REVUE FRANÇAISE D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. SÉRIE VERTE

E. VALENSI
N. DE BUNGE

Calcul de meilleurs itinéraires de distribution et de collecte

Revue française d'informatique et de recherche opérationnelle. Série verte, tome 4, n° V1 (1970), p. 7-17

<http://www.numdam.org/item?id=RO_1970__4_1_7_0>

R.I.R.O.
(4e année, V-1, 1970, p. 7-18)

CALCUL DE MEILLEURS ITINÉRAIRES DE DISTRIBUTION ET DE COLLECTE

E. Valensi et N. de Bunge (1)

Résumé. — *Des marchandises devant être prises en différents points et transportées par camions vers un point central, le programme de distribution et de collecte DIC construit les itinéraires des véhicules nécessaires au transport sous des contraintes de capacité et d'autonomie. L'algorithme a permis l'obtention de 12 itinéraires desservant 100 villes en 10 secondes sur UNIVAC 1108 .*

INTRODUCTION

Calcul de meilleurs itinéraires de distribution et de collecte

Le programme de distribution et de collecte, DIC, est un ensemble de sous-programmes rédigés en FORTRAN, permettant de résoudre une famille de problèmes de transport liés à la recherche de plus courts circuits sous des contraintes d'autonomie et de capacité. Il généralise, à des problèmes variés, une procédure utilisée pour la résolution du problème du voyageur de commerce.

Après avoir décrit la méthode d'affectation au centre de l'application, nous montrons les larges possibilités d'adaptation offertes par la structure DIC. Le déroulement des calculs est le suivant :

On cherche des sous-ensembles maximaux de points « voisins » à desservir, satisfaisants aux contraintes d'autonomie et de capacité. Sur les sous-ensembles ainsi déterminés, on minime le coût de transport en appliquant une procédure de recherche de plus court-circuit hamiltonien, puis en cherchant le meilleur sens de parcours du circuit obtenu. L'algorithme a permis de décomposer en 12 itinéraires la desserte de 100 villes et de trouver leur parcours en 10 secondes sur UNIVAC 1108.

(1) Centre Interarmées de Recherche Opérationnelle.

DIC est une partie d'un ensemble de programmes FORTRAN qui forment l'amorce d'un code de programmation de recherche opérationnelle. Il s'agit de sous-programmes modulaires que l'on assemble comme les pièces d'un meccano logique pour effectuer les opérations rencontrées dans les procédures de recherche d'optimum combinatoire les plus variées. DIC montre comment utiliser un tel code :

Les sous-programmes sont hiérarchisés. Ils comprennent des programmes de tri, de traitement des matrices (permutation de ligne), de gestion d'arborescence, etc., qui sont les fonctions de base. Ces fonctions sont préassemblées par des procédures standards qui effectuent des recherches d'optimum combinatoire de caractère théorique :

— Recherche d'un plus court cycle hamiltonien, etc...

DIC est une telle procédure standard. Enfin, ces procédures sont utilisées pour résoudre des problèmes réels comme sous-programmes d'algorithme à caractère itératif. Il faut insister sur ce dernier point, car il impose aux procédures d'être rapides ce qui entraîne l'utilisation de méthodes heuristiques, au niveau des procédures standards.

L'intérêt du travail présenté, est de montrer comment il est possible, à partir de ces procédures standards d'obtenir relativement facilement les solutions de problèmes complexes par l'utilisation de plusieurs procédures heuristiques simples mais complémentaires.

1. LE PROBLEME ET LA METHODE

1.1. Problème

Pour livrer ou prendre des marchandises, en un certain nombre de points ou zones géographiques, on dispose d'un ensemble de véhicules dont on doit déterminer les itinéraires qui miniment une fonction de coût qui sera précisée plus loin. Chaque problème est caractérisé par le graphe représentant les relations géographiques des villes à desservir, par les contraintes associées aux itinéraires et par la forme de la fonction économique.

1.1.1. Le graphe

Il est caractérisé par deux types de données :

— Les points à desservir sont assimilés aux sommets d'un graphe complet dont les arcs représentent la plus courte distance entre deux sommets (on pourrait supprimer *a priori* certaines liaisons considérées comme impossibles dans une solution optimale.)

