Routage et reroutage dans un réseau dynamique géographique(*)

Moustafa NAKECHBANDI, Jean-Yves COLIN, Ahmed Salem OULD CHEIKH,
LITIS, RIIC, Université du Havre

moustafa.nakechbandi@univ-lehavre.fr
jean-yves.colin@univ-lehavre.fr
ahdsalem@univ-nkc.mr

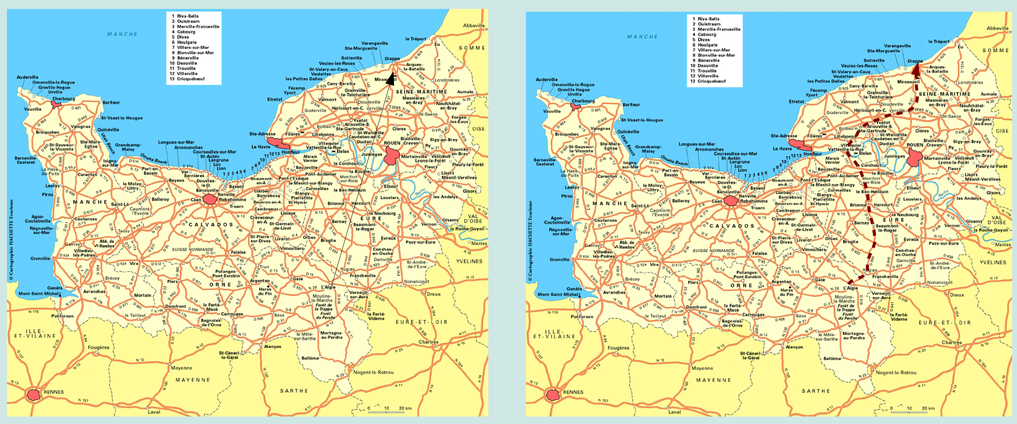
Introduction

Si on veut aller rapidement d'un point éloigné à un autre, on consulte une carte

et on cherche le chemin le plus court en connaissant la durée de trajet de chaque route

mais si un passage passe par un pont levant ? ou par un pont comme le pont de Tancarville qui est parfois chargé ? sa traversée a une durée qui peut varier fortement et il peut être intéressant de passer par un autre pont comme le pont de Brotonne. Pour les logiciels actuels, ces ponts sont toujours baissés et les routes sont supposées libres par défaut (mais quiconque s'est déjà retrouvé bloqué devant sait que ce n'est pas vrai !)

Les applications de calcul d'itinéraires, type Mappy, TomTom ou autre sur un PC ou un GPS, se limitent malheureusement à les calculer dans des conditions précises et à les recalculer de temps à autres, ce qui peut prendre du temps.



Modélisation du problème par un graphe (Graphe Faiblement Dynamique)

Définition : On considère un graphe orienté valué *G(V, E)* sans cycle (DAG) avec *V* ensemble des nœuds ou sommets et *E* ensemble des arcs. Les arcs sont valués par des valeurs positives qui sont stables et connues, sauf pour un arc de valeur *x* et un autre arc de valeur *y* connus non stables. Un tel graphe sera appelé graphe faiblement dynamique.

Le problème à résoudre : On cherche à trouver un plus court chemin allant d'un point de départ à tout autre sommet du graphe pour toutes les valeurs possibles des arcs variables x et y.

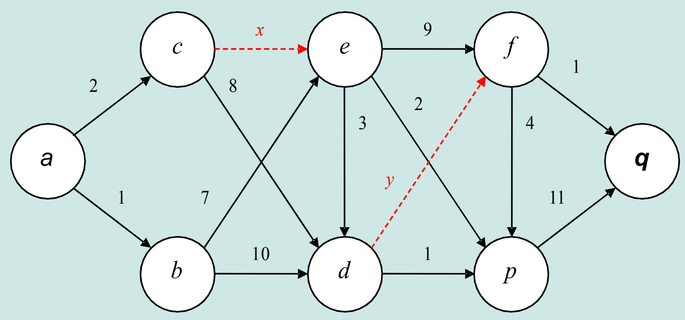


Figure 1 : Graphe faiblement dynamique, avec deux arcs variables en rouge pointillé sur le graphe.

