RAIRO-Oper. Res. 42 (2008) 389–400
DOI: 10.1051/ro:2008014

RAIRO Operations Research
www.rairo-ro.org

UNE NOUVELLE MÉTHODE D'INITIALISATION POUR LE PROBLÈME DE TRANSPORT

François Dubeau[1] et Oumar Mandione Guèye[2]

Abstract. A new initialization method for the transportation problem is presented. It assigns value only if necessary. It gives good results and often the optimal solution.

Résumé. Dans cet article nous proposons une nouvelle méthode d'initialisation du problème de transport classique. Cette méthode est basée sur le principe d'une affectation seulement si nécessaire. Elle donne de bons résultats et souvent la solution optimale.

Mots Clés. Problème de transport, problème de Hitchcock, méthode du simplexe, initialisation.

Classification Mathématique. 90C08, 90C10, 90C59.

Introduction

Depuis le milieu du siècle dernier plusieurs chercheurs se sont intéressés à trouver une méthode d'initialisation permettant de trouver une solution initiale le plus près possible de la solution optimale pour le problème de transport, aussi connu sous le nom de problème de Hitchcock. Les méthodes d'initialisation les plus connues à ce jour pour ce genre de problèmes sont la méthode du coin Nord-Ouest [3,4], la méthode de Vogel [10], la méthode de Russell [9] et des variantes de ces méthodes [1,2,5–8,11,12]. Dans ce travail, nous proposons une nouvelle méthode permettant de trouver une solution proche de l'optimum.

Reçu le 1er juillet 2005. Accepté le 19 février 2008.

[1] Département de mathématiques, Faculté des sciences, Université de Sherbrooke, 2500 boul. de l'Université, Sherbrooke (Qc), J1K 2R1, Canada ; Francois.Dubeau@USherbrooke.ca

[2] Faculté des sciences, Collège universitaire de Saint-Boniface, 200 avenue de la Cathédrale, Winnipeg (Mb), R2H 0H7, Canada ; ogueye@ustboniface.mb.ca

La section 1 servira à présenter le problème de transport. À la section 2 nous ferons un bref rappel sur les méthodes classiques d'initialisation du problème de transport, les méthodes du coin Nord-Ouest, de Vogel et de Russell. Ensuite à la section 3 nous présenterons la nouvelle méthode appelée la méthode des Demandes et Offres Résiduelles (DOR). Nous donnerons l'algorithme et un exemple numérique. Une étude comparative avec les méthodes classiques d'initialisation de la section 2 est présentée à la section 4. Cette section montre également la sensibilité de notre méthode face à une étape de réaffectation. Enfin la section 5 concluera cet article.

1. Formulation du problème

Le problème de transport consiste à déterminer un réseau de distribution permettant d'expédier à moindre coût une certaine quantité d'un produit de m origines $I = \{1, ..., m\}$ à n destinations $J = \{1, ..., n\}$. Sous forme standard le problème de transport s'écrit

$$(P) \quad \begin{cases} & \min z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} & \\ \text{s.à.} & & \\ & \sum_{j=1}^{n} x_{ij} = a_i & (i \in I), \\ & \sum_{i=1}^{m} x_{ij} = b_j & (j \in J), \\ & x_{ij} \geq 0 & (i \in I \text{ et } j \in J), \end{cases}$$

où a_i représente l'offre à l'origine i, b_j représente la demande à la destination j, x_{ij} est la quantité expédiée de l'origine i à la destination j, c_{ij} est le coût unitaire de transport de l'origine i à la destination j. On appelle *solution réalisable* ou *programme réalisable* tout ensemble de valeurs non-négatives des variables x_{ij} qui satisfont aux contraintes d'offre et de demande du problème (P). Une *solution réalisable de base* peut s'écrire sous la forme $x = (x_B, x_N)$ où x_B contient $m + n - 1$ composantes et $x_N = 0$. Comme conséquence du théorème des écarts complémentaires, cette solution est *optimale* si et seulement si il existe des variables duales u_i $(i \in I)$ et v_j $(j \in J)$ telles que

$$\begin{cases} x_{ij} \in x_B \Rightarrow c_{ij} - u_i - v_j = 0, \\ x_{ij} \in x_N \Rightarrow c_{ij} - u_i - v_j \geq 0. \end{cases}$$

Pour trouver une solution optimale au problème de transport on commence par déterminer une solution réalisable et ensuite on améliore cette solution jusqu'à l'obtention de la solution optimale. Afin de diminuer le nombre d'étapes d'amélioration de la solution réalisable, il est intéressant d'utiliser des méthodes d'initialisation qui donnent une solution très proche de la solution optimale. Nous exposons brièvement quelques méthodes classiques d'initialisation du problème de transport avant d'en proposer une nouvelle.

