RAIRO. RECHERCHE OPÉRATIONNELLE

M. BEGHIN-PICAVET

P. HANSEN

Deux problèmes d'affectation non linéaires

RAIRO. Recherche opérationnelle, tome 16, n° 3 (1982), p. 263-276

<http://www.numdam.org/item?id=RO_1982__16_3_263_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 16, n° 3, août 1982, p. 263 à 276)

DEUX PROBLÈMES D'AFFECTATION NON LINÉAIRES (*)

par M. BEGHIN-PICAVET (1) et P. HANSEN (2)

Résumé : *Le problème de l'affectation de locaux à des services de façon à minimiser la somme des longueurs des trajets entre services des usagers se pose dans de nombreux organismes publics et privés, notamment les hôpitaux. Cet article est consacré à l'affectation optimale de locaux situés d'un côté ou des deux côtés d'un couloir. Ces problèmes sont exprimés sous la forme de problèmes d'affectation cubique et quadratique à structure particulière, et sont NP-complets. On propose des algorithmes de programmation dynamique pour les résoudre et on présente des résultats de calcul sur ordinateur.*

Mots clés : Algorithme; problèmes d'affectation non linéaire; programmation dynamique; solution optimale..

Abstract : *The problem of assigning rooms to departments in order to minimize the total distance between departments covered by the users arises in many contexts, notably hospitals. The optimal assignment of rooms on one or on both sides of a corridor is studied. This gives rise to a cubical and to a quadratic assignment problem, with special structure, respectively. These problems are shown to be NP-complete and are solved by dynamic programming. Computational experience is reported.*

Keywords: Algorithm; programming non linear; Programming assignment; programming dynamic; optimal solution.

1. INTRODUCTION

Le problème de l'affectation de locaux à des services se pose dans presque tous les organismes publics et privés. Un critère pour réaliser une telle affectation est la minimisation de la somme des longueurs des trajets entre services des usagers. Ce critère est adéquat dans les cas où les trajets entre services sont nombreux, par exemple dans les hôpitaux.

L'objet de cet article est l'étude de l'affectation à des services de locaux situés d'un côté ou des deux côtés d'un couloir. Tous les locaux seront supposés être de même taille, ce qui correspond au cas fréquent de construction modulaire; dans le premier cas, on supposera en outre que chaque service doit occuper un nombre donné de locaux contigus. Le problème est alors identique à un problème étudié

(*) Reçu Octobre 1980.
(1) I.U.T. Calais Dunkerque, Département Techniques de Commercialisation, Université Lille I, et U.E.R. I.E.E.A., Service Informatique, 59655 Villeneuve d'Ascq Cedex.
(2) Institut d'Économie Scientifique et de Gestion, Lille et Faculté Universitaire Catholique de Mons.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/1982/263/$ 5.00

par D. Simmons [8] : ranger des pièces de longueurs données le long d'un couloir de façon à minimiser la somme des déplacements entre pièces des gens qui s'y rendent.

Dans la section suivante, les deux problèmes cités sont formulés respectivement comme des problèmes d'affectation cubique et quadratique, à structure particulière. On montre dans la section 3 qu'ils sont *NP*-complets [5]. On présente à la section 4 des algorithmes de programmation dynamique permettant de les résoudre, illustrés d'exemples. La section 5 est consacrée à des résultats de calcul sur ordinateur, et de brèves conclusions complètent l'article. Les données et les solutions des problèmes de plus grande taille sont reproduites en annexe.

2. FORMULATION

Les hypothèses du problème 1 sont les suivantes :

H1. Un nombre donné n de services, indicés par i, sont à localiser le long d'un couloir bordé de locaux de taille identique, d'un seul côté. Le service i occupera un nombre donné d_i de locaux contigus.

H2. Le nombre de trajets entre les services i et j est donné par l'élément c_{ij} d'une matrice $C=(c_{ij})$ symétrique et de diagonale nulle. Ce nombre est indépendant des localisations de ces services.

H3. La longueur d'un trajet entre les services i et j est égale à $(d_i+d_j)/2$, plus la somme des longueurs d_k des services localisés entre i et j (on suppose donc que les portes sont situées au milieu des ensembles de locaux affectés à chaque service, et que les longueurs parcourues perpendiculairement au couloir sont constantes).

