RAIRO. RECHERCHE OPÉRATIONNELLE

IRÈNE PAUMELLE

Un indicateur informatique d'itinéraire pour les réseaux d'autobus

RAIRO. Recherche opérationnelle, tome 20, n° 3 (1986), p. 199-211

<http://www.numdam.org/item?id=RO_1986__20_3_199_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 20, n° 3, août 1986, p. 199 à 211)

UN INDICATEUR INFORMATIQUE D'ITINÉRAIRE POUR LES RÉSEAUX D'AUTOBUS (*)

par Irène PAUMELLE (1)

Résumé. — *Le problème posé est celui de la conception d'un indicateur informatique d'itinéraires de station à station, donnant le meilleur trajet — ayant au plus deux changements — dans un réseau d'autobus suivant un critère choisi par l'utilisateur parmi les trois suivants :*

mini-temps, mini-marche, mini-changements.

Nous utilisons le fait que les lignes du réseau forment une partition des arrêts en prenant comme unité de base non plus la station, comme cela est fait en général, mais la ligne.

La recherche d'un chemin se fait alors dans un graphe dont le nombre de sommets est nettement diminué puisque ce n'est plus le nombre de stations, mais le nombre de lignes.

De plus l'algorithme présenté ici a une complexité qui ne dépend plus du nombre de sommets ni du nombre d'arcs du graphe mais d'autres paramètres.

Mots clés : Chemin optimal; réseau de transport; indicateur d'itinéraire.

Abstract. — *We consider the design of an informatic itenerary indicator from station to station, giving the best path — of at most two correspondances — in a bus network, according to a criterion chosen by the user among three possibilities:*

shortest time, shortest walk, least number of correspondances.

We use the property that the lines of the network define a partition of the stations in not taking as base unity the station, as being done in general, but the line.

The search of a path is done on a graph whose number of vertices clearly decreases: it is no more the number of stations but the number of lines.

Furthermore the algorithm we present has a complexity which does not depend on the number of vertices neither on the number of edges but on other parameters.

Keywords : Optimal path; transportation network; itinerary indicator.

(*) Reçu en septembre 1985.
(1) Institut de Programmation, Université Pierre et Marie Curie, 4, place Jussieu, 75005 Paris.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/86/03 199 14 $ 3.40,

0. INTRODUCTION

Il s'agit de concevoir un indicateur conversationnel d'itinéraires dans un réseau d'autobus, orienté vers la prise en compte d'éventuelles modifications du réseau, ce qui exclut donc un calcul des itinéraires une fois pour toutes.

L'indicateur doit donner le meilleur chemin d'au plus deux changements d'une station à une autre suivant un critère choisi par l'utilisateur parmi les trois suivants :

mini-temps, mini-marche, mini-changements.

Ce problème nous a été plus particulièrement posé pour le réseau d'autobus de Paris intra muros.

1. LE PROBLÈME

1.1. Réseau d'autobus

— La notion de correspondance de type métro n'y existe pas : s'ils ne portent pas le même nom, on ne sait pas à priori si deux arrêts sont en correspondance. Le seul critère est celui de la distance acceptable à parcourir à pied entre ces deux arrêts. De plus on ne peut regrouper les arrêts en correspondance, la relation « être proche de » n'étant pas transitive.

— Du fait des exigences de la circulation, le parcours aller d'une ligne est rarement le même qu'au retour : pour Paris, sur 55 lignes seules 3 ont un parcours aller et retour identique. C'est pourquoi chaque ligne sera décomposée en deux « parcours » distincts.

1.2. Méthodes de résolution

En général les études de cheminement dans un réseau de transport s'appuient sur la théorie des graphes et ses algorithmes de plus court chemin. Le réseau y est représenté par un graphe dont les sommets sont les arrêts. Les arcs sont de deux types :

— Les « arcs de parcours » qui relient deux stations consécutives d'une même ligne.

— Les « arcs de passage » qui relient deux arrêts en correspondance.

En utilisant par exemple l'algorithme de Moore-Dijkstra on obtient pour le 1er critère une complexité en $O(m \log n)$, où m est le nombre d'arcs et n le nombre de sommets. Pour les deux autres critères, on ne peut pas utiliser cet algorithme sans modifications.

