REVUE FRANÇAISE D'AUTOMATIQUE, INFORMATIQUE, RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

G. FONTAN

Sur les performances d'algorithmes de recherche de chemins minimaux dans les graphes clairsemés

Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle, tome 8, n° V2 (1974), p. 31-37

<http://www.numdam.org/item?id=RO_1974__8_2_31_0>

R.A.I.R.O.
(8e année, mai 1974, V-2, p. 31 à 37)

SUR LES PERFORMANCES D'ALGORITHMES DE RECHERCHE DE CHEMINS MINIMAUX DANS LES GRAPHES CLAIRSEMES

par G. FONTAN (1)

Résumé. — *L'auteur établit une classification des principaux algorithmes de recherche de chemins minimaux dans un graphe. Lorsque des comparaisons analytiques n'ont pu être établies, il compare expérimentalement le meilleur de chaque classe pour pouvoir conclure quant à leurs performances respectives.*

I. INTRODUCTION

De nombreux articles sur les comparaisons de méthodes de recherche de chemins minimaux dans un graphe ont déjà été publiés [1], [2]. Depuis, de nouvelles méthodes ont été découvertes, elles méritent un examen attentif de leurs possibilités. D'autre part, ces comparaisons ne considéraient généralement que des graphes complets alors que nous sommes surtout intéressés par les réseaux grands et clairsemés tels qu'on les trouve dans les problèmes de Trafic et dans la résolution par la théorie des flots de problèmes en Recherche Opérationnelle. Il est donc indispensable de faire le point sur les méthodes les plus efficientes qui nous sont proposées.

En nous appuyant sur certaines publications, nous ferons une classification analytique des algorithmes lorsque cela sera possible. Dans les cas où ce genre de comparaison sera impossible nous procéderons à une étude expérimentale.

(1) Laboratoire d'automatique et d'analyse des systèmes, Toulouse.

II. CLASSIFICATION DES METHODES

Nous établirons trois classes, dans lesquelles nous regrouperons les principales méthodes existantes :

1re classe : Construction d'arborescences.

Les principales méthodes sont dues à Dijkstra, Moore, Dantzig. Ces méthodes consistent à marquer progressivement les sommets du graphe en calculant au fur et à mesure la longueur du chemin minimum qui permet de les atteindre à partir de l'origine. A chaque étape, nous avons donc une plus courte distance [3], [4].

2e classe : Méthodes itératives.

Nous regrouperons dans cette classe les méthodes dues à Bellman, Ford et Yen.

On associe à chaque sommet k, un nombre λ_k. On modifie ces nombres lorsque on trouve un arc de longueur l_{ij} tel que $\lambda_j - \lambda_i < l_{ij}$ en posant $\lambda_j = \lambda_i + l_{ij}$. Lorsque on ne trouve plus de tels arcs on arrête la procédure [5], [6].

3e classe : Méthodes par élimination.

Les deux principaux auteurs sont Carré et Roy. La recherche des plus courts chemins dans un graphe se réduit à la résolution de l'équation matricielle : $Y = AY + B$.

Les auteurs cités plus haut résolvent cette équation à l'aide de méthodes connues en algèbre linéaire : Méthodes de Gauss et Jordan [1], [7], [8].

Dans une même classe, certaines méthodes ont pu être comparées analytiquement :

a) Les arborescences construites par les algorithmes de Dantzig et Dijkstra sont identiques mais Dijkstra demande moins de comparaisons et sera plus efficient [1], [2], [8], [9].

b) Les méthodes de Bellman, Ford et Yen ont pu être décrites matriciellement et reliées aux méthodes classiques d'algèbre linéaire. Ces méthodes demandent un même travail par itération. Mais la convergence des algorithmes est différente. Il a été démontré que l'algorithme de Ford (méthode de Gauss-Seidel) est toujours au moins aussi bon que l'algorithme de Bellman (méthode de Jacobi). De la même façon il peut être démontré que la méthode de Yen (Gauss-Seidel symétrique) est au moins aussi bonne que la méthode de Ford [5], [6], [8].

c) Dans le cas des algorithmes de la 3e classe, si l'on cherche toutes les distances et si le graphe est complet, toutes les méthodes directes sont équivalentes. Pour un graphe peu dense (comme cela sera le cas pour nous) et un

nombre de chemins inférieurs au nombre de nœuds, la méthode de Carré (Gauss généralisée) sera la plus efficiente [1], [8].

En conclusion, nous pouvons dire que à l'intérieur de chaque classe, les meilleures méthodes sont :

Classe 1 : Dijkstra.

Classe 2 : Yen.

