intermédiaires, conduisant finalement à une matrice résultante qui contient les plus courts chemins entre toutes les paires de sommets.

Bien que son temps d'exécution soit cubique (Complexité temporelle en O(V^3)), l'algorithme de Floyd-Warshall est robuste et peut être appliqué à divers types de graphes, tout en permettant de détecter la présence de cycles de poids négatif.



	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0

Figure 4 - Graphe orienté à poids positifs et négatifs et sa matrice de sortie.

2.2 Algorithme Specifique au Contexte

D* Lite

L'algorithme D* Lite commence par une phase d'initialisation des coûts. Ces coûts sont basés sur une fonction d'estimation qui évalue le coût probable du déplacement entre deux points.

Ensuite, l'algorithme effectue une propagation inverse. Cela signifie qu'il met à jour les coûts des sommets en partant du point d'arrivée vers le point de départ. Cette étape permet à l'algorithme d'ajuster les coûts en fonction des obstacles présents dans l'environnement, et ainsi, de mieux anticiper les chemins optimaux.

Une fois la propagation inverse terminée, l'algorithme procède à une propagation directe. Cette étape ajuste à nouveau les coûts des sommets, mais cette fois en se basant sur de nouvelles informations obtenues pendant la propagation inverse. Ainsi, l'algorithme tient compte des obstacles détectés et peut rapidement s'adapter à des changements dans l'environnement.

L'étape cruciale suivante est la recherche de chemin. L'algorithme utilise les coûts mis à jour pour déterminer le chemin optimal entre le point de départ et le point d'arrivée. Il sélectionne le chemin qui minimise la somme des coûts des sommets le long de la trajectoire, garantissant ainsi une solution optimale.

L'une des caractéristiques distinctives du D* Lite est sa capacité à réagir aux changements dynamiques de l'environnement **[8]**. Si de nouvelles informations sont introduites, telles que la présence d'un nouvel obstacle, l'algorithme peut ajuster le chemin de manière incrémentielle sans recalculer l'intégralité du chemin, ce qui le rend particulièrement adapté aux environnements dynamiques.

3 COMPARAISONS

Critères	**Dijsktra**	**Bellman-Ford**	**A***	**Floyd-Warshall**	**D* Lite**
Type	Chemin le plus court	Chemin le plus court	Chemin le plus court avec heuristique	Chemins les plus courts	Chemin le plus court incrémental
Objectif	Trouver le plus court chemin d'un seul nœud à tous les autres	Trouver le plus court chemin d'un seul nœud à tous les autres	Trouver le plus court chemin avec une estimation heuristique pour guider la recherche	Trouver tous les chemins les plus courts entre tous les nœuds	Trouver le plus court chemin dans des graphes dynamiques ou en évolution
Complexité temporelle	O((V + E) * log(V))	O(VE)	O(E)	O(V^3)	Variable, dépend des perturbations
Complexité spatiale	O(V + E)	O(V)	O(V)	O(V^2)	O(V)
Gestion des obstacles dynamiques	Non	Non	Non	Oui	Oui
Type d'arêtes acceptées	Positives	Positives et négatives	Positives	Positives et négatives	Positives et négatives

D* Lite se distingue comme un choix privilégié pour la génération de plans de navigation routière en raison de sa gestion efficace des obstacles dynamiques, et de son adaptation remarquable aux changements de carte.

Contrairement à d'autres algorithmes tels que Dijkstra, Bellman-Ford et A*, D* Lite excelle dans la prise en compte des obstacles dynamiques, ajustant le chemin en temps réel face aux variations des conditions du trafic. Dans l'ensemble, ces caractéristiques font de D* Lite le choix optimal pour la génération de plans de navigation dans des scènes routières dynamiques.

Cependant, sa complexité temporelle variable peut entraîner des variations de performances en fonction des changements fréquents dans l'environnement, et dans des situations où la réactivité immédiate est cruciale, des algorithmes plus simples mais moins flexibles, tels que A* pourraient être préférés **[9]**.

4 ETUDE DE L'EXISTANT

Google Maps

Google Maps est un service de cartographie et de recherche d'itinéraire appartenant à l'entreprise mondiale Google.

Disponible sur le site web et en application, Google Maps permet de se déplacer d'un point à un autre de la carte en proposant une vue en plan cartographique, en plan satellite et une possibilité d'explorer le trajet en direct grâce à l'affichage Street View. Il est la référence aujourd'hui en termes de déplacements en voitures ou autres transports en affichant également les différentes commodités (les différents lieux tels que des hôtels, les restaurants, etc).

Google Maps utilise principalement deux algorithmes pour définir la trajectoire optimale à suivre lors d'un trajet : l'Algorithme de Dijkstra et l'Algorithme A*(A star) **[6][10]**.

Google Maps permet également d'adapter sa conduite en direct en permettant de choisir un itinéraire avec ou sans péages, avec ou sans autoroutes, et permet d'afficher les accidents, travaux et ralentissements sur notre chemin.