Nous présentons ici une méthode différente : le réseau est représenté par un graphe dont les sommets sont les *parcours*, (un « parcours » est une ligne associée à un sens de parcours de cette ligne), deux sommets étant reliés par un arc si une station de l'un est en correspondance avec une station de l'autre. Le problème revient à chercher les chemins de longueur au plus deux dans le graphe ainsi défini.

On trouve alors une complexité qui ne dépend plus du nombre de sommets ni du nombre d'arcs du nouveau graphe.

De plus on ne génère par cette méthode dans le pire des cas que des chemins allant effectivement de la source à l'arrivée et non tous les plus courts chemins allant de la source aux autres sommets.

Nous présentons ensuite une méthode pour éviter de générer des chemins aberrants.

2. DÉFINITIONS. LE GRAPHE G

2.1. Définitions préliminaires

1. Un *parcours* P est une ligne associée à un sens de parcours; c'est une suite d'arrêts :

$$P = (a_1, \ldots, a_p).$$

2. Une *ligne* L est la paire constituée par un parcours aller P et son parcours retour $\overline{P}$:

$$L = \{P, \overline{P}\}.$$

3. Le *rang* d'un arrêt a, noté $rg_p(a)$ est le rang qu'occupe a sur le parcours P.

4. Un *chemin du réseau* R de l'arrêt a_1 à l'arrêt b_q est une suite de la forme :

$$a_1 P_1 b_1 \quad a_2 P_2 b_2 \ \ldots \ a_q P_q b_q, \quad q \geqslant 1$$

où pour tout i, a_i et b_i sont des arrêts, P_i est un parcours.

Le chemin « s'énonce » :

Aller à l'arrêt a_1; prendre le parcours P_1; descendre à b_1
⋮
aller à a_q; prendre le parcours P_q; descendre à b_q.

5. *Évaluation d'un chemin C du réseau R*

Définissons trois fonctions d'évaluation d'un chemin C suivant les 3 critères :

Critère 1 : mini-temps

Fonction durée : f_d

$f_d(C)$ = durée globale du chemin C
(durée de marche, d'attente, et de transport).

Critère 2 : mini-marche

Fonction marche : f_m

$f_m(C)$ = durée de marche à pied de C.

Critère 3 : mini-changements

Fonction changements : f_c

$f_c(C)$: nombre de changements de C.

Dans le cas général ces fonctions sont égales à :

$$f_d(C) = \sum_{i=1}^{q} m(b_{i-1}, a_i) + ta(P_i) + t_{P_i}(a_i, b_i) + m(Ad_2, b_q)$$

$$f_m(C) = \sum_{i=1}^{q} m(b_{i-1}, a_i) + m(Ad_2, b_q)$$

$$f_c(C) = q$$

en posant : $b_0 = Ad_1$: adresse de départ; Ad_2 : adresse d'arrivée

$m(b, a)$: temps de marche de b à a

$t_p(a, b)$: temps de parcours en autobus pour aller de l'arrêt a à l'arrêt b.

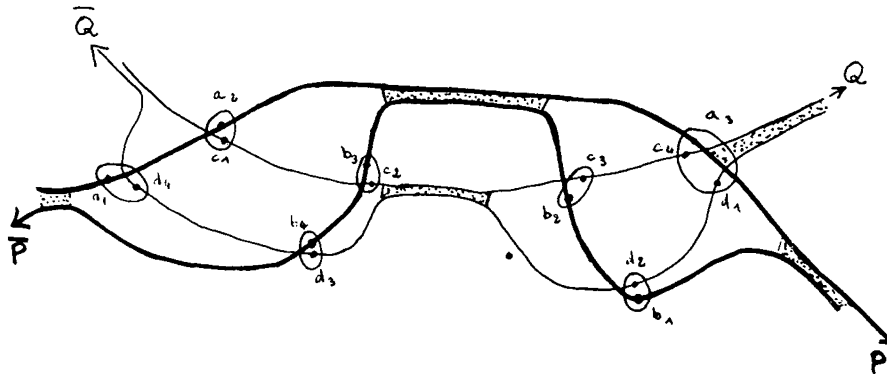
$ta(P_i)$: temps d'attente sur le parcours P_i.