— A chaque sommet est associé un nombre qui représente la quantité de marchandise à livrer ou à prendre en ce point ; cette quantité peut être nulle, le problème est alors celui du voyageur de commerce.

---

(1) Le listing complet du programme sera donné aux personnes qui en feront la demande.

1.1.2. Itinéraires

Les itinéraires partent et aboutissent en un point central, la base. Leur longueur peut être limitée par deux types de contraintes sur les véhicules :

— La contrainte d'autonomie qui empêche de parcourir plus d'une certaine distance.

— La contrainte de capacité qui limite la quantité de biens qui peut être chargée.

De plus, pour faciliter la résolution, on impose aux villes de n'être desservies que par un seul véhicule, excepté quand la quantité à livrer (ou à retirer) est supérieure à la capacité des véhicules. Dans ce cas, nous associerons, *a priori*, à chacune de ces villes un nombre de véhicules suffisant pour que la quantité de biens qui puisse être transportée soit inférieure à la plus grande capacité des véhicules restants. Dans le cours de l'algorithme on ne considérera que cette quantité restante.

1.1.3. Critère économique

Deux éléments interviennent dans le coût du transport :

— La distance totale à faire parcourir aux véhicules (on suppose que les coûts dus aux trajets varient linéairement avec la distance) ;

— Le nombre de véhicules nécessaires pour assurer le transport.

Si on se contente de chercher des itinéraires sans faire intervenir de contrainte de capacité, c'est le cas en particulier de la recherche de circuits sur des objectifs à reconnaître, minimer la distance à parcourir est équivalent à minimer le nombre de véhicules.

Si les contraintes de capacité interviennent, c'est le cas général, il n'est plus possible de minimer en même temps, le nombre de véhicules et la distance à parcourir. Car, si on désire remplir complètement un véhicule, les biens à prendre étant des quantités discrètes non fractionnables, il faut assembler des charges de façon que leur somme soit très voisine de la capacité maximale ; ces charges complémentaires n'ont aucune raison d'être sur un itinéraire optimal. Il faut donc utiliser une fonction économique qui prenne en compte, simultanément, les deux éléments du coût ; ceci sera fait de façon indirecte au moment de la formation des cycles.

1.2. Principe de l'algorithme. Les écartements

Dans cette méthode [1], on sait que l'on introduit un certain couplage entre les arcs en remplaçant la notion de distance, par celle d'écartement.

Faisons choix d'un centre, le sommet 1 par exemple ; on caractérisera l'arc (i, j) par son écartement.

$$e_{ij} = d_{ij} - d_{1j} - d_{ij}$$

(1) Cf B. Roy. — Algèbre moderne et théorie des graphes (DUNOD).

La longueur M d'un circuit passant par les points $i_1 = 1$, i_2, ... i_{n-1}, $i_n = 1$ est :

$$L = 2 \sum_{j=2,n-1} d_{1,i_j} - \sum_{j=2,\ldots n} e_{i_{j-1},\, i}$$

Minimer la longueur du circuit est équivalent à maximer la somme des écartements. Un ensemble de sous-programmes appelés par un programme « leader » est chargé de réaliser de façon heuristique cette fonction. Ces sous-programmes déterminent un circuit ; solution plus ou moins approchée du problème du voyageur de commerce.

1.3. Introduction des contraintes d'autonomie

La méthode de recherche du plus court-circuit hamiltonien consiste à chercher le premier circuit que l'on peut obtenir en sélectionnant les arcs par ordre d'écartements décroissants, avec la réserve qu'ils ne forment ni fourche, ni boucle avec les arcs précédemment retenus.

La recherche d'itinéraires est fondée sur un principe semblable, mais modifié de façon à faire apparaître la décomposition suivante :

— Choisir un arc de départ, gaine d'un circuit.

— Construire un circuit cohérent mais provisoire. Tâcher d'améliorer le circuit provisoire.

