7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France
« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

UNE METAHERISTIQUE HYBRIDE POUR LE PORBLEME DE TOURNEES DE VEHICULE AVEC CONTRAINTES DE CAPACITE(CVRP)

L. BOUHAFS, A. HAJJAM

Laboratoire Systèmes et Transport, Université de Technologie de Belfort-Montbéliard,
90010 Belfort Cedex France
Lyamine.Bouhafs@utbm.fr, Amir.Hajjam@utbm.fr

A. KOUKAM

Laboratoire Systèmes et Transport, Université de Technologie de Belfort-Montbéliard,
90010 Belfort Cedex France
Abder.koukam@utbm.fr

RESUME : *Le problème des tournées de véhicules (VRP) est une classe très connue des problèmes d'optimisation combinatoire NP-difficile. En raison de leur complexité, plusieurs heuristiques et métaheuristiques ont été proposées pour les résoudre. Les méthodes exactes, quand à elles, montrent des limites en terme de temps de calcul pour trouver les solutions dés lors qu'il s'agit d'instances de grandes tailles. Dans cet article, nous proposons un algorithme hybride à base de colonies de fourmis et du recuit simulé pour résoudre le VRP. Les résultats d'expérimentations montrent la qualité des solutions obtenues par notre approche.*

MOTS-CLES : *Problème de tournées de véhicules, métaheuristique, Algorithmes de colonies de fourmis, Recuit simulé.*

1. INTRODUCTION

Le problème des tournées de véhicules (VRP) est une classe très connue des problèmes d'optimisation combinatoire NP-difficile. Le VRP consiste à planifier les routes d'une flotte homogène de véhicules, stationnés au dépôt central, pour servir un ensemble de clients avec des demandes connues. Dans la version de base du VRP, connue sous le nom de CVRP (problème des tournées de véhicules avec contraintes de capacité), seules les contraintes de capacité des véhicules sont considérées. L'objectif est de minimiser le coût total des tournées (longueur des routes parcourues par les véhicules). L'importance du VRP se montre dans ses applications tant elles sont nombreuses. La plupart de ces applications se concentrent sur des problèmes dans des domaines tels que la distribution.

En raison de la complexité des problèmes des tournées de véhicules (NP-difficile), plusieurs heuristiques et métaheuristiques ont été proposées pour les résoudre. Les méthodes exactes, quand à elles, montrent des limites en terme de temps de calcul pour trouver les solutions dés lors qu'il s'agit d'instances de grandes tailles. Nous nous sommes intéressés aux métaheuristiques utilisées pour la résolution des VRP et particulièrement aux colonies de fourmis.

Le premier algorithme à base de colonies de fourmis, appliqué au CVRP, a été proposé par [Bullnheimer et al. 1998] dans sa forme de base « Ant System » (AS), appliqué la première fois pour le TSP, de [Dorigo et al. 1996]. Il utilise la phéromone et l'heuristique du plus proche voisin pour la construction des routes de véhicules. Pour l'amélioration de ces derniers, une heuristique de 2-OPT est combinée avec l'AS. Ils ont utilisé également un renforcement par élitisme pour les pistes de phéromones. L'algorithme a été testé sur une quarantaine de benchmark, mais les meilleures solutions n'ont pas pu être améliorées. De même, les performances de l'algorithme sont inférieures à celles de la recherche tabou. Par contre, en terme de temps d'exécution, l'AS est un rival sérieux pour les autres métaheuristiques.

Les mêmes auteurs [Bullnheimer et al. 1999] ont proposé un AS amélioré en remplaçant l'heuristique de plus proche voisin, dans la règle de transition de l'algorithme de base (AS), par l'heuristique de Savings de [Paessens 1988]. Pour la mise à jour des pistes de phéromones, le même procédé utilisé dans leur premier algorithme a été appliqué. A la fin de chaque itération, une heuristique de recherche locale (2-OPT) a été appliquée pour améliorer les routes. Une deuxième heuristique de recherche locale est utilisée pour la sélection des clients. Pour chaque client, on trie par ordre croissant des distances, la liste des clients qui ne sont pas encore visités. Les tests effectués confirment l'amélioration des résultats obtenus par rapport à la première version de l'algorithme de [Bullnheimer et al. 1998]. Cependant, l'algorithme montre encore des limites, car les meilleures solutions publiées sont rarement atteintes.