2.2. Représentation du réseau par le graphe G

— Les sommets de G sont constitués par les parcours du réseau.

— Il existe un arc de P à Q si et seulement si P et Q se coupent, autrement dit s'il existe un arrêt c de P et un arrêt d de Q qui sont en correspondance. Deux arrêts sont en correspondance si leur distance est inférieure à une distance d_0 fixée qui dépend de l'endroit considéré : pour le centre de Paris, d_0 sera faible, pour l'est de Paris, mal desservi, d_0 sera plus grande. L'intersection $I(P, Q)$ de deux parcours P et Q est l'ensemble des couples d'arrêts de P et Q en correspondance.

Exemple :



On a sur cet exemple :

$$I(P, Q) = \{(a_2, c_1), (a_3, c_4)\}$$
$$I(P, \overline{Q}) = \{(a_1, d_4), (a_3, d_1)\}$$
$$I(\overline{P}, Q) = \{(b_2, c_3), (b_3, c_2)\}$$
$$I(\overline{P}, \overline{Q}) = \{(b_1, d_2), (b_4, d_3)\}.$$

Une intersection multiple correspond par exemple au cas où deux parcours suivent le même itinéraire sur plusieurs stations.

Pour Paris le cardinal d'une intersection peut être important; cependant on peut se ramener à des intersections de cardinal maximum 4; en effet si deux parcours se suivent sur plus de deux stations consécutives, seules les stations extrêmes de la portion commune présentent un intérêt pour la correspondance.

2.3. Relations entre chemins de R et chemins de G

Un chemin de G est appelé *métachemin.*

Un chemin de R à i changements est appelé un *i-chemin.*

Comment rechercher les chemins entre deux adresses Ad_1 et Ad_2 à partir du graphe G?

Notons $p(Ad)$ l'ensemble des parcours passant à proximité de Ad.

Soient $P \in p(Ad_1)$, $Q \in p(Ad_2)$.

La condition « il existe un métachemin de P à Q » n'est pas suffisante pour affirmer qu'il existe un chemin de Ad_1 à Ad_2.

Il faut ajouter des contraintes sur les rangs :

Soit $P_1, P_2, \ldots, Q$ un métachemin de P à Q.

Pour qu'il existe des chemins de Ad_1 à Ad_2 il faut que $\forall i$ il existe un point de correspondance (c, d) entre P_i et P_{i+1} tel que c soit après l'arrêt de montée sur P_i et que d soit avant l'arrêt de descente sur P_{i+1}.

Nous allons voir de quelle manière utiliser ces résultats pour calculer les chemins.

3. RECHERCHE DES CHEMINS ENTRE DEUX ADRESSES. COMPLEXITÉ

3.1. L'algorithme

Le problème revient donc à trouver des métachemins de longueur maximum 2 entre deux ensembles de sommets, respectivement les parcours proches de Ad_1 et ceux proches de Ad_2, puis en déduire les chemins associés du réseau.

La recherche se fait dans l'ordre croissant des nombres de changements.

Nous donnons ici l'algorithme de recherche des 2-chemins, celui des 1-chemins et des 0-chemins s'en déduisant de façon naturelle.

```
Pour tout P∈p(Ad1) faire
    Pour tout Q∈Γ(P) faire
        Pour tout (c, d)∈I(P, Q) faire
            si (c, d) satisfait les contraintes de rangs alors
                Pour tout R∈Γ(Q) faire
                    Si R∈p(Ad2) alors
                        Pour tout (e, f)∈I(Q, R) faire
                            Si (e, f) satisfait les contraintes sur
                                    les rangs alors
                                Générer les chemins de R correspondant.
                            finsi
                        fin faire
                    finsi
                fin faire
            finsi
        fin faire
    fin faire
fin faire.
```

Bien qu'à première vue cet algorithme présente de nombreuses boucles imbriquées, il faut remarquer que ces boucles portent sur des ensembles de cardinal faible.

3.2. Complexité de l'algorithme

Considérons la liste suivante des constantes d'un réseau suivies de leur valeur entre parenthèses pour Paris intra muros.

c : nombre maximum de points de correspondance entre deux parcours (4)

γ : degré maximum d'un sommet de G (70)

p : nombre maximum de parcours passant près d'une adresse (15).