H4. L'objectif est la minimisation de la somme des longueurs des trajets entre services.

Le problème 1 consiste à déterminer une permutation σ rendant minimale :

$$\sum_i \sum_{j\in J_i} c_{ij}\left[\frac{d_i+d_j}{2}+\sum_{k\in K_{ij}} d_k\right]=\sum_i \sum_{j\in J_i} c_{ij}\left(\sum_{k\in K_{ij}} d_k\right)+\text{constante}, \quad (1)$$

où :

$$J_i=\{j/\sigma(j)>\sigma(i)\}$$

et :

$$K_{ij}=\{k/\sigma(j)>\sigma(k)>\sigma(i)\}\ (^3).$$

(3) Cette formulation concise est due au rapporteur, que nous remercions.

Il s'agit là d'un problème d'affectation cubique; lorsque $d_k = 1$ pour $k = 1, 2, \ldots, n$, (1) devient :

$$\sum_i \sum_{j \in J_i} c_{ij} | K_{ij} | + \text{constante} = \sum_i \sum_{j \in J_i} c_{ij} [\sigma(j) - \sigma(i) - 1] + \text{constante} \qquad (2)$$

et on obtient un problème d'affectation quadratique.

Dans le problème 2, les hypothèses H1 et H3 seront remplacées par les suivantes :

H1'. Un nombre pair donné n de services sont à localiser le long d'un couloir bordé de locaux de taille identique, des deux côtés. Chaque service occupera un local.

H3'. La longueur du trajet entre les services (ou locaux) i et j est égale à 0 si les services sont face à face, et sinon à 1 plus le nombre de locaux situés entre i et j d'un côté du couloir.

Le problème 2 s'écrit de la même manière que le problème 1, σ étant ici une fonction prenant deux fois chaque valeur entre 1 et $n/2$. Les d_k sont égaux à 1 pour $k = 1, 2, \ldots, n$. Il s'agit d'un problème d'affectation quadratique.

3. COMPLEXITÉ

Examinons maintenant la complexité des problèmes 1 et 2.

Nous allons montrer que les problèmes de décision associés (existe-t-il une solution de valeur inférieure à r' ?) aux problèmes 1 et 2 sont NP-complets.

THÉORÈME : *Les problèmes* 1 *et* 2 *sont* NP*-complets lorsque* $d_i = 1$ *pour* $i = 1, 2, \ldots, n$.

Démonstration : Considérons *le problème du rangement des sommets d'un graphe :* étant donné un graphe $G = (X, E)$ et un entier r, déterminer s'il existe une permutation $(t_1, t_2, \ldots, t_n)$ les indices des sommets $x_1, x_2, \ldots, x_n$ telle que :

$$\sum_{\{x_i, x_j\} \in E} | t_i - t_j | \leqq r. \qquad (3)$$

Ce problème est NP-complet [6].

En posant $d_i = 1$ pour $i = 1, 2, \ldots, n$,

$$c_{ij} = 1 \quad \text{si} \quad \{x_i, x_j\} \in E \quad \text{et} \quad c_{ij} = 0 \quad \text{sinon},$$

pour $i, j = 1, 2, \ldots, n$, on le transforme en un cas particulier du problème 1 (qui peut être exprimé sous la forme : existe-t-il une solution de valeur $\leqq r$?).

Considérons maintenant le problème 2, avec $2n$ locaux.

En posant :

$$c_{ij}=c_{n+i,\,n+j}=c_{i,\,n+j}=c_{n+i,\,i}=1 \qquad \text{si} \quad (x_i, x_j)\in E;$$

$$c_{i,\,n+i}=c_{n+i,\,i}=2n^3 \qquad \text{et} \qquad c_{ij}=0 \qquad \text{sinon,}$$

le problème du rangement est ramené à la recherche d'une solution du problème 2 de valeur inférieure à $4r$.

D'autre part, le fait qu'une solution donnée ait une valeur $\leqq 4r$ peut être vérifié en calculant cette valeur en $O(n^3)$ et en $O(n^2)$ opérations pour les problèmes 1 et 2 respectivement.

Les problèmes 1 et 2 sont NP-difficiles.