2. Méthodes classiques d'initialisation

Les méthodes d'initialisation les plus connues sont toutes basées sur le même principe décrit ci-dessous.

Procédure d'initialisation :

Étape 1 : Déterminer une case (i, j) de la grille désignée par la méthode utilisée.

Étape 2 : Comparer a_i et b_j et aller à (2.1), (2.2) ou (2.3) selon le cas.

2.1 : Si $a_i < b_j$: $x_{ij} = a_i$, remplacer b_j par $b_j - x_{ij}$ et a_i par 0, et supprimer la ligne i.

2.2 : Si $a_i > b_j$: $x_{ij} = b_j$, remplacer a_i par $a_i - x_{ij}$ et b_j par 0, et supprimer la colonne j.

2.3 : Si $a_i = b_j$, affecter $x_{ij} = a_i = b_j$ et poser $a_i = b_j = 0$; s'il reste encore de la marchandise non distribuée, alors supprimer au choix la ligne i ou la colonne j, laissant ainsi la possibilité d'introduire, à une étape ultérieure, un zéro dans une case de la rangée non supprimée (dans ce cas la solution de base est dégénérée).

Étape 3 : Répéter l'algorithme sur le tableau réduit jusqu'à ce que toute la marchandise soit distribuée.

Saturant une rangée à chaque étape, nous obtiendrons une solution d'au plus $m + n - 1$ variables non nulles. Il y a m lignes et n colonnes qu'on sature l'une après l'autre, sauf lors de la dernière affectation où une ligne et une colonne sont saturées simultanément.

2.1. La méthode du coin du Nord-Ouest [3,4]

Certainement la méthode la plus facile à mettre en œuvre et de complexité minimale, seulement en $\mathcal{O}(m + n)$. Elle consiste à déterminer la variable de plus petits indices où il est possible d'affecter une quantité. Sa simplicité l'a rendue très populaire. Par contre, la solution trouvée n'a aucune raison d'être près de la solution optimale.

2.2. La méthode de Vogel [10]

Cette méthode mise sur la minimisation d'un système de pénalités. On appelle la pénalité d'une rangée, la différence de coût entre la case de plus petit coût et celle du deuxième plus petit coût de la rangée.

Étape 1 : Détermination d'une case (i, j) de la grille.

1.1 : Évaluer la différence entre les deux plus petits coûts de chaque ligne et de chaque colonne (pénalité).

1.2 : Identifier la rangée correspondant à la plus grande pénalité.

1.2.1 : Si cette rangée est unique alors choisir la case (i, j) la moins coûteuse de cette rangée.

1.2.2 : Si la pénalité maximale est associée à plusieurs lignes ou colonnes alors, l'un ou l'autre des cas suivants se présente :

(a) si le coût minimal d'une des lignes [ou colonnes] concernées est aussi le coût minimal de sa colonne [ligne] alors choisir la case (i, j) ainsi désignée ;

(b) si aucun des coûts ne répond aux exigences décrites en (a), alors calculer les « pénalités secondes » des rangées touchées par la même pénalité maximale. La pénalité seconde d'une ligne (colonne) est donnée par la différence entre le deuxième plus petit coût de cette ligne (colonne) et le plus petit coût de la colonne (ligne) où apparaît ce coût. Si, dans une des rangées, il existe plusieurs cases de coût égal au deuxième plus petit coût, alors calculer toutes les pénalités secondes associées à ces cases. Choisir alors la case (i, j) de moindre coût dans la rangée pour laquelle la pénalité seconde prend la plus grande valeur.

Étapes 2 et 3 : Ces étapes de la procédure d'initialisation sont effectuées en recalculant les pénalités à chaque itération.