L'algorithme proposé par [Doerner et al. 2002] est presque identique à l'algorithme de [Bullnheimer et al. 1999]. La seule différence est que celui-ci combine l'AS avec l'heuristique de Savings de [Calrke & Wright 1964]. Les résultats trouvés par cet algorithme n'améliorent pas d'une manière significative les résultats des approches précédentes. Contrairement à toutes ces méthodes fondées sur «Ant System», notre approche, [Bouhafs et al. 2004], est basée sur la nouvelle version des colonies de fourmis « Ant Colony System» [Dorigo & Gambardella 1997] proposée pour améliorer le «Ant System» et notamment pour les problèmes de grandes instances.

Dans cet article, nous proposons une amélioration de notre approche [Bouhafs et al. 2004] en intégrant un algorithme de recuit simulé dans la phase de mise à jour des phéromones de l'algorithme de colonie de fourmis.
Nous présentons dans la section 2 la formulation du CVRP. La section 3 est consacrée à la description de l'algorithme proposé pour la résolution du CVRP. La section 4 décrit les résultats expérimentaux et la section 5 conclue les travaux.

2. FORMULATION DU CVRP

Le CVRP peut être représenté sous la forme d'un graphe complet orienté et valué $G = (V, A)$, où :
$V = \{v_0, v_1, v_2, ..., v_n\}$ représente l'ensemble des sommets et $A = \{(v_i, v_j) : i \neq j\}$ représente l'ensemble des arcs entre les sommets.
Le sommet v_0 représente le dépôt et les autres représentent les clients. A chaque arc (v_i, v_j) est associé une valeur non négative d_{ij}. Cette valeur correspond à la distance entre le sommet v_i et le sommet v_j, au coût ou au temps entre les deux sommets. Une demande q_i et un temps de service δ_i $(q_0 = 0, \delta_0 = 0)$ sont associés à chaque client (sommet) v_i. L'objectif, dans ce cas, est de minimiser le coût total des tournées tout en respectant les contraintes suivantes : (1) Chaque client est visité une et une seule fois par un seul véhicule, (2) Toutes les routes des véhicules commencent et se terminent par un dépôt, (3) La demande totale des clients de chaque route ne doit pas dépasser la capacité de chaque véhicule.
Le nombre de véhicules est supposé illimité, il est calculé pendant la construction des tournées.
La figure 2.1 montre la représentation graphique d'un exemple du CVRP.

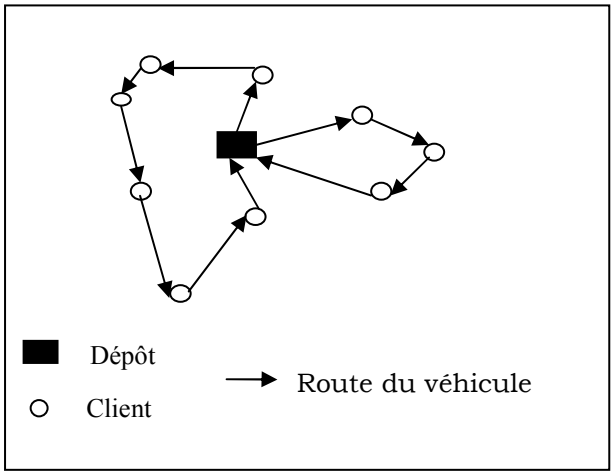

Figure 2.1 Exemple de solution du CVRP

Nous donnons, ici, la formulation de CVRP trouvée dans la littérature. Mais avant cela, nous décrivons la liste des variables qui nous aiderons dans la formulation :

- $G = (V, A)$
- $V = \{v_0, v_1, v_2, ..., v_n\}$ où v_0 représente le dépôt et $v_1, ..., v_n$ l'ensemble des clients
- q_i la demande du client i, $i \in V$
- d_{ij} la distance (coût) entre les sommets v_i et v_j
- $K = \{k_1, k_2, ..., k_m\}$ représente la flotte des véhicules
- Q la capacité de chaque véhicule $k_i \in K$ (la flotte est homogène)

Pour trouver l'ordre de visite des clients, nous définissons les variables de décision comme suit :

$$x_{i,j}^k = \begin{cases} 1 & \textit{si le véhicule v visite le client j directement après le client} \\ 0 & \textit{sinon} \end{cases}$$

$$y_i^k = \begin{cases} 1 & \textit{si le client i est desservi par le véhicule k} \\ 0 & \textit{sinon} \end{cases}$$