La complexité $c(i)$ de la recherche dépend de ces constantes et du nombre de changements i ($i = 0, 1, 2$) des chemins cherchés.

Supposons les données représentées de la façon suivante :

Les ensembles $p(Ad_1)$, $I(P, Q)$ sont représentés par des files.

L'ensemble $p(Ad_2)$ est représenté par sa fonction caractéristique.

Le graphe G est représenté par sa file de successeurs.

REMARQUE : en général P et $\overline{P}$ passent tous deux près d'une adresse Ad. Les successeurs actifs de P à partir de Ad ne le seront donc pas sur $\overline{P}$ et réciproquement. P et $\overline{P}$ ont donc à eux deux au plus γ successeurs actifs.

On a alors :

$$i = 0 : \quad c(0) = O(pc) \qquad (60)$$

$$i = 1 : \quad c(1) = O\left(p \frac{\gamma}{2} c\right) \qquad (2\,100)$$

$$i = 2 : \quad c(2) = O\left(pc^2 \frac{\gamma^2}{4}\right) \qquad (294\,000)$$

Ces complexités ne dépendent donc pas directement du nombre total d'arrêts ni du nombre d'arcs.

3.3. Comparaison avec une méthode classique de recherche de plus court chemin

Cette comparaison ne peut s'effectuer que pour le premier critère, les deux autres ne posant pas sans modification le problème du plus court chemin.

Dans une méthode classique, par exemple l'algorithme de Dijkstra, la recherche se ferait avec les deux contraintes suivantes :

— pas deux arcs de passage de suite

— pas plus de deux arcs de passage au total (sans compter les arcs de rabattement).

Soient $a_1, a_2, \ldots, a_k$ les arrêts proches de Ad_1 et $b_1, b_2, \ldots, b_q$ les arrêts proches de Ad_2.

On ajoute au départ tous les arcs — de rabattement — (Ad_1, a_i) pour $i = 1, \ldots, k$ et tous les arcs (b_j, Ad_2) pour $j = 1, \ldots, q$.

Puis on génère tous les plus courts chemins de Ad_1 aux autres sommets jusqu'à ce qu'on atteigne Ad_2.

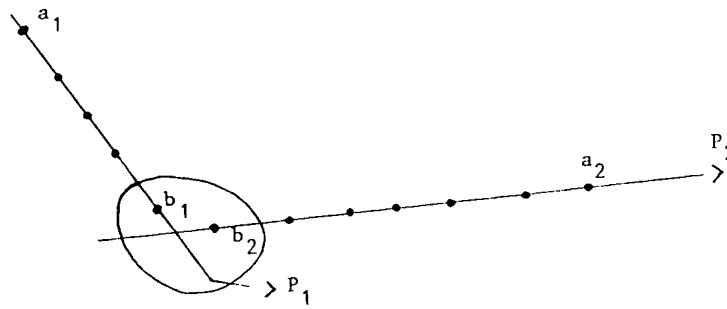
On ne génère donc qu'un seul chemin effectif de Ad_1 à Ad_2, le plus court. Dans le pire des cas on génère les plus courts chemins de Ad_1 à tous les autres sommets.

Dans la méthode des parcours, les chemins ne sont pas calculés par longueur croissante mais par nombre de changements croissants, dans un ordre quelconque pour un nombre de changements fixé.

On ne calcule que des chemins allant effectivement de Ad_1 à Ad_2, et dans le pire des cas on les calcule tous.

Le nombre d'étapes pour déterminer un chemin est fonction uniquement du nombre de changements de ce chemin et non de sa longueur. En effet on ne décrit pas un parcours arrêt par arrêt mais directement d'un arrêt de montée à un arrêt de descente :

EXEMPLE : pour déterminer un chemin de a_1 sur P_1 à a_2 sur P_2, sachant que P_1 et P_2 se coupent en (b_1, b_2), on a immédiatement le chemin $a_1 P_1 b_1\, b_2 P_2 a_2$ sans passer par les arrêts intermédiaires, en travaillant avec les distances cumulées.