La complexité de cette méthode est $\mathcal{O}(mn\ln(mn) + (m+n)^2)$. En effet, au début il faut ordonner, par valeurs croissantes, les n coûts pour chacune des m lignes et les m coûts pour chacune des n colonnes, ce qui donne $\mathcal{O}(mn\ln(mn))$. Ensuite, à chaque itération et dans le pire des cas, il y a un calcul des $m+n$ différences sur les lignes et les colonnes, cela donne une complexité de $\mathcal{O}(m+n)$ par itération. Comme il y a au plus $m+n$ itérations, le résultat suit.

2.3. La méthode de Russell [9]

Cette méthode repose sur une comparaison des programmes primal et dual. Elle commence par estimer les u_i et v_j en prenant respectivement le maximum des coûts pour chaque origine i et le maximum des coûts pour chaque destination j et ce pour tout i et tout j. Ensuite, la méthode calcule un coût réduit pour chacune des cases en soustrayant de son coût initial, la valeur des u_i et v_j correspondant.

Étape 1 : Détermination d'une case (i, j) de la grille.

1.1 : Pour chaque ligne i non saturée, calculer $u_i = \max_j c_{ij}$.

1.2 : Pour chaque colonne j non saturée, calculer $v_j = \max_i c_{ij}$.

1.3 : Calculer $c^*_{ij} = c_{ij} - u_i - v_j$.

1.4 : Choisir la case de moindre coût réduit c^*_{ij}. S'il y a égalité, choisir la case de moindre coût c_{ij} et, s'il y a à nouveau égalité, choisir celle dont le $\min\{a_i, b_j\}$ est le plus grand.

Étapes 2 et 3 : Ces étapes de la procédure d'initialisation sont effectuées en recalculant les u_i, v_j et c^*_{ij} à chaque itération.

Comme pour la méthode de Vogel, au début il faut ordonner, par valeurs croissantes, les coûts des lignes et des colonnes, ce qui donne $\mathcal{O}(mn\ln(mn))$. Ensuite, à chaque itération et dans le pire des cas, il y a un calcul de tous les c^*_{ij} sur les

lignes et les colonnes non saturées, ce qui donne $\mathcal{O}(mn)$. Comme il y a au plus $m+n$ itérations, le nombre d'opérations est au total $\mathcal{O}(mn\ln(mn)+mn(m+n))$, d'où $\mathcal{O}(mn(m+n))$.

3. Nouvelle méthode d'initialisation : la méthode des demandes et offres résiduelles (DOR)

Avec la méthode d'initialisation que nous proposons, au lieu de trouver les cases où il serait avantageux d'affecter une quantité non nulle maximale, nous trouvons les cases où il est préférable d'affecter la valeur minimale qu'on peut mettre dans cette case. À la fin nous obtenons une solution de base réalisable.

3.1. La méthode DOR

On amorce l'algorithme en affectant des zéros aux cases les plus coûteuses pour notre programme de minimisation jusqu'à ce qu'on soit obligé d'affecter des quantités non nulles à certaines cases moins coûteuses. Pour pouvoir vérifier s'il est possible d'affecter un zéro à une case, on devra tout au long de l'application de l'algorithme conserver une demande résiduelle pour chacune des origines et une disponibilité résiduelle pour chacune des destinations.

Représentons par : B_i la demande résiduelle à l'origine i (cette demande résiduelle est la somme des demandes des destinations pouvant possiblement être comblées par cette origine i) ; A_j l'offre résiduelle pour la destination j (cette offre résiduelle est la somme des offres des origines pouvant servir à satisfaire la demande de la destination j) ; J_i l'ensemble des destinations j telles que x_{ij} n'est pas encore fixée ou, de façon équivalente, l'ensemble des destinations que l'origine i peut encore desservir ; I_j l'ensemble des origines i telles que x_{ij} n'est pas encore fixée ou, de façon équivalente, l'ensemble des origines qui peuvent encore desservir la destination j.

Procédure DOR :

Étape 1 : Initialisation.

- Poser
  - $J_i := J$, $B_i := \sum_{j=1}^{n} b_j$ pour $i \in I$,
  - $I_j := I$, $A_j := \sum_{i=1}^{m} a_i$ pour $j \in J$.
- Pour tous $i \in I$ et $j \in J$, calculer
  - $u_i := \min_j c_{ij}$ et $v_j := \min_i c_{ij}$,
  - $c^*_{ij} := c_{ij} - u_i - v_j$.