Classe 3 : Carré.

Nous ne pourrons départager ces algorithmes qu'avec des résultats expérimentaux tirés de l'étude des graphes qui nous intéressent.

III. INFLUENCE DE LA TOPOLOGIE ET DE LA SOLUTION

Avant de passer à l'étude expérimentale proprement dite, nous pouvons faire quelques remarques préalables.

* *Pour la classe 1 :* Si n est le nombre de nœuds du graphe, nous ferons toujours n pas pour trouver l'arborescence complète. Au $k$ième pas, nous aurons l'arborescence des plus courts chemins entre k nœuds. Le travail nécessaire, ainsi que l'arborescence dépend des valeurs numériques et de la taille du problème.

* *Pour la classe 2 :* Aucune dépendance topologique, mais le nombre d'itérations à effectuer est conditionné par la solution finale [6].

* *Pour la classe 3 :* Il y a une forte dépendance topologique. En effet, cela détermine le nombre de termes produits au cours de la phase élimination. Les valeurs numériques n'ont ici aucune influence sur le travail à effectuer.

Nous pourrions donc construire des problèmes qui rendent telle ou telle méthode plus avantageuse. La comparaison que nous faisons n'est pas absolue, mais nous pourrons conclure sur un type de problème.

IV. DETAILS TECHNIQUES DE L'APPLICATION DE CES METHODES

Les algorithmes que nous devons comparer ont été programmés en FORTRAN IV sur IBM 370-165. Ils ont été construits le mieux possible [10] de manière à tirer parti au maximum de la faible densité des graphes étudiés. Ceci implique que nous ne faisons pas de reconnaissance d'arcs inexistants lors des calculs.

— Pour Dijkstra, seul le 1er élément non-nul de chaque ligne de la matrice du graphe sera repéré, les autres seront reliés par un chaînage.

— Pour Yen, on ne travaille que sur des demi-matrices triangulaires. On ne repère que le premier élément non-nul de chaque ligne des demi-matrices.

— Pour Carré, nous avons créé un système de pointeurs [11] (double chaînage) qui permet lors de l'élimination et de la restitution de faire un balayage d'élément non-nul en élément non-nul, ainsi que de placer dans ce chaînage les éléments créés au cours des calculs.

V. DEFINITION DES PROBLEMES

On a choisi un type de problèmes pour lequel les performances des méthodes pouvaient être considérées comme importantes pour les applications pratiques de la théorie des graphes.

Les 2 types de réseaux que nous avons étudiés sont schématisés par les figures 1 et 2.

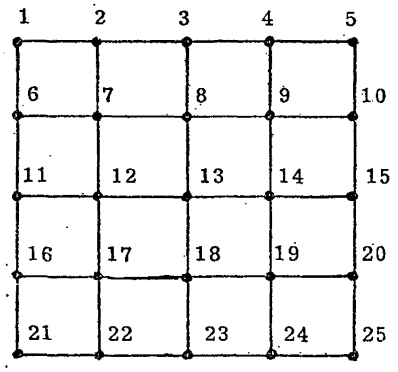

Figure 1

graphe symétrique
réseau 5 × 5 de 25 nœuds

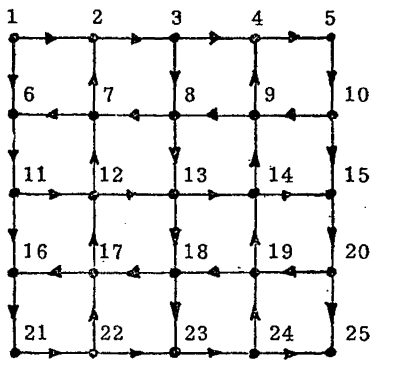

Figure 2

graphe antisymétrique
réseau 5 × 5 de 25 nœuds

Nous avons choisi comme longueur des arcs une valeur aléatoire uniformément distribuée dans l'intervalle [0, 10]. Pour tous les algorithmes, nous recherchons les plus courts chemins d'un certain nombre d'origines (ou de destinations) à tous les autres nœuds du graphe (ou à partir de tous les autres nœuds).

Ces origines ou ces destinations sont réparties à l'intérieur du réseau de manière à ne pas favoriser tel ou tel algorithme.