On part de l'arc (i, j) dont l'écartement est maximal, mais au lieu de sélectionner le second arc d'écartement maximal dans l'ensemble des arcs, on ne le choisit que dans l'ensemble des arcs ayant i ou j pour extrémité et on itère le procédé tant que la contrainte d'autonomie n'est pas violée. On obtient un chemin dont les extrémités ont tendance à se rapprocher de la base. Pour avoir l'itinéraire, il suffit de joindre la base aux deux extrémités ; on obtient ainsi un circuit. Si la contrainte d'autonomie est violée, on applique la procédure de recherche du plus court-circuit hamiltonien décrit en 1 à l'ensemble des sommets du circuit. Si la longueur du nouveau circuit obtenu est inférieure à la limite d'autonomie, le dernier sommet testé peut être conservé et l'algorithme poursuivi à partir des nouvelles extrémités trouvées. Sinon, l'algorithme sera poursuivi d'une façon différente suivant le nombre de circuits formés.

En effet il faut éviter de générer de multiples amorces de circuits. Pour cela on modifiera l'algorithme de choix si trop de circuits ont été commencés. Cette modification sera décidée en faisant référence à une borne inférieure du nombre d'itinéraires nécessaires pour résoudre le problème. Celle-ci est trouvée, en appliquant l'algorithme du voyageur de commerce à l'ensemble des sommets ce qui donne un itinéraire unique dont la longueur est une borne inférieure de la somme des longueurs des circuits de la solution optimale, et en cherchant par excès combien de fois cette longueur contient la limite d'autonomie.

Si le nombre d'itinéraires créés est inférieur à cette borne, on choisit l'arc libre de plus grand écartement pour former l'amorce d'un nouvel itinéraire. Les arcs suivants sont retenus comme précédemment dans

l'ensemble des arcs liés aux extrémités des itinéraires en cours de construction ; on choisit les arcs de plus grand écartement jusqu'à ce que les contraintes d'autonomie ne puissent plus être satisfaites.

Lorsque le nombre minimal de circuits est atteint, on rend plus difficile la création de nouveaux itinéraires en ne s'imposant plus, pour développer les tournées, de prendre comme arc candidat un arc de plus grand écartement mais, successivement, les arcs liés aux extrémités, rangés dans l'ordre d'écartements décroissants. On peut ainsi, soit épuiser l'ensemble des sommets, la solution est alors composée des itinéraires trouvés, soit être obligé de créer un nouvel itinéraire. Dans ce cas, la borne inférieure du nombre de tournées que l'on avait utilisée n'était pas exacte et l'on doit reprendre l'algorithme en augmentant sa valeur de un.

1.4. Introduction des contraintes de capacité

Si l'on souhaite minimer le nombre de véhicules en ne considérant que les contraintes de capacité, on peut se contenter de la procédure heuristique suivante :

On affecte les sommets pris dans l'ordre de charge décroissante aux véhicules classés dans l'ordre de tonnage décroissant. On remplit, autant qu'il est possible, le premier véhicule par les premières charges. Lorsqu'il n'est plus possible d'ajouter la charge la plus forte restante, on assigne cette charge au second véhicule. Dès lors, on essayera d'assigner les charges, d'abord aux véhicules dont la capacité non utilisée est la plus faible, aux autres si c'est impossible. Lorsqu'aucun des deux ne peut accepter une charge, elle est affectée à un troisième véhicule, et on poursuit ainsi l'algorithme jusqu'à ce que toutes les charges soient prises.

Cet algorithme ne tient pas compte des distances, pas plus que la méthode des écartements ne tenait compte des charges. Pour agréger ces points de vue on peut construire un algorithme mixte à partir de grandeurs composites

$$q(i, j) = \alpha(P(i) + P(j)) + \beta e(i, j)$$

$$r(i, j) = \alpha P(k) + \beta e(i, j)$$

où $P(i)$ est la charge généralisée du sommet i, $q(i, j)$ l'écartement généralisé de l'arc ij.

Une valeur de α/β étant donnée pour trouver les itinéraires, on transforme l'algorithme de recherche d'itinéraire sous contraintes d'autonomie seules en remplaçant les écartements par les écartements généralisés, et en prenant comme nouvelle borne du nombre de tournées, la plus grande des valeurs trouvées en considérant, indépendamment, les contraintes d'autonomie et de capacité.