4. MÉTHODES DE RÉSOLUTION

4.1. Algorithme 1

D. Simmons [8] a proposé une procédure de séparation pour résoudre le problème 1. Dans une brève note [9] cet auteur signale aussi que le cas particulier où tous les $d_i=1$, $i=1, 2, \ldots, n$ peut être résolu par un algorithme de programmation dynamique de M. Held et R. Karp [7]. Ce même cas a été étudié par D. Adolphson et T. C. Hu [1] qui ont obtenu une borne inférieure sur la valeur de la fonction économique et un algorithme $o(n \log n)$ dans le pire des cas, en temps utilisable lorsque les $c_{ij}\neq 0$ forment une arborescence.

L'interprétation du problème de ces auteurs est en termes de cablage de circuits électroniques : on désire placer dans n emplacements situés le long d'une ligne, à égale distance, n circuits intégrés de façon à minimiser la somme des longueurs des connexions.

Montrons que la programmation dynamique permet de résoudre également le problème 1 dans le cas général.

Supposons que des services dont l'ensemble d'indices est S aient été affectés aux $|S|$ premières positions; les décisions restantes sont les choix des positions des services dont les indices appartiennent à $N\setminus S$ (où $N=\{1, 2, \ldots, n\}$) parmi les $|N\setminus S|$ dernières positions. Considérons la situation en un point situé à une distance égale à $\sum_{i\in S} d_i$ de l'extrémité du couloir. Les longueurs pondérées des trajets pour $i, j\in S$ sont connues ainsi que les longueurs pondérées des trajets pour $i\in S$ jusqu'en ce point.

La somme de ces longueurs pondérées ne dépend pas des décisions ultérieures. On peut donc appliquer le principe d'optimalité de Bellman [3]. Les équations de récurrence s'écrivent :

$$\left.\begin{aligned} f_S &= 0 \qquad \text{si} \quad |S| = 1, \\ f_S &= \operatorname*{Min}_{J \in S} \left(f_{S-\{J\}} + d_J \sum_{i \in S-\{J\}} \sum_{k \in N \setminus S} c_{ik} \right). \end{aligned}\right\} \tag{4}$$

Notons qu'il faudra ajouter à f_N la constante :

$$\sum_i \sum_{j>i} \left(\frac{d_i + d_j}{2} \right) c_{ij}.$$

Exemple 1 : Considérons un problème avec 4 services de longueurs $d_1 = 1$, $d_2 = 2$, $d_3 = 3$, $d_4 = 4$, et des c_{ij} donnés par la matrice suivante :

$$C = \begin{pmatrix} 0 & 3 & 4 & 6 \\ 3 & 0 & 2 & 2 \\ 4 & 2 & 0 & 5 \\ 6 & 2 & 5 & 0 \end{pmatrix}.$$

La résolution est présentée dans le tableau I ci-après. La seconde colonne caractérise S par un codage binaire.

TABLEAU I

J	Bin (J)	S	$\lvert S \rvert$	f_S	Ordre
1	0001	{1}	1	0	1...
2	0010	{2}	1	0	2...
3	0011	{1, 2}	2	4	2, 1...
4	0100	{3}	1	0	3...
5	0101	{1, 3}	2	7	3, 1...
6	0110	{2, 3}	2	15	2, 3...
7	0111	{1, 2, 3}	3	22	2, 3, 1...
8	1000	{4}	1	0	4...
9	1001	{1, 4}	2	7	4, 1...
10	1010	{2, 4}	2	20	2, 4...
11	1011	{1, 2, 4}	3	25	4, 1, 2...
12	1100	{3, 4}	2	24	3, 4...; ou 4, 3...
13	1101	{1, 3, 4}	3	22	4, 1, 3...
14	1110	{2, 3, 4}	3	43	2, 3, 4...
15	1111	{1, 2, 3, 4}	4	22	2, 3, 1, 4; ou 4, 1, 3, 2

La solution optimale est de ranger les pièces dans l'ordre 2, 3, 1, 4 (ou 4, 1, 3, 2).

$$\sum_{i=1}^{n} \sum_{j>i} c_{ij} \left(\frac{d_i + d_j}{2} \right) = 56 \qquad \text{donc} \quad z_1^* = 78.$$