Étape 2 : Choisir, parmi les cases sans affectation, la case de plus grand c^*_{ij}.

- S'il y a égalité choisir la case de plus grand c_{ij} ; s'il y a de nouveau égalité, choisir la case ayant le plus petit $\min\{a_i, b_j\}$; s'il y a encore égalité, utiliser un autre critère de choix.
- Notons la case choisie par ses indices (l, c).

Étape 3 : Affectation à la case (l, c) choisie.

- Modifier

- $B_l := B_l - b_c$ et $J_l := J_l \setminus \{c\}$,
- $A_c := A_c - a_l$ et $I_c := I_c \setminus \{l\}$.

• Affecter à la case (l, c) la valeur $x_{lc} := \min\{\max\{0, a_l - B_l, b_c - A_c\}, a_l, b_c\}$.
• Poser $a_l := a_l - x_{lc}$ et $b_c := b_c - x_{lc}$.
• Mettre à jour les demandes et offres résiduelles :
- pour toute ligne $i \in I_c$, poser $B_i := B_i - x_{lc}$,
- pour toute colonne $j \in J_l$, poser $A_j := A_j - x_{lc}$.

Étape 4 : Affectation forcée avec priorité.

4.1 : Affectation nulle
• s'il existe une ligne $i \in I$ avec $a_i = 0$ et $J_i \neq \emptyset$, alors pour chaque $j \in J_i$ poser $x_{ij} := 0$ et $I_j := I_j \setminus \{i\}$, et poser $B_i := 0$ et $J_i := \emptyset$;
• s'il existe une colonne $j \in J$ avec $b_j = 0$ et $I_j \neq \emptyset$, alors pour chaque $i \in I_j$ poser $x_{ij} := 0$ et $J_i := J_i \setminus \{j\}$, et poser $A_j := 0$ et $I_j := \emptyset$.

4.2 : Offre et demande égales pour une rangée
• s'il existe une ligne $i \in I$ avec $a_i = B_i > 0$, alors
- pour chaque $j \in J_i$, poser
  * $x_{ij} := b_j$ et $b_j := 0$,
  * pour chaque $k \in I_j \setminus \{i\}$, poser $x_{kj} := 0$, $J_k := J_k \setminus \{j\}$ et $B_k := B_k - b_j$,
  * $A_j := 0$ et $I_j := \emptyset$;
- poser $a_i := 0$, $B_i := 0$ et $J_i := \emptyset$;

• s'il existe une colonne $j \in J$ avec $b_j = A_j > 0$, alors
- pour chaque $i \in I_j$, poser
  * $x_{ij} := a_i$ et $a_i := 0$,
  * pour chaque $k \in J_i \setminus \{j\}$, poser $x_{ik} := 0$, $I_k := I_k \setminus \{i\}$ et $A_k := A_k - a_i$,
  * $B_i := 0$ et $J_i := \emptyset$;
- poser $b_j := 0$, $A_j := 0$ et $I_j := \emptyset$.

4.3 : Case unique non affectée sur une rangée
• s'il existe une ligne $i \in I$ telle que $J_i = \{j\}$, alors
- poser $x_{ij} := \min\{a_i, b_j\}$,
- si $a_i \leq b_j$, poser
  * $b_j := b_j - x_{ij}$, $A_j := A_j - x_{ij}$ et $I_j := I_j \setminus \{i\}$,
  * $a_i := a_i - x_{ij} = 0$, $B_i := 0$ et $J_i := J_i \setminus \{j\} = \emptyset$,
  * pour toute ligne $k \in I$ telle que la case (k, j) n'a pas d'affectation et $B_k \neq 0$, poser $B_k = B_k - x_{ij}$,
- sinon $a_i > b_j$ et poser
  * $b_j := b_j - x_{ij} = 0$, $A_j := 0$ et $I_j := \emptyset$,
  * $a_i := a_i - x_{ij}$, $B_i := 0$ et $J_i := \emptyset$;
  * pour toute ligne $k \in I$ telle que la case (k, j) n'a pas d'affectation, poser
    · $x_{kj} := 0$,