En faisant varier α/β on obtient une famille de solutions ; on conservera celles minimant la fonction économique.

2. ORGANISATION DU PROGRAMME

Après avoir éliminé les points qu'il est impossible d'atteindre, parce qu'ils sont trop loin, et après avoir assigné des véhicules aux centres assez importants pour qu'isolément ils saturent un ou plusieurs éléments de transport, la recherche d'un ensemble de tournées revient à grouper les villes en itinéraires provisoires, construits à partir de règles des « proximité », satisfaisant les contraintes du problème puis à minimer une fonction économique sur chaque sous-ensemble de ville ainsi obtenus. Ces tâches sont réalisées par une famille de sous-programmes modulaires, assemblés à la demande, modifiables indépendamment les uns des autres. L'algorithme de recherche est une procédure heuristique, cette restriction est impérative pour résoudre des problèmes réels de grande taille puisqu'il faudrait, au stade actuel de nos connaissances, des mémoires et des temps de calculs astronomiques pour les résoudre de façon optimale.

Les sous-programmes composants l'algorithme de recherche de circuit de distribution et de collecte, DIC, se regroupent en trois classes :

— Les sous-programmes de servitude et de communication.

— Les sous-programmes de recherche des sous-ensembles maximaux.

— Les sous-programmes maximant la fonction économique.

Ces sous-programmes sont appelés par un programme gérant, PETALE.

Dans l'ensemble des sous-programmes de servitude, INITIA lit les données, réinitialise les variables de façon à permettre un appel répété de DIC.

SPECIA élimine les sommets que l'on ne peut atteindre parce qu'ils sont trop loin.

NOMIN détermine le nombre minimal de tournées à créer.

Pour générer des amorces de circuit, on appelle d'abord SOLIT qui traite, *a priori*, les sommets dont les capacités saturent un ou plusieurs véhicules. Ils forment des sous-ensembles dégénérés à 1 sommet. On appelle ensuite PREM qui cherche l'arc de plus grand écartement généralisé dont les extrémités n'ont pas encore été retenues. Si le circuit partant de la base utilisant l'arc candidat et revenant à l'origine est admissible, l'arc est retenu comme germe d'un sous-ensemble et on peut continuer l'algorithme. Si des contraintes sont violées, un seul sommet peut être conservé, PRESOM sélectionne le sommet maximant la quantité $P(i)\alpha + d(1, i)\beta$. Il forme un sous-ensemble dégénéré.

MEMOR garde en mémoire les villes affectées à des itinéraires.

Pour développer les familles de sous-ensembles générés par PREM on fait appel à un groupe de sous-programmes gérés par FORCIN. Chaque sous-ensemble contient deux sommets terminaux ; le sous-programme

OUKEJ, d'abord appelé, détermine pour chacun de ces sommets l'arc non encore pris de plus grand écartement généralisé. RANGE, classe ces arcs par ordre d'écartements généralisés décroissants. On sélectionne le premier arc de la liste et l'on vérifie que le sommet candidat peut être intégré dans la solution. Ce test se fait par deux sous-programmes :

