Comte rendu des derniers TP Algorithmique avancée

Ramzy CHIBANI

December 2024

1 Objectifs du TP :

Ce TP vise à implémenter et comparer les performances des algorithmes de Dijkstra et A* pour trouver le plus court chemin dans un graphe représentant une carte en 2D. Les deux algorithmes ont été intégrés dans un code fourni, modifié pour répondre aux consignes. Les résultats sont ensuite comparés pour évaluer leur pertinence dans divers contextes.

2 Approche pour résoudre le TP :

2.1 Lecture et représentation du graphe :

- Le fichier de la carte est lu ligne par ligne pour récupérer les métadonnées (dimensions, types de cases, etc.).
- J'ai utilisé la classe **Graph** pour créer une structure de graphe pondéré, où chaque sommet représente une case de la carte, et les arêtes sont pondérées en fonction des temps de déplacement entre les cases (prise en compte des voisins directs et diagonaux).
- Une distinction est faite entre les différents types de terrains grâce aux temps de traversée et aux couleurs associées.

2.2 Implémentation de Dijkstra :

- Cet algorithme trouve le chemin le plus court en explorant les sommets avec la plus petite distance temporaire **timeFromSource**.
- Les voisins sont mis à jour pour refléter une distance potentiellement plus courte en passant par le sommet courant. Une fois un sommet visité, il n'est plus revisité.

2.3 Implémentation de A* :

- A* reprend la logique de Dijkstra mais y ajoute une heuristique.
- J'ai choisi Manhattan comme heuristique par défaut, car elle est bien adaptée aux grilles et garantit l'optimalité dans ce contexte (elle ne surestime jamais la distance).
- Une version utilisant la distance euclidienne comme heuristique a également été implémentée et laissée en commentaire, je l'ai essayé et il n'y a pas une grande difference avec la version Manhattan, mis à part dans le nombre de sommet parcouru qui différe un peu.

2.4 Réponse aux questions :

2.4.1 Pourquoi utiliser Manhattan ?

- La distance de Manhattan, qui correspond à la somme des différences absolues des coordonnées, est plus rapide à calculer et plus pertinente pour des déplacements en grille (haut/bas, gauche/droite).
- Par comparaison, l'heuristique euclidienne peut sembler plus précise, mais elle n'est pas significative pour des déplacements limités aux axes et diagonales.

2.4.2 Observations sur les performances :

J'ai representé dans ce tableau les differents résultats que j'ai eu en appliquant l'algorithme sur trois labyrinthes que j'ai généré grace à un script en python (le sccript a été fait à l'aide d'une IA).

Fichier graphe utilisé	Dijkstra	A*
graphe.txt	Nombre de sommet parcouru : 4156 Temps total de parcours : 303.838...	Nombre de sommet parcouru : 4106 Temps total parcours : 303.838...
graphe2.txt	Nombre de sommet parcouru : 3973 Temps total de parcours : 1434.397...	Nombre de sommet parcouru : 3877 Temps total de parcours : 1434.397...
graph3.txt	Nombre de sommet parcouru : 4069 Temps total de parcours : 1458.815...	Nombre de sommet parcouru : 3829 Temps total de parcours : 1458.815...

2.4.3 Comparaison Dijkstra et A* :

- Dijkstra explore un plus grand nombre de sommets car il ne tient pas compte d'une estimation de la distance jusqu'à l'objectif.
- A* est plus efficace grâce à son heuristique, ce qui réduit significativement le nombre de sommets explorés, surtout sur des grandes cartes.

3 Cas réels d'utilisation des algorithmes de Dijkstra et A* :

3.1 Cas d'utilisation de l'algorithme de Dijkstra :

- Tous les chemins doivent être calculés : Il calcule les distances minimales depuis un point de départ à tous les autres nœuds. Cela le rend pertinent dans les systèmes nécessitant une carte complète des coûts.
- Graphes statiques : Il fonctionne efficacement lorsque le graphe ne change pas fréquemment.
- Absence de contraintes directionnelles : L'algorithme n'utilise pas d'heuristique, ce qui le rend idéal pour des problèmes où il faut explorer tout le graphe de manière exhaustive.

Exemples réels :

- Planification de réseaux urbains : Déterminer les distances les plus courtes pour des systèmes comme les réseaux électriques, l'adduction d'eau ou les réseaux de gaz, où tous les nœuds doivent être connectés.
- Systèmes de navigation maritime ou aérien : Calcul des routes minimales entre plusieurs points (par exemple, trouver le chemin optimal pour couvrir plusieurs ports ou aéroports).

3.2 Cas d'utilisation de l'algorithme A* :

- Une cible spécifique est définie : Grâce à l'heuristique, il se concentre sur les zones pertinentes en direction de l'objectif
- Performance critique : A* explore moins de sommets que Dijkstra, ce qui le rend plus rapide pour des problèmes de grande taille.
- Graphes dynamiques : Son utilisation d'heuristiques permet de s'adapter à des situations où les coûts ou les structures peuvent changer fréquemment.

Exemples réels :

- Navigation GPS : Trouver le chemin le plus court entre deux points dans un système routier.

- Logistique et livraison : Optimisation des itinéraires pour les livraisons ou les déplacements de flottes de véhicules.
- Jeux vidéo : Recherche de chemin pour des personnages ou des entités, particulièrement lorsqu'il y a un objectif à atteindre (par exemple, aller vers un endroit de la map en évitant des obstacles).

4 Difficultés rencontrées :

- Gestion des voisins diagonaux : Le calcul des poids des arêtes diagonales, nécessitant la moyenne des coûts combinée à une distance géométrique, était complexe à intégrer correctement.
- Validation des heuristiques : Bien que Manhattan soit intuitivement adapté, vérifier qu'elle respecte les propriétés d'une heuristique admissible a nécessité pas mal de tests.

5 Sources utilisées :

- Documentation des algorithmes de Dijkstra et A* sur Wikipedia et Wikipedia A*.
- Explications sur l'heuristique de Manhattan issues de discussions sur le forum StackOverflow et GeeksforGeeks, et quelques correction de mon code à l'aide d'une IA.

6 Conclusions :

- A* est clairement supérieur à Dijkstra pour ce type de problème, notamment avec l'heuristique Manhattan. Il offre des performances comparables (en termes de temps) tout en explorant moins de sommets, comme on l'a vu avec les differents exemples de test.
- Ce TP m'a permis de mieux comprendre l'importance des heuristiques dans la résolution de problèmes par des algorithmes.