L'intérêt principal de la méthode des parcours et qu'elle permet d'obtenir très rapidement les chemins directs et à un changement. En ce qui concerne les 2-chemins, si on compare la complexité d'un calcul de plus court chemin par Dijkstra utilisant un minimier, qui est en $O(m \log n)$, on constate que pour une ville comme Paris les résultats sont à peu près équivalents :

En supposant qu'en moyenne un arrêt sur trois est en correspondance avec en moyenne cinq arrêts on obtient 8 000 arcs (3 000 arcs de parcours et 5 000 arcs de passage) et 3 000 arrêts, cela donne :

$$m \log n \approx 96\,000.$$

4. OPTIMISATION. RECHERCHE DU MEILLEUR CHEMIN

Mais on ne cherche qu'un chemin, le meilleur, et la plupart des chemins générés par l'algorithme précédent sont mauvais; nous allons voir maintenant comment éviter de les générer tous. Pour cela nous utiliserons deux méthodes :

– La notion de chemin « *acceptable* » :

Quand on aura trouvé un chemin acceptable on ne continuera pas la recherche.

– Éviter de générer certaines classes de 2-chemins qui manifestement sont mauvais.

Tout d'abord fixons-nous les constantes théoriques suivantes :

Distance inter-arrêt théorique : $\hat{d}$

vitesse théorique active maximale : $\hat{v}$

temps d'attente théorique actif minimal sur un parcours : $\hat{ta}$.

4.1. Méthode 1

Nous avons choisi de générer les chemins par nombre de changements croissant. En effet, il y a des chances pour qu'un trajet à i changements soit meilleur qu'un trajet à $(i + 1)$ changements, sauf peut-être suivant le critère 2.

On ne calculera donc les $(i + 1)$-chemins que s'il n'y a pas de solution acceptable à i changements.

Pour un j-chemin, soit m_j la durée de marche à pied acceptable et α_j le coefficient d'acceptabilité : α_j exprime de combien on accepte de s'écarter de la ligne droite pour rester dans la catégorie des j-chemins acceptables.

On définit alors la durée T_j d'un j-chemin acceptable :

$$T_j = \hat{m}_j + (j + 1)\hat{ta} + \alpha_j \hat{d}\hat{v}$$

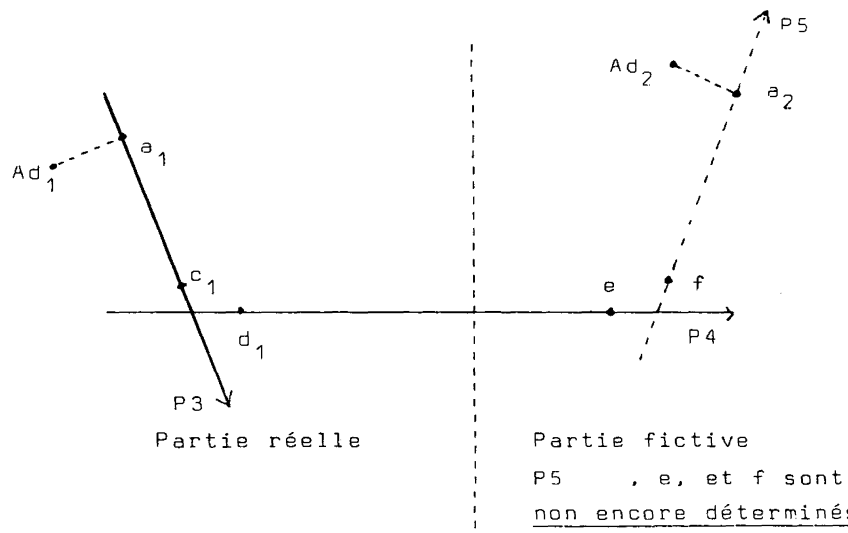
correspondances	attente	temps en autobus

Si le meilleur j-chemin a une durée inférieure à T_j on n'entreprend pas le calcul des $(j + 1)$-chemins.

4.2. Méthode 2

Critère 1. (Mini-temps)

Supposons qu'on ait déterminé les deux premiers parcours P_3 et P_4 d'un métachemin et un point d'intersection (c_1, d_1) :



On connaît la partie du chemin allant de a_1 à d_2 et prenant un temps T_1. La deuxième partie, non encore fixée, sera évaluée au plus court suivant le critère et prendra un temps T_2.