CTRDEV, qui calcule les paramètres du nouveau sous-ensemble formé,

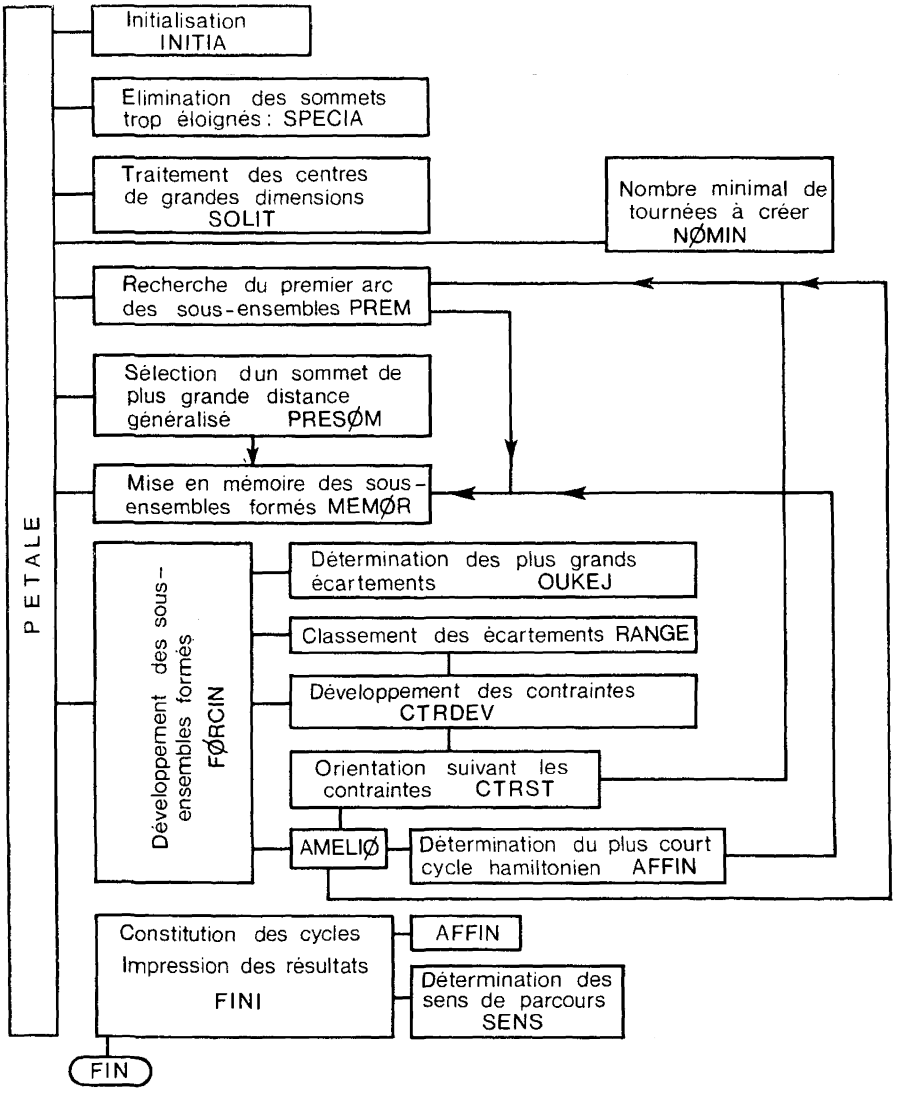


Figure A

Organigramme de Dic.

CTRST qui exploite les résultats de CTRDEV.

Si toutes les contraintes sont respectées, le sommet est affecté à la tournée. On met à jour la liste des écartements car d'autres arcs de la liste, que l'arc retenu, peuvent avoir comme extrémité, le dernier sommet sélectionné.

On itère l'algorithme sur la nouvelle liste obtenue.

Si la contrainte d'autonomie n'est plus respectée, on utilise le sous-programme de recherche du plus court circuit hamiltonien pour ordonner les sommets des sous-ensembles ; c'est le rôle d'AMELIO et d'AFFIN. Si le nouvel itinéraire respecte l'autonomie, le dernier sommet est inclus et les extrémités du chemin calculé deviennent celles du sous-ensemble. On met à jour la liste des écartements et on itère la procédure. Si AMELIO ne permet pas de réduire suffisamment la longueur du circuit, ou si la contrainte de capacité n'est plus satisfaite, on abandonne provisoirement le sommet.

Si le nombre minimal de tournées que l'on doit former n'est pas atteint, on commence un nouveau circuit en remontant à l'appel de PREM. Sinon, on décrit la liste des écartements pour voir s'il est possible d'affecter tous les sommets aux tournées déjà existantes. Si c'est impossible, on augmente de 1 le nombre de tournées minimum et on itère l'algorithme.

On appelle FINI lorsque tous les sommets sont affectés. Pour tous les sous-ensembles, FINI fait calculer par AFFIN le plus court circuit hamiltonien, puis appelle SENS qui détermine quel est le meilleur sens de parcours des circuits ; on peut prendre, par exemple, le sens minimant la quantité de tonne-kilomètre transportée.