4.2. Algorithme 2

Un raisonnement semblable à celui fait plus haut conduit aux équations de récurrence du problème 2 :

$$\left.\begin{aligned} f_{\{j,k\}} &= \sum_{l \in N-\{j,k\}} \frac{c_{jl}+c_{kl}}{2}, \\ f_S = \min_{j,k \in S}\Bigg(f_{S-\{j,k\}} &+ \sum_{i \in S-\{j,k\}} \frac{c_{ij}+c_{ik}}{2} \\ &+ \sum_{l \in N \setminus S} \frac{c_{jl}+c_{kl}}{2} + \sum_{i \in S-\{j,k\}} \sum_{l \in N \setminus S} c_{il} \Bigg). \end{aligned}\right\} \quad (5)$$

Exemple 2 : Considérons un problème avec 4 services et la matrice des c_{ij} suivante :

$$C = \begin{pmatrix} 0 & 3 & 4 & 6 \\ 3 & 0 & 7 & 1 \\ 4 & 7 & 0 & 5 \\ 6 & 1 & 5 & 0 \end{pmatrix}.$$

La résolution est présentée dans le tableau II. L'interprétation des quatre premières colonnes sera discutée à la section suivante :

TABLEAU II

J	$2 \times J$	Bin $(2 \times J)$	Bin (J^*)	S	f_S	Ordre
1	2	0010	0011	$\{1, 2\}$	9	1 | 2
2	4	0100	0101	$\{1, 3\}$	10,5	1 | 3
3	6	0110	0110	$\{2, 3\}$	6,5	2 | 3
4	8	1000	1001	$\{1, 4\}$	6,5	1 | 4
5	10	1010	1010	$\{2, 4\}$	10,5	2 | 4
6	12	1100	1100	$\{3, 4\}$	9	3 | 4
7	14	1110	1111	$\{1, 2, 3, 4\}$	13	2, 3 | 1, 4

La solution optimale, de valeur $z_2^* = 13$, consiste à ranger d'un côté les services 1 et 2, et de l'autre les services 4 et 3, dans cet ordre (ou à ranger ces mêmes ensembles de services dans l'ordre opposé).

5. CALCULS SUR ORDINATEUR

Comme c'est fréquemment le cas en programmation dynamique, la place occupée en mémoire est le facteur limitatif. La programmation de la formule (4) se fait de manière à ne nécessiter qu'un vecteur de taille $2^n - 1$, pour conserver les

valeurs de f_S. Ces valeurs sont calculées, suivant une méthode due à K. Baker [2] : on adopte pour les ensembles S successifs l'ordre du codage binaire correspondant aux entiers naturels (seconde colonne du tableau I). De la sorte, la valeur de $f_{S'}$ pour tout sous-ensemble S' de S est calculée avant celle de S. Le calcul d'une valeur de f_S correspondant à un J donné consiste à déterminer les valeurs 1 dans le codage binaire de J, à en déduire les valeurs J' obtenues en remplaçant un de ces 1 par 0 à la fois, à rechercher les valeurs de $f_{S'}$ pour S' correspondant à J' et à appliquer la formule (4).

TABLEAU III

	Problème 1		Problème 2	
Nombre de services	Temps algo Simmons (mn)	Temps algo 1 (mn)	Temps algo 2 (mn)	Temps algo Burkard-Derigs (mn)
4		0,003	0,003 0,003	0,004 0,004
5	0,003 0,003	0,004 0,004		
6	0,003 0,005 0,007	0,005 0,005	0,004 0,004 0,004	0,010 0,012
7	0,007 0,01 0,017			
8	0,031 0,057 0,1	0,015 0,013	0,016 0,015	0,242 0,226 0,092 (1)
9	0,077 0,391 1	0,029 0,031		
10	0,5 1,067	0,076	0,090 0,090	0,739 0,939
11	2,33 8	0,164		
12		0,407	0,704 0,634	> 60 9,303
14		2,881	4,883	> 60

(1) Exemple pris par R. Burkard et U. Derigs.