Soit n_1 le nombre d'arrêts théorique de d_1 à Ad_2, calculé en fonction de la distance à vol d'oiseau entre d_1 et Ad_2.

Posons alors :

$$T_2 = n_1 \times \hat{d} \times \hat{v} + t\hat{a} + m(e, f) + m(Ad_2, a_2)$$

où les deux dernières quantités sont choisies minimales.

T_2 est choisi tel qu'un 2-chemin réel dont le début serait $a_1 P_3 c_1 d_1 P_4$ dure certainement plus longtemps que $T_1 + T_2$.

Donc si f_0, meilleure valeur obtenue jusque là est telle que $f_0 < T$, on ne continue pas le calcul de la classe de chemins commençant par P_3 et P_4.

L'estimation de T_2 prend 8 opérations, ce qui est rentable par rapport au calcul de tous les chemins commençant par P_3 et P_4.

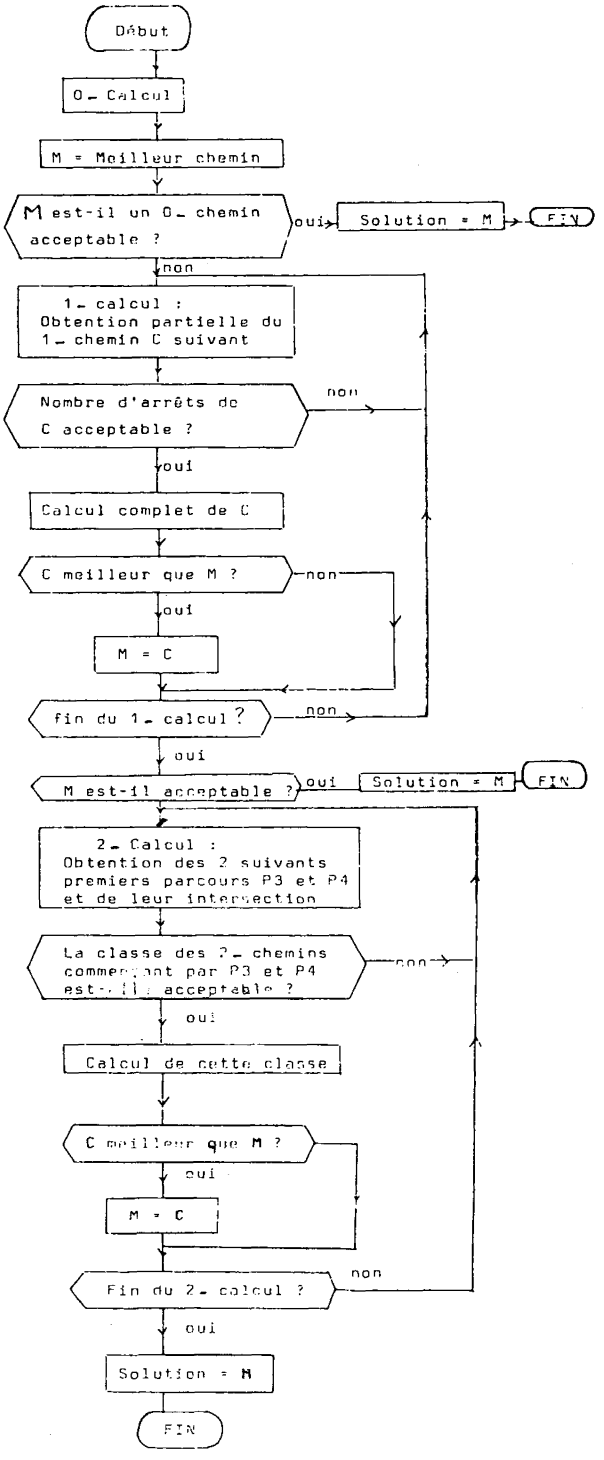
Critère 2. (Mini-marche)

On fait une évaluation analogue à celle du critère 1, tenant compte surtout des durées de marches à pied.

Critère 3. (Mini-changements)

Comme on obtient les chemins dans l'ordre de leur nombre de changements, on arrête tout naturellement la recherche lorsqu'on a trouvé un i (i est donc minimum) tel qu'il existe des (i-chemins). Parmi ces i-chemins on choisira par exemple le plus court.