Algorithme

On sait calculer (par l'algorithme classique de Dijkstra par exemple) le meilleur chemin d'un point vers tous les autres, s'il n'y a pas d'arcs variables.

Dans ce cas plus complexe, on va l'utiliser pour obtenir jusqu'à 4 chemins qui seront en compétition pour le titre de "plus court chemin de tous"
- le plus court chemin qui ne passe par aucun arc variable (voir Table 1)
- le plus court chemin qui passe uniquement par le 1er arc variable (voir Table 2, 1ère ligne)
- le plus court chemin qui passe uniquement par le 2ème arc variable (voir Table 2, 2ème ligne)
- le plus court chemin qui passe par les deux arcs variables (voir Table3)

Toutefois, savoir quel est le meilleur à un moment donné dépend des durées actuelles des arcs variables (voir Table 4)

Heureusement, un des résultats de ce projet est d'avoir montré qu'on peut calculer dès le départ les conditions exactes qui déterminent quel chemin parmi ces quatre sera le meilleur à un moment donné.

Par exemple, dans la table 4 dernière colonne (pour aller de *a* vers *q*), si les deux arcs variables *x, y* sont très grands (*x= 10, y =13*) (*18 < 12+x < 11+ y < 6+ x+ y*) alors le premier chemin, qui ne passe pas par aucun arc variable, dont la longueur = 18 sera le meilleur. Par centre si (*x=3, y=3)* alors le dernier chemin qui passe par les deux arcs variables dont la longueur = *6+ x+ y = 12* sera le meilleur car (*6+ x+ y < 11+ y < 12+x < 18*).

Illustration de l'algorithme sur le graphe de la Figure 1 :

Table 1 : Longueur des plus courts chemins ne passant par aucun arc variable partant de *a :*

Destination j	**a** (origine)	*b*	*c*	*e*	*d*	*f*	*p*	*q*
$ds^0(a, j)$	*0*	*1*	*2*	*8*	*10*	*17*	*10*	*18*

Table 2 : Longueur des plus courts chemins passant uniquement par l'arc *x ou par l'arc* et partant de *a* :

Destination j	**a**	*b*	*c*	*e*	*d*	*f*	*p*	*q*
$ds^0(i, c) + x + ds^1(e, j)$	-	-	-	2+x	2+ x+3	2+ x+9	2+ x+2	2+x+10
$ds^0(i, d) + y + ds^1(f, j)$	-	-	-	-	-	10+ y	10+ y+4	10+ y+1

Table 3 : Longueur des plus courts chemins passant par les deux arcs variables *x* et *y* et partant de *a* :

Destination j	*a*	*b*	*c*	*e*	*d*	*f*	*p*	*q*
$ds^0(i, c) + x + ds^1(e, d) + y + ds^2(f, j)$	-	-	-	-	-	2+x+3+ y	2+x+3+y+4	2+x+3+ y +1

Table 4 : le plus court chemin dépend en fait des valeurs actuelles de *x* et *y* :

Destination j	**a**	*b*	*c*	*e*	d	*f*	*p*	*q*
Plus courts chemins partant de *a et* paramétrés par les valeurs x et *y*	*0*	*1*	*2*	*Min (8, 2+x)*	*Min (10, 5+x)*	*Min (17, 11+x, 10+ y, 5+x+ y)*	*Min (10, 4+x, 14+ y, 9+ x+ y)*	*Min (18, 12+x, 11+ y, 6+ x+ y)*