· $B_k := B_k - x_{ij}$,
· $J_k := J_k \setminus \{j\}$;
∗ réaffectation : tant que $a_i > 0$, choisir la colonne $t \in J$ telle que $b_t \neq 0$ et de coût c^*_{it} minimum, calculer $\Delta x_{it} = \min\{a_i, b_t\}$ et poser
· $x_{it} := x_{it} + \Delta x_{it}$,
· $a_i := a_i - \Delta x_{it}$,
· $b_t := b_t - \Delta x_{it}$,
· $B_s := B_s - \Delta x_{it}$ pour tout $s \in J_t \setminus \{i\}$;

• s'il existe une colonne $j \in J$ telle que $I_j = \{i\}$, alors
– poser $x_{ij} := \min\{a_i, b_j\}$
– si $b_j \leq a_i$, poser
∗ $b_j := b_j - x_{ij} = 0$, $A_j := 0$ et $I_j := I_j \setminus \{i\} = \emptyset$,
∗ $a_i := a_i - x_{ij}$, $B_i := B_i - x_{ij}$ et $J_i := J_i \setminus \{j\}$,
∗ pour toute colonne $k \in J$ telle que la case (i, k) n'a pas d'affectation et $A_k \neq 0$, poser $A_k = A_k - x_{ij}$;
– sinon $b_j > a_i$ et poser
∗ $b_j := b_j - x_{ij}$, $A_j := 0$ et $I_j := \emptyset$,
∗ $a_i := a_i - x_{ij} = 0$, $B_i := 0$ et $J_i := \emptyset$;
∗ pour toute colonne $k \in J$ telle que (i, k) n'a pas d'affectation, poser
· $x_{ik} := 0$,
· $A_k := A_k - x_{ij}$,
· $I_k := I_k \setminus \{i\}$;
∗ réaffectation : tant que $b_j > 0$, choisir la ligne $t \in I$ telle que $a_t \neq 0$ et de coût c^*_{tj} minimum, calculer $\Delta x_{tj} = \min\{a_t, b_j\}$ et poser
· $x_{tj} := x_{tj} + \Delta x_{tj}$,
· $a_t := a_t - \Delta x_{tj}$,
· $b_j := b_j - \Delta x_{tj}$,
· $A_s := A_s - \Delta x_{tj}$ pour tout $s \in I_t \setminus \{j\}$.

Étape 5 : S'il n'y a plus d'affectation forcée à l'étape 4, retour à l'étape 2.

Il est important de respecter la priorité à l'étape 4. Dès qu'une affectation de type 4.1 est possible elle est prioritaire face aux affectations de types 4.2 et 4.3. De même, une affectation de type 4.2 passe avant une affectation de type 4.3.

La complexité de cette méthode est en $\mathcal{O}(mn(m+n))$. En effet il y a $\mathcal{O}(mn)$ calculs d'initialisation des c^*_{ij}. Ensuite il y a un classement de ces mn valeurs, ce qui donne $\mathcal{O}(mn \ln(mn))$. Il peut y avoir au plus mn itérations. À chaque itération, s'il y a affectation d'un 0 il y a deux mises à jour, sinon le nombre de mises à jour peut être égal au nombre de lignes ou au nombre de colonnes non saturées et il y a saturation d'une nouvelle rangée, ce qui donne au maximum $\mathcal{O}(mn(m+n))$. Pour la réaffectation, qui ne peut s'effectuer qu'au plus $m+n$ fois, il peut y avoir jusqu'à mn mises à jour, ce qui donne encore un $\mathcal{O}(mn(m+n))$. Nous obtenons donc une complexité de $\mathcal{O}(mn + mn\ln(mn) + mn(m+n)) = \mathcal{O}(mn(m+n))$.

TABLEAU 1. Tableau initial de l'exemple numérique.

	Destination j							
					v_j			
Origine i / u_i	1 / 56	2 / 23	3 / 9	4 / 8	5 / 12	a_i	B_i	J_i
1 / 9	73 / 8	40 / 8	9 / -9	79 / 62	20 / -1	8	32	{1,...,5}
2 / 8	62 / -2	93 / 62	96 / 79	8 / -8	13 / -7	7	32	{1,...,5}
3 / 50	96 / -10	65 / -8	80 / 21	50 / -8	65 / 3	9	32	{1,...,5}
4 / 12	57 / -11	58 / 23	29 / 8	12 / -8	87 / 63	3	32	{1,...,5}
5 / 12	56 / -12	23 / -12	87 / 66	18 / -2	12 / -12	5	32	{1,...,5}
b_j	6	8	10	4	4			
A_j	32	32	32	32	32			
I_j	{1,...,5}	{1,...,5}	{1,...,5}	{1,...,5}	{1,...,5}			

TABLEAU 2. Solution de l'exemple numérique.