PROGRAMME MEILLEURCHEMIN



Les considérations précédentes mènent donc à l'organigramme suivant, qui reprend l'algorithme précédent, en l'accélérant sensiblement.

5. APPLICATION

L'algorithme du § 3 a été testé sur un réseau s'inspirant du réseau parisien, comportant 28 parcours et 395 arrêts. Voici un exemple d'exécution du programme donnant les chemins d'une station à une autre, ainsi que leur durée et leur temps de marche à pied.

```
DONNEZ DEUX STATIONS ?          15              71
HEURE DE POINTE (HP) OU HEURE CREUSE (HC)?  HP
PAS DE CHEMIN DIRECT
AL31 LIGNE 6 CHANGER A BF10 LIGNE 5 BG09
 TEMPS TOTAL : 116        TEMPS DE MARCHE : 12
AL31 LIGNE 6 CHANGER A BL10 LIGNE 10 HG09
 TEMPS TOTAL : 122        TEMPS DE MARCHE : 12      } 1-chemins
AL31 LIGNE 6 CHANGER A AN28 LIGNE 14 BG09
 TEMPS TOTAL : 135        TEMPS DE MARCHE : 12

    3 CHEMINS A 1 CHANGEMENT
AL31 LIGNE 6 CHANGER A BK11 LIGNE 7 CHANGER A BG15 LIGNE 5 BG09
TEMPS TOTAL : 538        TEMPS DE MARCHE : 24
AL31 LIGNE 6 CHANGER A AT21 LIGNE 9 CHANGER A AN22 LIGNE 10 BGO9
TEMPS TOTAL : 397        TEMPS DE MARCHE : 24
AL31 LIGNE 6 CHANGER A AT21 LIGNE 9 CHANGER A AK22 LIGNE 14 BG09
TEMPS TOTAL : 430        TEMPS DE MARCHE : 24
AL31 LIGNE 6 CHANGER A AO27 LIGNE 11 CHANGER A AM26 LIGNE 14 BG09
TEMPS TOTAL : 276        TEMPS DE MARCHE : 24
AL31 LIGNE 6 CHANGER A AN28 LIGNE 25 CHANGER A BJ02 LIGNE 5 BG09
TEMPS TOTAL : 215       TEMPS DE MARCHE : 24
AL31 LIGNE 6 CHANGER A AN28 LIGNE 25 CHANGER A AN28 LIGNE 14 BG09
TEMPS TOTAL : 233        TEMPS DE MARCHE : 24
6 CHEMINS A 2 CHANGEMENTS
```

6. CONCLUSION

Nous avons présenté dans cet article une méthode de recherche originale d'un chemin optimal suivant différents critères dans un réseau d'autobus. Cette méthode utilise un graphe dont la combinatoire est fortement diminuée par

rapport à la représentation classique des réseaux de transport. L'algorithme qui en découle est très rapide pour les chemins directs et à 1 changement. Pour les chemins à 2 changements, nous exposons une méthode d'accélération des calculs basée sur l'évaluation à priori de chemins non encore complètement calculés.

BIBLIOGRAPHIE

RATP, *Calcul de chemins de valeurs minimales appliqué aux études de réseaux ferroviaires urbains.* Oct. 1970, vol. 1 et 2.

RATP, *Critère d'accessibilité urbaine.* Application aux transports en commun dans Paris intra-muros, mai 1974.

B. Roy et D. Galland, *Énumération des chemins ε-admissibles entre deux points, RAIRO verte* septembre 73, 1972.

D. R. Shier, *On algorithms for finding the k shortest paths in a network,* Networks, vol. 9, p. 195-214, 1979.

Gondran et Minoux, *Graphes et algorithmes, Paris Eyrolles,* 1979.

E. W. Dijkstra, *A note on two problems in connexion with graphs,* Numerische Mathematik 1, p. 269-271, 1959.

B. Meyer et C. Baudouin, *Méthodes de Programmation,* Eyrolles, 1978.

A. V. Aho, J. E. Hopcroft, J. D. Ullman. *Data structures and algorithms,* Addison-Wesley, 1983.

I. Paumelle, *Thèse de 3e cycle,* Université Paris-Dauphine. Un outil informatique d'aide aux usagers des transports en commun. Application au réseau d'autobus parisien, juin 1984.