La fonction objectif est :

$$Min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} d_{ij} x_{i,j}^k \qquad (2.1)$$

Avec les contraintes :

$$\sum_{k \in V} \sum_{j \in V} x_{i,j}^k = 1, \quad \forall i \in V \qquad (2.2)$$

$$\sum_{j \in V} x_{i,j}^k - \sum_{j \in V} x_{j,i}^k = 0, \ \forall i \in V, k \in K \qquad (2.3)$$

$$\sum_{j \in V} x_{0,j}^k = 1, \ \forall k \in K \qquad (2.4)$$

$$\sum_{j \in V} x_{j,n+1}^k = 1, \ \forall k \in K \qquad (2.5)$$

$$x_{i,j}^k = 1 \Rightarrow y_i - q_j = y_j, \ \forall i, j \in V, k \in K \qquad (2.6)$$

$$y_0 = Q, \ 0 \leq y_i, \ \forall i \in V \qquad (2.7)$$

$$x_{i,j}^k \in \{0,1\}, \ \forall i, j \in V, k \in K \qquad (2.8)$$

La fonction de coût euclidien de la solution $X = \left(x^k{}_{ij}\right)$, $\forall i, j \in V, k \in K$ est définie par :

$$coût(X) = \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} d_{ij} x_{i,j}^k \qquad (2.9)$$

Le nombre de véhicules utilisés par la solution X, est défini par :

$$Nb\ véhicules(X) = \sum_{k \in K} \sum_{i \in V} x_{0,j}^k \qquad (2.10)$$

3. ALGORITHME HYBRIDE A BASE DE COLONIES DE FOURMIS ET DE RECUIT SIMULE POUR LE CVRP

L'algorithme que nous proposons est basé sur un ACS hybride avec un algorithme de Savings et une recherche locale (2-opt). Nous introduisons dans la règle de transition l'heuristique de Savings qui permet d'enrichir la recherche et de calculer l'utilité de combiner deux clients dans une même route ou de les mettre dans deux routes différentes. Le 2-opt est appliqué à chaque définition des tournées par les fourmis, juste avant la règle de mise à jour globale des phéromones. Le recuit simulé intervient dans la phase de mise à jour des phéromones où deux règles sont appliquées comme dans le « Ant Colony System » (ACO) introduit dans [Dorigo & Gambardella 1997]. Avant de développer les différentes étapes de l'algorithme, nous expliquons le principe de fonctionnement des deux heuristiques 2-opt et Savings.

3.1 L'heuristique 2-OPT

Le principe de 2-OPT est de supprimer deux arcs d'une même route et de les remplacer par deux autres arcs dans le but d'améliorer le coût de cette route et d'éliminer les croisements. La figure ci-dessous montre le principe de cette heuristique.

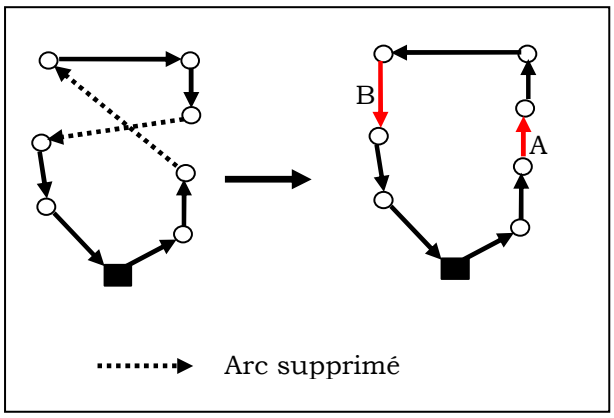

Figure 3.1 Principe de l'heuristique de recherche locale 2-OPT

Dans la figure 3.1, les deux arcs en pointillés sont supprimés dans le graphe gauche. Ils sont remplacés par les deux arcs étiquetés (A et B) dans le graphe droite pour trouver une nouvelle route. Dans le cas des graphes orientés, le sens de certains arcs, qui ne sont pas concernés par une opération 2-opt, peut être modifié lors de la construction de la nouvelle route.