Pour le problème 2, on n'utiliserait qu'une position sur deux en appliquant la méthode précédente car seuls les nombres pairs de pièces doivent être considérés. Pour éviter cette perte de place, on multiplie J par 2, on en cherche l'équivalent binaire Bin $(2 \times J)$, et si le nombre de valeurs 1 est impair, on remplace la dernière valeur, nulle car $2 \times J$ est pair, par 1. (Bin J^*, dans la quatrième colonne du tableau II). Il est aisé de voir qu'on obtient ainsi une bijection entre les entiers naturels et les sous-ensembles S de N avec $|S|$ pair. Le calcul de f_S se fait alors avec des étapes semblables à celles de ce calcul pour le problème 1.

Une série de problèmes de tailles croissantes ont été résolus sur un ordinateur IRIS 80 C.I.I.-H.B. Les temps de calculs, en minutes, figurent au tableau III. Les temps obtenus par D. Simmons [8] sur I.B.M. 360-20/65, sont présentés également, ainsi que les temps de calculs obtenus sur IRIS 80 avec le code de R. Burkard et U. Derigs [4] pour le problème d'affectation quadratique.

Les temps obtenus par D. Simmons varient avec la difficulté du problème pour un nombre n de services donné; ils sont inférieurs à ceux de l'algorithme 1 pour les plus petits problèmes, mais deviennent plus élevés pour les plus grands; un problème difficile avec $n = 10$ n'a pu être résolu par l'algorithme de Simmons, la capacité de la mémoire étant dépassée. La croissance du temps de calcul de l'algorithme 2, pour le problème du rangement des services des deux couloirs, est légèrement plus élevée que celle de l'algorithme 1. Les temps obtenus avec le code de R. Burkard et U. Derigs, qui n'exploite pas la structure particulière du problème 2, sont beaucoup plus élevés.

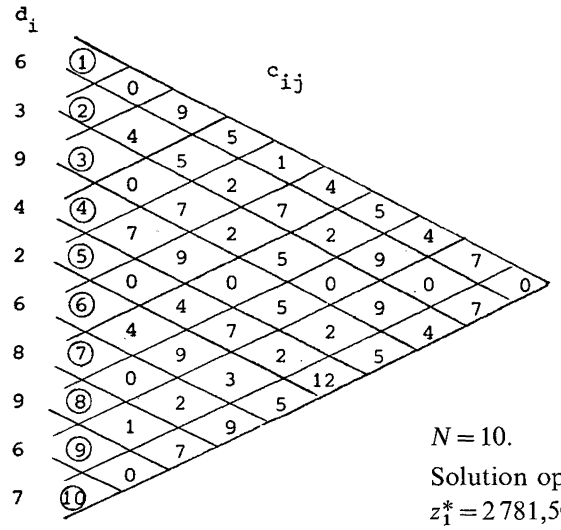
6. CONCLUSION

Les deux problèmes d'affectation de locaux à des services étudiés sont *NP*-complets et proches de problèmes connus pour leur difficulté pratique de résolution comme le problème d'affectation quadratique. Les algorithmes proposés permettent de résoudre des problèmes avec un nombre de services à localiser n modéré, mais comparable au nombre maximal d'unités qui peuvent être localisées de manière optimale en un temps raisonnable par les algorithmes actuellement disponibles pour le problème d'affectation quadratique. Pour de plus grands problèmes, les algorithmes pourraient être appliqués à des sous-ensembles de l'ensemble des services à ranger dans une méthode d'approximations successives. Notons enfin que les modèles et les algorithmes peuvent aisément être modifiés pour prendre en compte les usagers arrivant de l'une ou de l'autre extrémité du couloir.