	Destination j				
Origine i	1	2	3	4	5
1	0_8	0_9	8_{11}	0_4	0_{12}
2	0_{14}	0_5	0_1	4_{17}	3_{16}
3	5_{23}	4_{22}	0_7	0_{18}	0_{13}
4	1_{20}	0_6	2_{10}	0_{19}	0_3
5	0_{24}	4_{25}	0_2	0_{15}	1_{21}

3.2. UN EXEMPLE NUMÉRIQUE

En guise d'exemple numérique traitons le problème décrit au tableau 1. Dans chacune des cases (i, j) du tableau nous représentons en diagonale le coût c_{ij} et le coût réduit c^*_{ij}. Pour ce problème les 10 premières affectations se font selon les étapes 2 et 3 de la procédure DOR. Ensuite il y a une affectation forcée avec une étape 4.2. Les affectations 13 à 16 se font selon les étapes 2 et 3. Alors il doit y avoir une affectation forcée nécessitant l'étape 4.2. Suite à cette affectation, il y a deux cas de case unique non affectée réglés à l'aide de l'étape 4.3 à deux reprises. L'affectation 22 se fait selon les étapes 2 et 3 et les dernières affectations nécessitent l'étape 4.2. Le tableau 2 donne la solution obtenue avec la méthode DOR. Nous

avons indiqué en indice l'ordre d'affectation des cases. Notons que cette solution est non seulement une solution de base réalisable mais est une solution optimale et vaut 1102. La méthode de Russell donne une solution non-optimale de 1103 tandis que la méthode de Vogel en donne une de valeur 1104.

4. Résultats numériques : comparaison et sensibilité

Nous comparons la performance de la méthode DOR à celle des méthodes de Vogel et de Russell. Nous noterons également la sensibilité de cette méthode à la réaffectation de l'étape 4.3.

4.1. Caractéristiques des problèmes tests

Les tests numériques sont effectués sur un échantillon de 40 problèmes dont les dimensions sont données au tableau 3. Certains de ces problèmes sont tirés de la littérature, d'autres ont été générés aléatoirement. Les problèmes 1 à 23 sont de petite taille puisqu'ils comportent moins de 100 variables et les problèmes 24 à 40 sont de grande taille puisqu'ils comportent au moins 100 variables. Les énoncés des problèmes sont disponibles auprès du second auteur.

4.2. Comparaison avec les méthodes de Vogel et de Russell

Le tableau 3 donne la valeur de l'objectif et le nombre d'itérations pour chacune des trois méthodes : Vogel, Russell et DOR. Nous présentons en caractères gras la meilleure solution parmi celles données par les trois méthodes. Le signe * indique que la solution est optimale, l'optimalité ayant été vérifiée par la méthode du simplexe via CPLEX.

Nous observons que pour les problèmes de petite taille notre approche semble mieux se comporter que les deux autres méthodes. En effet nous observons qu'elle donne une meilleure solution pour 19 des 23 problèmes de petite taille. De plus elle donne une solution optimale pour 15 problèmes contre 5 pour la méthode de Vogel et 10 pour la méthode de Russell. Nous en déduisons ainsi que de ce point de vue notre méthode est supérieure aux deux autres méthodes et que dans beaucoup de cas elle permet d'obtenir une solution optimale. Pour les problèmes de plus grande taille, nous remarquons que notre méthode semble être à efficacité égale avec la méthode de Russell. Elle est supérieure à la méthode de Vogel pour 11 problèmes tandis que la méthode de Russell est supérieure à la méthode de Vogel pour 9 des problèmes. Nous en déduisons que de ce point de vue la méthode DOR est meilleure que la méthode de Vogel et est competitive avec la méthode de Russell sur des problèmes de grande taille.