3. UTILISATION DU PROGRAMME DIC

La troisième partie de cet exposé met en évidence la possibilité d'utiliser les sous-programmes de DIC comme les pièces d'un jeu d'assemblage permettant de construire facilement des programmes complexes ; nous donnerons deux exemples :

— L'emploi d'un parc de véhicules ;

— Le choix d'un site d'entrepôt.

Enfin, nous donnerons quelques détails sur les dimensions des programmes.

3.1. Emploi d'un parc de véhicules

Il ne s'agit plus de résoudre le problème théorique traité par DIC : trouver le nombre de véhicules nécessaires pour réaliser un ensemble de tournées, mais de tirer le meilleur parti d'une flotte de véhicules hétérogène et de capacité limitée. Ici, il n'est pas toujours possible de desservir l'ensemble des points, et les véhicules n'ont pas tous la même valeur éco-

nomique, si bien qu'il faut définir sur les points à desservir un ordre de priorité, et sur les véhicules un ordre d'utilisation. Il va de soi que l'on n'utilise un véhicule que si tous les véhicules plus rentables ont été employés et que l'on ne peut affecter un sommet que si tous les sommets classés avant lui ont été affectés.

Le nouvel algorithme d'affectation DISPAT est itératif, car DIC pour être efficace doit affecter directement le nombre exact de sommets au nombre exact de véhicules. Si, à une solution, on ajoute des véhicules et des sommets et que l'on forme de nouvelles tournées, l'ensemble obtenu n'est en général, pas bon. Notons que, s'il y a excès de véhicules on ne devra prendre que les meilleurs, et que s'ils sont en nombre insuffisant, il faudra sélectionner les points à desservir en priorité. Pour savoir quels véhicules utiliser, on appelle SELECT ; on emploie PRIOR pour calculer la priorité des sommets. Les sommets sont indicés dans cet ordre, et la matrice des distances transformée, les permutations de ligne et de colonne étant effectuées par TRANS.

L'organisation de DISPAT est la suivante :

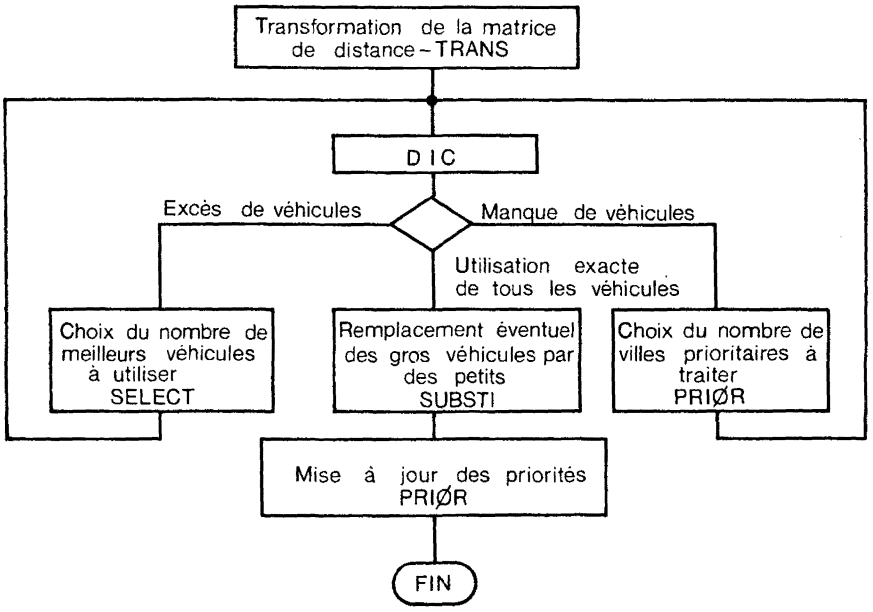


Figure B

Il n'est pas toujours possible d'utiliser les véhicules à plein ; en particulier pour la dernière tournée. Lorsque les affectations sont trouvées, SUBSTI améliore l'utilisation de la flotte en substituant, si c'est possible, des véhicules plus petits à des véhicules importants mal remplis.