ANNEXE 1

DONNÉES ET RÉSULTATS DES PROBLÈMES DE GRANDE TAILLE RÉSOLUS PAR L'ALGORITHME 1

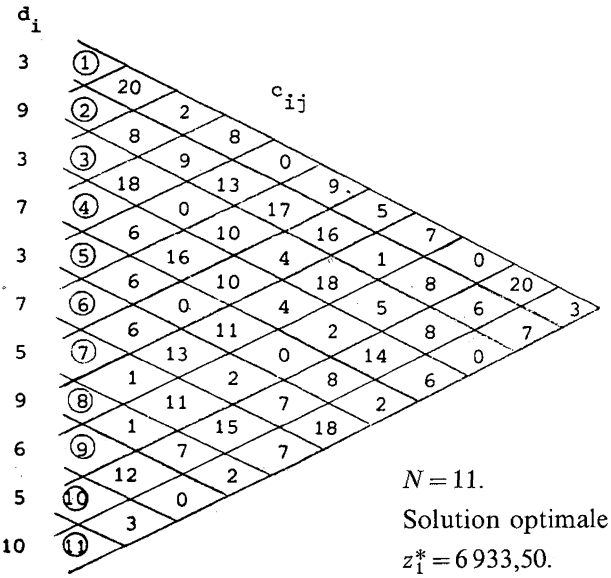
Ce sont, jusque $N = 11$, ceux qui sont proposés par Simmons [8].



$N = 10$.

Solution optimale : 9-3-1-7-5-10-4-2-6-8.

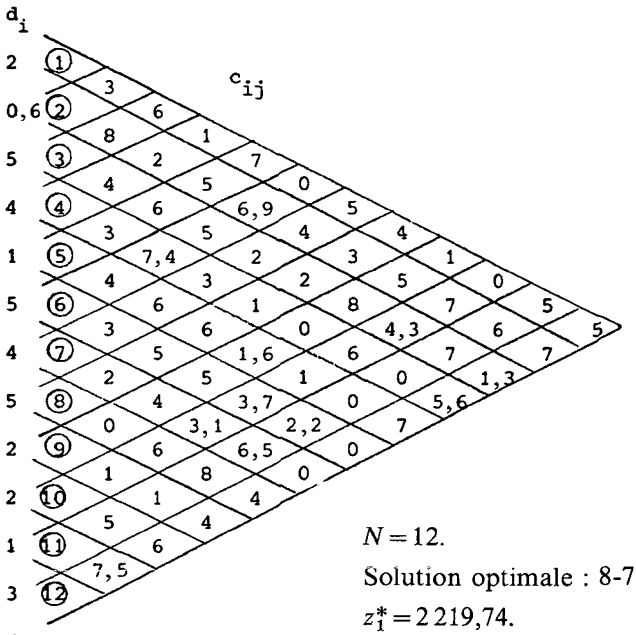
$z_1^* = 2\,781{,}50$.



$N = 11$.

Solution optimale : 11-8-5-6-3-4-10-1-2-7-9.

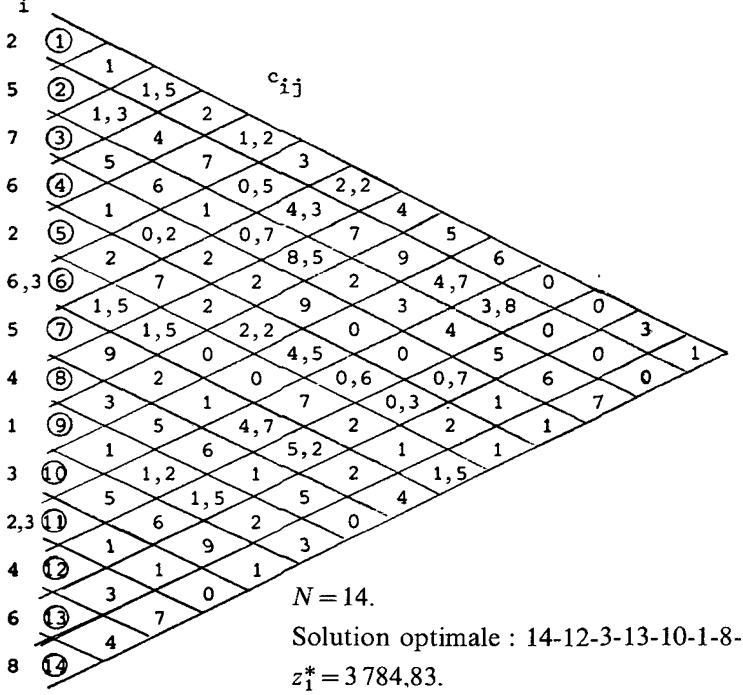
$z_1^* = 6\,933{,}50$.



$N = 12$.

Solution optimale : 8-7-1-5-11-12-2-10-9-3-6-4.

$z_1^* = 2\,219{,}74$.