Le nombre d'itérations pour les trois méthodes augmente avec la taille du problème. Notre approche requiert en général plus d'itérations avant d'obtenir une solution. Par contre cette remarque qui *a priori* peut sembler la désavantager n'a pas un gros impact sur le temps d'exécution. Le tableau 4 donne les temps d'exécution des différentes méthodes et le nombre d'itérations effectuées par CPLEX

TABLEAU 3. Comparaison des différentes méthodes.

Problème			**CPLEX**	**Vogel**		**Russell**		**DOR**			
Pb	n orig	m dest	Valeur opt	Nb iter	Valeur obj	Nb iter	Valeur obj	Nb iter	Nb réaf	Valeur obj	Écart %
P 1	3	3	1763	5	1801	5	**1763***	3	0	**1763***	0
P 2	3	3	537	5	547	5	541	4	0	**537***	0
P 3	3	4	103	5	**103***	6	**103***	5	0	**103***	0
P 4	3	4	12	6	**12***	6	**12***	6	0	**12***	0
P 5	4	3	785	6	**785***	6	**785***	4	0	**785***	0
P 6	3	4	80	6	82	6	82	6	0	**80***	0
P 7	3	4	254	6	262	6	262	6	0	**254***	0
P 8	3	4	425	6	**425***	6	**425***	6	0	430	1.2
P 9	3	5	86	6	90	7	**86***	8	0	**86***	0
P10	4	5	2063	8	**2105**	8	2133	11	1	2111	2.3
P11	4	6	111	8	112	9	113	15	0	**111***	0
P12	4	6	5605	8	**5605***	9	**5605***	13	0	**5605***	0
P13	5	5	1102	8	1104	9	1103	15	0	**1102***	0
P14	6	5	338	10	380	10	402	20	0	**338***	0
P15	6	5	2340	10	2378	10	**2340***	9	0	2358	0.8
P16	5	7	121	11	136	11	**121***	23	0	**121***	0
P17	7	7	271	12	287	13	403	32	0	**271***	0
P18	7	7	271	12	281	12	**271***	32	0	**271***	0
P19	7	9	459	15	491	14	498	34	0	**471**	2.6
P20	9	7	367	15	491	15	380	38	0	**374**	1.9
P21	9	7	459	15	632	15	589	44	0	**467**	1.7
P22	8	8	382	14	400	14	426	44	0	**388**	1.6
P23	7	10	179	15	203	16	**189**	49	4	206	15.1
P24	10	10	426	18	436	16	453	76	2	**427**	0.2
P25	10	10	148		16 151	18	172	73	3	**148***	0
P26	13	15	1444	25	1566	27	1764	134	1	**1450**	0.4
P27	10	20	1024	29	1069	29	1403	151	0	**1024***	0
P28	15	20	1208	33	**1287**	33	1767	228	7	1333	10.3
P29	20	15	2179	32	2406	33	2328	194	0	**2308**	5.9
P30	20	25	3278	44	**3679**	42	3839	304	5	4086	24.6
P31	25	20	856	39	925	40	945	442	9	**867**	1.3
P32	20	30	4676	49	5213	47	**4719**	450	33	6375	36.3
P33	25	25	2055	49	2257	49	**2206**	448	8	2247	9.3
P34	25	35	13711	59	16978	59	**15063**	489	11	15169	10.6
P35	20	40	70477	59	75645	58	**73184**	544	20	74461	5.7
P36	30	40	7220	57	9847	58	**9002**	695	39	9068	25.6
P37	25	40	60565	64	68068	64	**64188**	717	34	69031	13.9
P38	30	40	61033	69	77561	69	**63533**	952	21	77721	27.3
P39	35	40	79070	74	105937	74	87972	953	13	**80162**	1.4
P40	40	40	86334	79	**88977**	79	99093	1091	50	103781	20.2

TABLEAU 4. Comparaison des temps d'exécution en seconde.