3.2 Heuristique de Savings

C'est une heuristique proposée par [Clarcke & Wright 1963] et améliorée par [Paessens 1988]. Elle est la base de la plupart des logiciels commerciaux pour la résolution des problèmes des tournées de véhicules dans les applications industrielles. L'objectif de cette heuristique est de déterminer s'il est préférable de combiner les clients v_i et v_j dans une même route (ceci quand la valeur de γ_{ij} est grande) ou de les mettre dans deux routes différentes. La valeur de Savings des clients v_i et v_j est calculée comme suit :

$\gamma_{ij} = d_{i0} + d_{0j} - g.d_{ij} + f.\left|d_{i0} - d_{0j}\right|$ où :

- d_{ij} représente la distance entre le sommet i et le sommet j,
- l'indice 0 correspond au dépôt, g et f représentent deux paramètres de l'heuristique.

3.3 Description de l'Algorithme

3.3.1 Construction des routes

L'algorithme ACS de [Dorigo & Gambardella 1997], présenté la première fois pour résoudre le TSP, tient compte de la phéromone et de l'heuristique du plus proche voisin pour la construction des routes. Notre approche ACS hybride, présentée ici, tient compte de l'heuristique de Savings en plus pour cette construction.
Initialement dans l'ACS hybride, m fourmis sont positionnées sur l'ensemble des sommets du graphe et une quantité de phéromone initiale est appliquée sur les arcs. Chaque fourmi prend le départ à partir du dépôt pour visiter les clients. Chaque client est visité une et une seule fois par une fourmi, mais le dépôt peut être visité plusieurs fois. Si la charge accumulée d'une fourmi dépasse la contrainte de capacité du véhicule, la fourmi doit retourner au dépôt. Nous obtenons alors une route complète pour un véhicule. Quand une fourmi retourne au dépôt, sa charge est remise à zéro. Elle initialise une autre route pour visiter d'autres clients qui ne l'ont pas encore été et ce, jusqu'à ce que tous les clients soient visités. Cela, signifie qu'une solution complète du CVRP a été trouvée.
Pendant la construction d'une route, la fourmi modifie la quantité de phéromone sur l'arc choisi en appliquant une règle de mise à jour locale. Dés que toutes les fourmis ont construit leurs tournées, la quantité de phéromone sur les arcs, appartenant à la meilleure tournée, est mise à jour selon la règle de mise à jour globale.
La règle utilisée pour la construction des routes est décrite ci-dessous. Une fourmi k, positionnée sur un nœud i, choisit le prochain sommet j à visiter en appliquant la règle probabiliste $p_k(i,j)$ donnée dans les équations (3.1) et (3.2).

$$j = \begin{cases} \arg\max_{u \in F_k(i)} \left\{ (\tau_{iu})^{\alpha} . (\eta_{iu})^{\beta} . (\gamma_{iu})^{\lambda} \right\} & si\ q \le q_0 \\ J & sinon \end{cases} \quad (3.1)$$

J est une variable aléatoire générée selon la fonction de distribution donnée par :

$$p_{ij} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha} . (\eta_{ij})^{\beta} . \ (\gamma_{ij})^{\lambda}}{\sum_{u \in F_k(i)} (\tau_{iu})^{\alpha} . (\eta_{iu})^{\beta} . \ (\gamma_{iu})^{\lambda}} & \text{si } u \in F_k(i) \\ 0 & \text{sinon} \end{cases} \quad (3.2)$$

Où :

- $F_k(i)$ est la liste des sommets qui ne sont pas encore visités pas la fourmi k positionnée au sommet i,
- q est une variable aléatoire qui suit une loi uniforme sur $[0,1]$,
- q_0 est un paramètre $(0 \le q_0 \le 1)$ qui détermine l'importance relative de l'exploitation versus l'exploration. Avant qu'une fourmi ne visite le prochain sommet, q est généré aléatoirement. Si $q \le q_0$ alors l'exploitation est encouragée, sinon le processus de l'exploration est encouragé,
- τ_{ij} est la quantité de phéromone associée à l'arc (i, j),
- η_{ij} est l'heuristique de visibilité qui est l'inverse de la distance entre les sommets i et j,
- $\gamma_{ij} = d_{i0} + d_{0j} - g.d_{ij} + f.\left|d_{i0} - d_{0j}\right|$ est l'heuristique de Savings où f et g sont deux paramètres,
- α, β et λ sont trois paramètres qui déterminent l'importance relative de la phéromone, de la distance et de Savings (respectivement).

3.3.2 Mise à jour des phéromones

La nouveauté dans notre algorithme est d'introduire le recuit simulé dans la phase de mise à jour des phéromones.