Waze

Waze est une application mobile d'assistant d'aide à la conduite et d'assistance à la navigation. Elle a été rachetée par la société Google en 2013. Elle est similaire à Google Maps mais propose moins d'options. Elle diffère notamment car elle est couplée à une cartographie modifiable en temps réel par les utilisateurs. Les utilisateurs peuvent notamment signaler un accident, des ralentissements, etc.

La génération d'itinéraires concernant le trajet est basée sur un seul algorithme : l'Algorithme A*(A star).

Waze propose une fonctionnalité intéressante : changement automatique de l'itinéraire si de fortes perturbations apparaissent sur celui-ci et s'il existe un autre itinéraire.

TomTom

TomTom est un éditeur de logiciels de planification d'itinéraires et un fabricant de systèmes de navigation par positionnement satellite mobiles ou embarqué dans certains véhicules.

Un inconvénient du système de navigation embarqué est la mise à jour des cartes. Les modèles mobiles ont la possibilité de faire une mise à jour via un service web.

Tomtom utilise l'Algorithme A*(A star)pour définir l'itinéraire à suivre afin d'arriver à destination **[11]**.

Depuis 2013 TomTom se lance également sur le marché des montres de sport avec différents modèles pour les coureurs, les nageurs, les cyclistes, et un an plus tard pour les golfeurs. Ils font également partie des entreprises qui travaillent sur la voiture autonome.

Apple Plans

Lancée en 2007 en utilisant les données de **Erreur ! Source du renvoi introuvable.** et mise à jour en 2012 sans utiliser les données de **Erreur ! Source du renvoi introuvable.**, Apple plans est une application de cartographie en ligne développée par Apple dans le but de remplacer l'application **Erreur ! Source du renvoi introuvable.** qui était l'application par défaut.

Il s'agit de l'application de cartographie par défaut sur tous les produits de l'écosystème Apple. Il fournit la fonctionnalité "flyover" équivalente à Google Earth ou Google Street View, permettant de visiter les lieux via l'application.

OpenStreetMap

Lancé en 2004, OpenStreetMap (OSM) est un projet de cartographie collaboratif et open source qui permet aux individus du monde entier de créer, modifier et partager des données géographiques.

Les utilisateurs utilisent des appareils GPS, des images aériennes et d'autres sources de données pour contribuer à la précision et à la richesse d'OpenStreetMap. La plateforme est particulièrement utile dans les zones où les services de cartographie commerciaux peuvent être limités ou obsolètes, permettant la création de cartes détaillées et à jour dans des régions éloignées ou en développement.

OpenStreetMap a trouvé des applications dans divers secteurs, notamment l'urbanisme, la réponse aux catastrophes, l'aide humanitaire et le tourisme. Il sert de base à de nombreuses applications tierces et ses données sont largement utilisées par les développeurs, les chercheurs et les organisations à des fins diverses.

La communauté OpenStreetMap met l'accent sur l'accessibilité et la démocratisation de l'information géographique, ce qui en fait un outil puissant pour les individus et les organisations à la recherche de données cartographiques fiables et ouvertes.

5 REFERENCES

[1] Organisation mondiale de la santé. 2023. « Road traffic injuries », OMS. Retrieved January, 2024, from https://www.who.int/fr/news-room/fact-sheets/detail/road-traffic-injuries

[2] Représentation de la Commission européenne en France. 2023. « La Commission publie les chiffres de 2022 et les estimations de 2023 sur le nombre de victimes de la route », UE. Retrieved January 2024, from https://france.representation.ec.europa.eu/informations/la-commission-publie-les-chiffres-de-2022-et-les-estimations-de-2023-sur-le-nombre-de-victimes-de-la-2023-10-19_fr

[3] Transitions Énergies. « Combien y a-t-il de voitures dans le monde? » Retrieved January 2024, from https://www.transitionsenergies.com/combien-voitures-monde/#:~:text=En%20prolongeant%20les%20estimations%20faites,pr%C3%A8s%20de%201%2C4%20milliard%E2%80%A6

[4] Dijkstra's Shortest Path Algorithm. Brilliant.org. Retrieved January 2024, from https://brilliant.org/wiki/dijkstras-short-path-finder/

[5] Zeng, W., Church, R.L. 2009. « Finding Shortest Paths on Real Road Networks: The Case for A*. », International Journal of Geographical Information Science.

[6] Goldberg, A.V., Harrelson, C. 2005. « Computing the Shortest Path: A* Search Meets Graph Theory », Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms.

[7] Eppstein, D. 1998. « Finding the k Shortest Paths », SIAM Journal on Computing.

[8] Stolfi, J., de Figueiredo, L.H. 1995. « An Introduction to the D* Algorithm for Robot Navigation », ACM SIGACT News.

[9] Comparing the shortest path algorithms. Hyperskill.org. https://hyperskill.org/learn/step/15436

[10] Does Apple Maps use Dijkstra's algorithm? Studycountry.com. https://www.studycountry.com/wiki/does-apple-maps-use-dijkstras-algorithm

[11] Batty, M. 2013. « The New Science of Cities », The MIT Press, Cambridge, MA.

[12] https://www.tomtom.com/solutions/autonomous-driving-solutions/