Problème	**Vogel**		**Russell**		**DOR**	
No	temps init	CPLEX iter	temps init	CPLEX iter	temps init	CPLEX iter
P24	0.00	5	0.00	9	0.00	1
P25	0.00	2	0.00	11	0.00	0
P26	0.00	23	0.00	35	0.02	3
P27	0.00	9	0.00	27	0.01	0
P28	0.01	6	0.01	21	0.03	6
P29	0.01	17	0.00	11	0.01	8
P30	0.01	27	0.01	27	0.03	30
P31	0.01	11	0.01	13	0.03	2
P32	0.01	15	0.01	5	0.05	27
P33	0.01	13	0.01	6	0.04	9
P34	0.02	37	0.02	13	0.07	11
P35	0.02	47	0.02	30	0.05	33
P36	0.01	37	0.02	24	0.06	25
P37	0.02	31	0.02	28	0.08	31
P38	0.02	41	0.03	33	0.12	38
P39	0.04	41	0.03	35	0.15	7
P40	0.04	13	0.04	19	0.19	21

pour obtenir la solution optimale à partir de la solution initiale fournie par la méthode d'initialisation. Nous ne donnons pas les temps pour les 23 premiers problèmes car pour ces problèmes de petite taille le temps d'exécution est approximativement le même et est négligeable pour les trois méthodes. Pour les problèmes de taille plus grande, nous observons que notre approche est plus sensible à la taille du problème que les deux autres méthodes. En ce qui concerne le nombre d'itérations fait par CPLEX, la méthode DOR est tout à fait compétitive.

4.3. SENSIBILITÉ À LA RÉAFFECTATION

Rappelons qu'à l'étape 4 une réaffectation est faite lorsque l'unique case non affectée sur une rangée est remplie et qu'il reste une offre ou une demande non satisfaite. Le tableau 3 illustre l'impact de cette réaffectation sur la qualité de la solution. La dernière colonne donne le pourcentage d'écart entre la valeur de la solution obtenue par la méthode DOR et la valeur optimale du problème. Bien que la réaffectation se fasse en utilisant les coûts c^*_{ij} les plus faibles, nous remarquons qu'elle tend à détruire la qualité de la solution. En effet nous observons que pour les cas où il n'y a pas de réaffectation la solution obtenue par la méthode DOR est souvent, mais pas toujours, optimale. Ceci est plus fréquent pour les problèmes de petite taille. Inversement, il peut arriver qu'il y ait des réaffectations et qu'on obtienne quand même la solution optimale. Par ailleurs pour les problèmes où l'écart

entre la valeur de la solution initiale et la solution optimale est assez important, nous remarquons en général un nombre plus élevé de réaffectations.

5. Conclusion

Une nouvelle méthode d'initialisation pour le problème de transport a été proposée dans cet article. Pour des problèmes de petite taille, la solution optimale est souvent atteinte alors que pour des problèmes de grande taille nous obtenons des solutions proches de l'optimum et comparables ou meilleures que les solutions obtenues par d'autres méthodes classiques d'initialisation.

Références

[1] R.L. Ackoff, E.L. Arnoff and C.W. Churchman, *Operations Research*. Wiley, New York (1957).
[2] N. Balakrishnan, Modified vogel's approximation method for the unbalanced transportation problem. *Appl. Math. Lett.* **3** (1990) 9–11.
[3] G.B. Dantzig, *Application of the simplex method to a transportation problem*, Chap. XXIII of Cowles Commission Monograph, No. 19. Wiley, New York (1951).
[4] G.B. Dantzig, *Linear Programming and Extensions*. Princeton University Press, Princetion, N.J. (1963).
[5] S.K. Goyal, Improving VAM for unbalanced transportation problems. *JORS* **35** (1984) 1113–1114.
[6] H.S. Houthakker, On the numerical solution of the transportation problem. *Oper. Res.* **3** (1955) 210–214.
[7] R.E. Larson, Normalizing Vogel's approximation method. *Math. Mag.* Nov-Dec (1972) 266–269.
[8] T.S. Lee, A complete Russell's Method for the transportation problem. *SIAM Rev.* **28** (1986) 547–549.
[9] E.J. Russell, Extension of Dantzig's algorithm to finding an initial near-optimal basis for the transportation problem. *Oper. Res.* **3** (1955) 210–214.
[10] N.V. Reinfeld and W.R. Vogel, *Mathematical Programming*. Prentice-Hall, Englewood Cliffs, N.J. (1958).
[11] W. Swarc, The initial solution of the transportation problem. *Oper. Res.* **8** (1960) 727–729.
[12] D.G. Shimshak, J.A. Kaslik and T.D. Barclay, A modification of Vogel's approximation method through the use of heuristics. *INFOR* **19** (1981) 256–263.