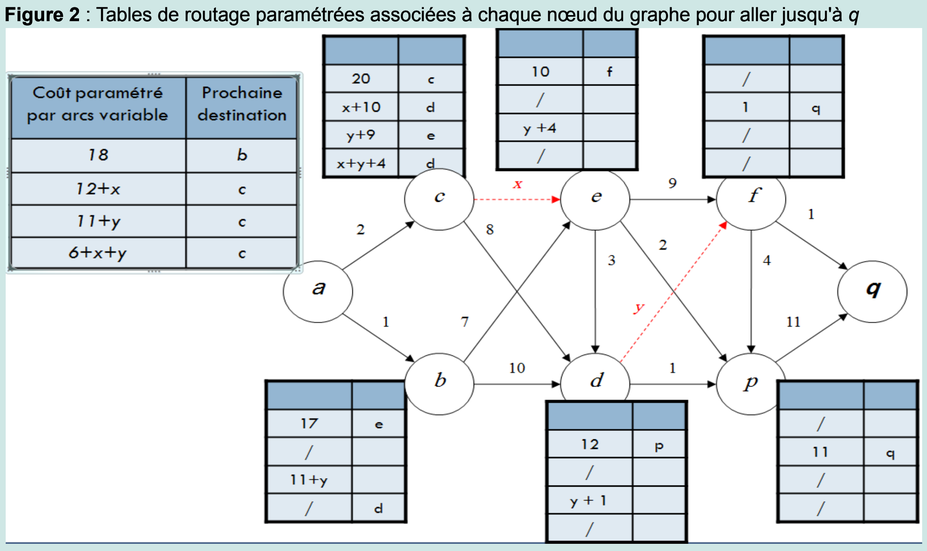
Exemple d'application à un problème de transport routier

En reprenant le graphe de la Figure 1 et en supposant qu'un camion part de *la ville a* pour aller à la ville *q*.

Table 5 : plus courts chemins paramétrés par x et y de tout sommet jusqu'à la destination *q*

sommet	*a*	*b*	c	e	d	*f*	*p*	q
Plus courts chemins vers q en fonction de *x et y*	*Min(18, 12+x, 11+y, 6+x+y)*	*Min(17, 11+y)*	*Min(20, 10+x, 9+y)*	*Min(10, 4+y)*	*Min(12, 1+y)*	*1*	*11*	*0*

Figure 2 : Tables de routage paramétrées associées à chaque nœud du graphe pour aller jusqu'à *q*



Application à un problème de transport routier

Avec les résultats obtenus précédement , On peut calculer une table de plus court chemin parametré (voir table 5). Si les valeurs de *x* et *y* changent brutalement, on peut éviter de recalculer entièrement un nouveau chemin.

En effet, en surveillant en permanence les quelques points de traversée variables de la région, notre algorithme et partant de la table 5 calcule des tables de routage (voir Figure 2) que le camion consultera à chaque intersection importante. Chaque point du graphe aura une table de routage paramétrée spécifique précalculée et qui ne change pas.

Ces tables de routage diront directement au véhicule quelle est la meilleure prochaine destination en fonction des durées actuelles de traversée des arcs variables *x* et *y.*

Conclusions

- Une fois toutes les tables paramétrées calculées, il est facile de déterminer directement vers quel lieu voisin aller pour finalement atteindre la destination, même si les arcs variables *x* et *y* changent plusieurs fois en cours de trajet, et pour n'importe quelles valeurs. Aucun recalcul de plus courts chemins n'est nécessaire. Notre méthode, dans ce cas de graphe faiblement dynamique, est plus rapide que les méthodes classiques de routage dans les réseaux informatiques (RIP par exemple) ou dans les problèmes de transports logistiques.
- Nos recherches actuelles visent le calcul de plus longs chemins pour les problèmes d'ordonnancement (PERT, graphes Potentiel-Tache).

References

1. H.R. Bajgan, and R.Z. Farahani. "Using colony system and neighborhood search for dynamic vehicle routing problem". American Journal of Operational Research, vol. 2, no.4, pp. 31-44, 2012.
2. S. Balev, F. Guinand, Y. Pigné, "Maintaining Shortest Paths in Dynamic Graphs". In proceedings of the International Conference on Non-Convex Programming: Local and Global Approaches Theory, Algorithms and Applications (NCP'O7). 2007, December 17-21, Rouen, 2007.
3. D. Fulkerson, Expected critical path lengths in PERT networks. Operations Research 10 808 - 817, 1962.
4. M. Nakechbandi,J-Y Colin, A. S. Ould Cheikh, "Routing and Rerouting in Territorial Systems Modeled by Weakly Dynamics Graphs " STaN'2013 within ECCS'13, September 16-20, 2013, Barcelona, Spain(2013).
5. A. Orda and R. Rom. "Shortest-path and minimum-delay algorithms in networks with time dependent edge length". J. ACM, 37(3), 1990.

(*) Recherche soutenue par la Région Haute-Normandie : Projet APLOG "Amélioration de la Performance Logistique Globale"