VI. RESULTATS

TABLEAU 1. *Réseau à double sens* (cf. fig. 1)

ALGORITHME \ RÉSEAU		8 × 8 64 nœuds	16 × 8 128 nœuds	32 × 16 512 nœuds
Djikstra	durée en s	0,019 × n	0,037 × n	0,45 × n
	mémoire en K octet	3	6	26
Yen		0,013 × n	0,030 × n	0,18 × n
		3	6,5	26
Carré.....		0,07 + 0,01 × n	0,16 + 0,018 × n	1,53 + 0,10 × n
		22	51	400

TABLEAU 2. *Réseau à sens unique* (cf. fig. 2)

ALGORITHME \ RÉSEAU	8 × 8 64 nœuds	16 × 8 128 nœuds	32 × 8 512 nœuds
Djikstra	0,015 × n	0,030 × n	0,42 × n
	2	4	16,5
Yen	0,011 × n	0,019 × n	0,095 × n
	2	4,5	17
Carré.....	0,04 + 0,0075 × n	0,07 + 0,01 × n	0,35 + 0,08 × n
	13	29	208

Pour avoir des résultats explicites, pour un réseau de taille donnée, nous indiquerons le temps moyen de calcul d'une arborescence des plus courts chemins. Pour ceci nous ne faisons qu'un seul tirage des longueurs aléatoires, mais nous calculerons la moyenne sur une dizaine de constructions d'arborescences.

Dans le cas de l'algorithme de Carré, nous avons un temps fixe, correspondant à la phase de réduction de la matrice du graphe, puis un temps proportionnel au nombre de destinations. Dans le même tableau, nous indiquons la place mémoire occupée par les variables nécessaires aux calculs.

REMARQUE : Pour avoir le temps moyen pour n arborescences, on multipliera ces résultats par n comme cela est indiqué dans les 2 tableaux.

VII. CONCLUSION

Des deux tableaux du paragraphe VI, nous pouvons tirer les remarques suivantes :

* *Temps de calcul :*

— Pour des nombres peu élevés de destinations, la méthode de Yen est la meilleure.

— Lorsque ce nombre augmente, la méthode de Carré devient rapidement plus avantageuse.

— Dijkstra se détériore lorsque la taille du graphe devient importante.

* *Place mémoire :*

L'algorithme de Carré nécessite en général une place mémoire beaucoup plus importante que les autres algorithmes. On doit noter que cette place dépend de la structure du réseau.

Dans le cas d'utilisations répétées, après modification de la longueur des arcs, le temps calcul de l'algorithme de Carré est diminué. Cette diminution est de l'ordre de 30 % de la durée de la phase d'élimination. Ceci est dû au fait que le « chaînage » (cf. paragraphe IV) est réutilisable pour les calculs successifs.

Pour un utilisateur de tels algorithmes, l'algorithme « optimal » sera donc un compromis entre les résultats que nous avons donnés plus haut, ses besoins spécifiques, ses possibilités de calcul.

REMERCIEMENTS

L'auteur tient à remercier très vivement M. le Professeur B. A. Carré de l'Université de Southampton pour les très précieux conseils qu'il lui a prodigués, et pour certains résultats relatifs à ses travaux qu'il lui a fournis.

BIBLIOGRAPHIE

[1] B. ROY, *Algèbre moderne et théorie des graphes*, Tome II, Dunod, 1970, pp. 127-161.

[2] S. E. DREYFUS, *An Appraisal of some shortest path Algorithm*, Operational Research, 1969, 17, n° 3.

[3] G. B. DANTZIG, *Linear Programming and extensions*, Princeton university Press, pp. 361-366, 1963.

[4] W. L. PRICE, *Graphs an Network An Introduction*, Operational Research Series Butterworks, Chapitre III, pp. 46-59.

[5] L. R. FORD Jr et D. R. FULKERSON, *Flows in Network*, Princeton University Press, pp. 130-134, 1962.

[6] J. Y. YEN, *An algorithm for finding shortest Routes from all source nodes to a given destination in General networks*, Quaterly of Appl. Maths, 1969, 27, pp. 526-530.

[7] B. A. CARRÉ, *An elimination method for Minimal cost network flow Problem*, Large sparse sets of linear equations, J. K. Reid, Academic Press, 1971, pp. 191-209.

[8] B. A. CARRÉ, *An algebra for Network routing problems*, J. Inst Maths Applics, 1971 ; 7, pp. 273-294.

[9] M. PETITFRÈRE, *Sur l'algorithme de Dijkstra pour l'obtention des plus courts chemins dans un graphe*, Cahier du Centre d'Etude de R. O., Vol. 13, n° 3, 1971, pp. 111-123.

[10] G. FONTAN, *Sous programmes de recherche de chemins minimaux dans un graphe*, Note Interne L.A.A.S., n° 73 I 14, mai 1973.

[11] KNUTH, *The art of computer programming*, vol. 1, Fundamental Algorithm, Addison Wesley, 1972, pp. 228-304.