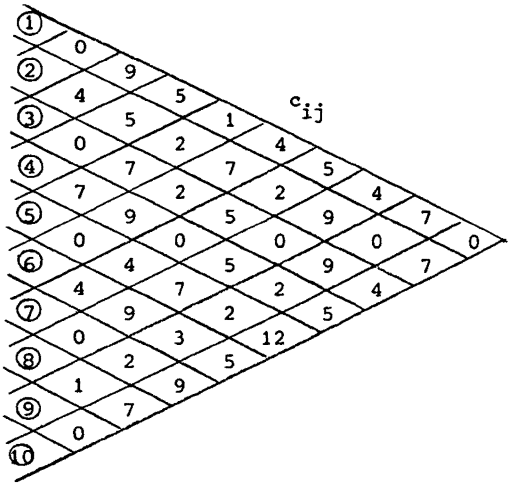
$N = 14$.

Solution optimale : 14-12-3-13-10-1-8-5-7-11-9-2-4-6.

$z_1^* = 3\,784{,}83$.

ANNEXE 2

DONNÉES ET RÉSULTATS DES PROBLÈMES DE GRANDE TAILLE RÉSOLUS PAR L'ALGORITHME 2, ET PAR LA MÉTHODE DE BURKARD ET DERIGS

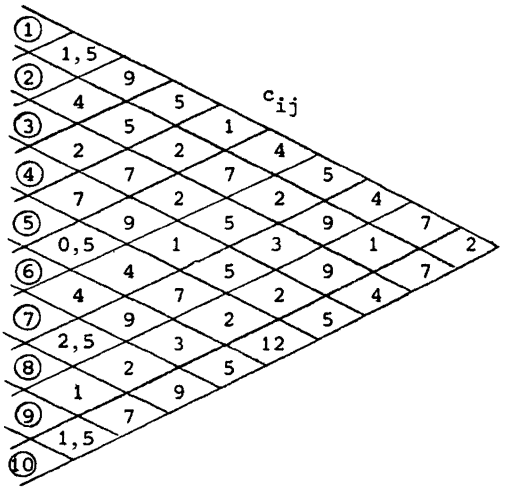


$N = 10$.

Solution optimale :

2	4	5	3	1
8	6	10	7	9

$z_2^* = 236$.

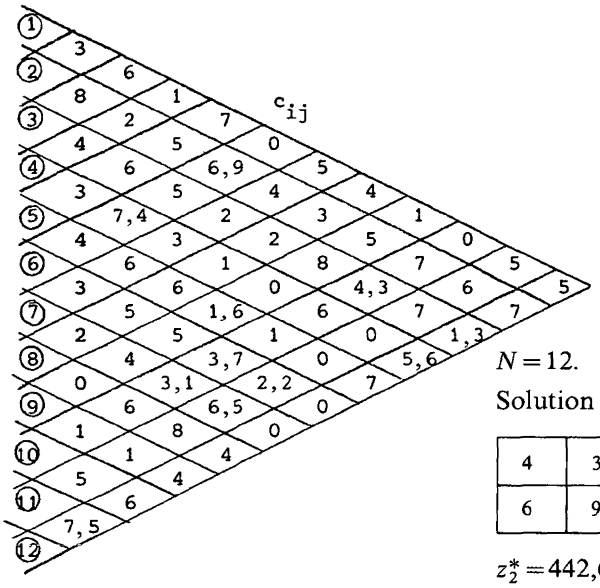


$N = 10$.

Solution optimale :

4	2	5	3	1
6	8	10	7	9

$z_2^* = 274{,}50$.