3.2. Localisation d'un entrepôt de distribution

Un produit est fabriqué dans un point U. Il peut être transporté en vrac dans un endroit V où il sera conditionné et à partir duquel il sera distribué. Entre U et V, le coût de transport est faible et donné par un système de tarification fixe. A partir de U, le coût de distribution est élevé et complexe ; il est évalué par DIC.

Voici une recherche de meilleure localisation. On utilise DIC pour calculer le coût de distribution $D(V_i)$ et on lui ajoute le coût de transport $T(U)$; on retient les emplacements minimant la somme :

$$D(V) + T(U)$$

Pratiquement, il s'agit de calculer DIC pour des points centraux différents. Si on convient de choisir, pour point central, le point repéré par l'indice 1, donc représenté par la première ligne et par la première colonne de la matrice de distance, il suffit d'appeler TRANS pour changer de centre.

3.2.1. Organigramme pour la localisation d'un centre de distribution

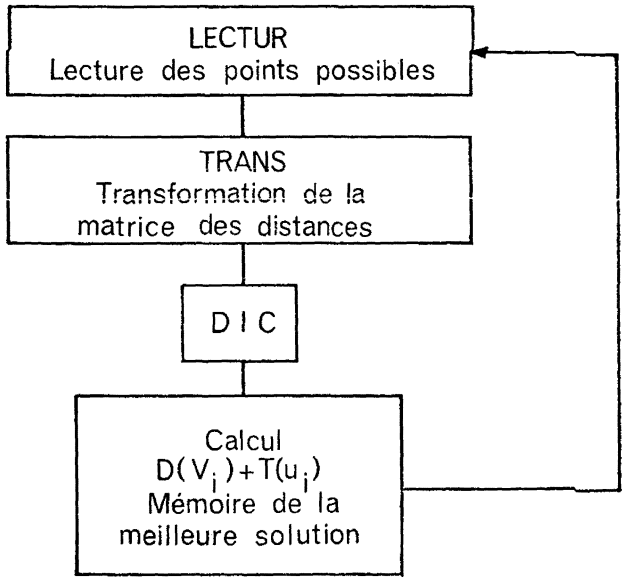
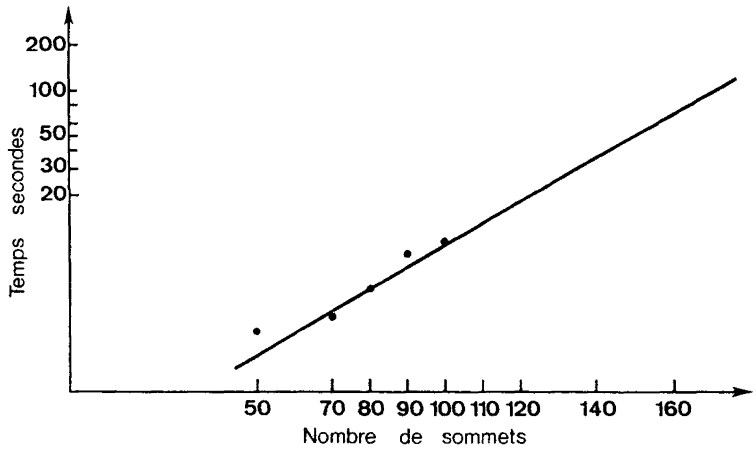


Figure C

4. TEMPS D'EXECUTION ET ENCOMBREMENT

Le programme DIC a été mis au point et testé sur UNIVAC 1108 avec des problèmes allant jusqu'à 100 sommets. Le temps d'exécution est donné par le diagramme en annexe, qui montre que, dans sa forme actuelle, la limite raisonnable de fonctionnement de l'algorithme doit se situer aux alentours de 160 villes. Pour cette dimension, on peut obtenir une solution en une minute environ. C'est à ce niveau que se situe la limite de la capacité de mémoire d'une machine de 64 K mots de 36 bits.



L'ensemble des ordres du programme demande 5 000 mémoires, et une majorante des dimensions des tableaux est donnée par :

$$\frac{3}{2} N^2 + 40\, N \text{ pour } N \text{ sommets}$$

Ainsi, un programme portant sur cent villes réclame 24 000 mémoires et peut être traité par une machine de 32 K mots de 24 ou 36 bits.