ESIGELEC – Projet S8

Synthèse d'un article scientifique :

Comparaison d'algorithmes de plus courts chemins sur des graphes routiers de grande taille

Chaque personne dans le groupe a choisi une orientation différente afin de regrouper un maximum d'idées de différentes recherches.
J'ai choisi pour ma part de comparer les différents algorithmes de plus courts chemins.
J'ai donc choisi cette étude qui est une comparaison des différents algorithmes utilisés pour des graphes routiers afin de répondre à notre problématique.

Problématiques posées

- Tout d'abord, l'étude distingue 3 méthodes de calculs du plus court chemin :
  - Problème A : D'un point A vers un point B
  - Problème B : D'un point A vers n'importe quel point
  - Problème C : De n'importe quel point vers n'importe quel point

  Un algorithme pouvant faire le problème C peut faire le B.
  Un algorithme pouvant faire le problème B peut aussi faire le A par simplification.
- L'étude traite des cas de grande taille : Les calculs sont appliqués sur des carte de 5000 à 30000 sommets.
- L'étude est déjà restreinte à 9 algorithmes, considérés comme efficace pour des rechercher du plus court chemin
  - Dijkstra sans tas
  - Dijkstra avec tas
  - Dijkstra avec tas de Fibonacci
  - Bucket 1
  - Bucket L
  - Ahuja
  - FIFO
  - Esopo
  - Thresh-X

Approche

La comparaison se fait en plusieurs parties.

Le premier test se fait sur des graphes constitués d'hexagones parfaits. Ce test est lancé avec différentes tailles (N = 1000, 5000, 10000 et 15000). Pour chaque taille et pour chaque algorithme, une moyenne du temps de calcul sur 20 lancés est fait. Ensuite l'algorithme de Dijkstra avec tas est pris en référence et tous les temps de calcul sont exprimés en fonction de celui-ci.
Tous les algorithmes ont bien évidemment un temps de calcul plus importants avec un nombre de nœuds importants, cependant ce premier test démontre pour ce cas, l'efficacité des algorithmes à buckets (BUCK1 et BUCKL) qui reste plus rapide que les autres.

Le second test se base sur 5000 nœuds uniquement, mais le graphe est aléatoire avec des routes entre les nœuds égaux à d, qui prends les valeurs suivantes : 3, 10, 15, 20 ou 30 selon le test (3 étant la valeur pour un graphe hexagonnal). Le graphe est construit avec pour chaque nœud un nombre aléatoire de routes mais la moyenne des routes des nœuds est égale à d.
L'algorithme de Dijkstra reste lent, mais ne varie que très peu. Les autres algorithmes se dégradent cependant plus ou moins rapidement. L'algorithme à Bucket (BUCK1) reste le meilleur algorithme. L'algorithme BUCKL lui est bien trop affecté. L'algorithme Ahuja et

la version de Dijkstra avec tas de Fibonacci deviennent un peu compétitif mais reste moins rapide que BUCK1.

Deux autres tests sont réalisés mais sans l'ensemble des algorithmes étudiés.

Le premier fait varier la valeur des coûts des routes. Les coûts pour les tests précédents étaient compris en 1 et 1000, ici on prend des valeurs plus importantes. L'algorithme de Dijkstra n'ait pas influencé par cette modification, contrairement aux algorithmes à buckets. Ici on chercher donc à voir leurs limites.
L'algorithme BUCK1 perd devant Dijkstra pour une valeur maximum des coûts supérieure à 7000 environ pour 5000 nœuds et environ 12000 pour 10000 nœuds.

Le dernier test est utilisé pour comparé l'algorithme de Dijkstra avec l'algorithme de Sedgewick-Vitter. C'est un algorithme qui n'était pas comparé au départ car il est vraiment efficace que pour le problème A pour des graphes réels.

Apports éventuels à notre problématique

- L'algorithme de BUCK1 est considéré par l'étude comme étant le plus performant. Son soucis principal est qu'il est affecté par des coûts de routes très élevés. Un avantage mis en avant dans l 'étude est qu'il est plutôt simple à implémenter.
- Le problème C permet de construire un distancier, c'est-à-dire une matrice NxN (N = nombre de nœuds). Ce distancier permet de trouver le chemin le plus court entre deux points simplement en regardant dans la matrice. Le seul soucis de ce distancier étant la taille en mémoire.
- L'algorithme de Sedgewick-Vitter est noté comme étant très rapide pour le problème A de façon ponctuelle mais est lent quand il s'agit des autres problèmes.

Référence :
C. Prins, Comparaison d'algorithmes de plus courts chemins sur des graphes routiers de grande taille, *RAIRO Recherche Opérationnelle*