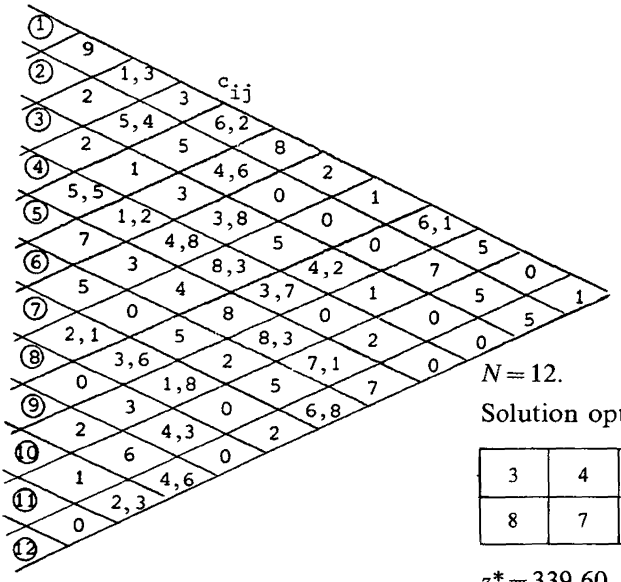


$N = 12$.

Solution optimale :

4	3	2	11	5	1
6	9	10	12	8	7

$z_2^* = 442{,}60$.

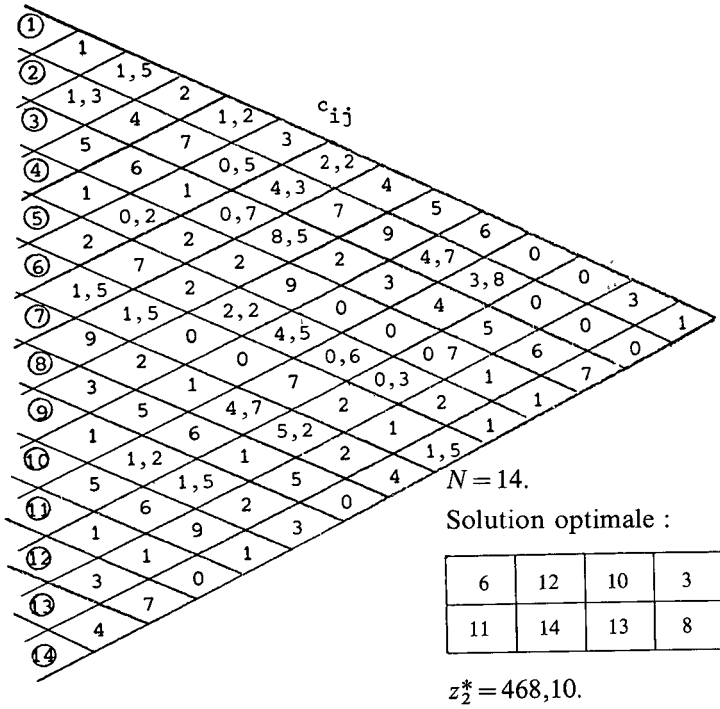


$N = 12$.

Solution optimale :

3	4	9	5	1	10
8	7	11	6	2	12

$z_2^* = 339{,}60$.



$N = 14$.

Solution optimale :

6	12	10	3	5	1	4
11	14	13	8	7	2	9

$z_2^* = 468{,}10$.

BIBLIOGRAPHIE

1. D. ADOLPHSON et T. C. HU, *Optimal Linear Ordering*, S.I.A.M. J. on Applied Math., vol. 25, 1973, p. 403-423.
2. K. R. BAKER, *Introduction to Sequencing and Scheduling*, New York, Wiley, 1974.
3. R. BELLMAN, *Dynamic Programming*, Princeton, Princeton University Press, 1957.
4. R. E. BURKARD et U. DERIGS, *Assignment and Matching Problems: Solution Methods with Fortran-Programs*, vol. 184, Lecture Notes in Economics and Mathematical Systems, Springer Verlag, Berlin, Heidelberg, New York.
5. M. GAREY et D. S. JOHNSON, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, San Francisco, Freeman, 1979.
6. M. GAREY, D. S. JOHNSON et L. STOCKMEYER, *Some Simplified NP-Complete Graph Problems*, Theoretical Computer Science, vol. 1, 1976, p. 237-267.
7. M. HELD et R. M. KARP, *Finite-State Processes and Dynamic Programming*, S.I.A.M. J. on Applied Math., vol. 15, 1967, p. 693-718.
8. D. M. SIMMONS, *One Dimensional Space Allocation: An Ordering Algorithm*, Operations Research, vol. 17, 1969, p. 812-826.
9. D. M. SIMMONS, *A Further Note on One-Dimensional Space Allocation*, Operations Research, vol. 19, 1971, p. 249.