Pendant qu'une fourmi construit sa solution, le niveau de phéromone sur chaque arc (i, j) visité est modifié selon la règle de mise à jour locale suivante :

$$\tau_{ij}^{new} = (1 - \rho)\tau_{ij}^{old} + \rho \Delta \tau_{ij} \quad (3.3)$$

Où :

ρ est le paramètre d'évaporation $(0 < \rho < 1)$,

$\Delta \tau_{ij} = \tau_0$ est la quantité de phéromone initiale.

Dés que chaque fourmi a défini ses tournées, une recherche locale 2-opt est appliquée pour améliorer les routes des véhicules. Cette heuristique est appliquée à la fin de chaque itération et juste avant la règle de mise à jour globale des phéromones.

Dans l'algorithme ACS, seule la fourmi ayant trouvé la meilleure solution est autorisée à déposer une quantité de phéromone pour guider la recherche.
Dans notre algorithme, toute fourmi qui trouve une solution S' dont :
$\Delta S (= Cost(S') - Cost(S)) \prec 0$ Ou $\exp(-\Delta S / T) \succ \mu$ est autorisée à déposer une quantité de phéromone.
Où :

- S est la meilleure solution courante,
- μ est une variable aléatoire qui suit une distribution uniforme sur $[0,1]$,
- $Cost(S)$ est le coût de la solution S,
- $\exp(-\Delta S / T)$ est le critère de Metropolis utilisé pour le recuit simulé.
- T paramètre qui représente la température dans le recuit simulé

L'intégration de ce critère de choix dans la règle de mise à jour globale permet d'accepter certaines solutions qui ne sont pas forcément meilleures que la solution courante et ainsi de diversifier la recherche.

La règle de mise à jour globale est donnée dans l'équation (3.3).

Si l'arc (i, j) est utilisé par une fourmi et que sa solution est acceptée (répond au critère défini ci-dessus), alors la quantité de phéromone est augmentée sur cet arc par $\Delta \tau_{ij}$ qu'est égal à $1/L^*$ avec L^* est la longueur de la tournée trouvée par cette fourmi.

4. EXPERIMENTATIONS ET RESULTATS

Après la description de l'algorithme ACS hybride, nous évaluons sa performance en considérant un ensemble d'instances du problème CVRP. L'algorithme est codé en langage java et exécuté sur un Pc portable, équipé du système Windows XP, d'un processeur P4 de vitesse 2.4 GHz et d'une mémoire vive (RAM) de 512 Mo. Nous avons effectué notre étude expérimentale sur un ensemble d'instances de différentes tailles. Ces instances sont détaillées dans le tableau 4.1. Elles peuvent être regroupées en 3 classes selon leurs auteurs :
Les instances (A, B et P) : [Augerat et al. 1995], Les instances E : [Christofides & Eilon 1969], Les instances C : [Christofides et al. 1979].
Pour chaque instance, on donne le nombre n de clients et la capacité Q d'un véhicule.

Instance	n	Q	Instance	n	Q
B-n68-k9	68	100	C1	50	160
B-n78-k10	78	100	C2	75	140
B-n66-k9	66	100	C3	100	200
B-n50-k8	50	100	C4	150	200
B-n57-k9	57	100	C5	199	200
B-n63-k10	63	100	C11	120	200
B-n67-k10	67	100	C12	100	200
P-n51-k10	51	80	A-n63-k9	63	100
P-n76-k5	76	280	A-n63-k10	63	100
E-n76-k10	76	140	A-n69-k9	69	100
E-n76-k15	76	100	A-n80-k10	80	100
A-n60-k9	60	100			

Tableau 4.1 détails des instances de tests (n : nombre de client, Q : capacité d'un véhicule)

4.1 Réglage des paramètres de l'algorithme

Nous initialisons les paramètres de l'algorithme par les valeurs suivantes :

- Le nombre de fourmis $m = 10$ initialement placées aléatoirement sur l'ensemble des sommets,
- $\alpha = 1,\ \beta = 2\ \ \lambda = 2$ avec α , β et λ sont trois paramètres qui déterminent respectivement l'importance relative de la phéromone, de la distance et du Savings,
- $\rho = 0.1$ est le paramètre d'évaporation de la phéromone utilisé dans les règles de mise à jour des phéromones,
- $f = g = 2$ avec f et g les deux paramètres de l'heuristique de Savings,
- $q_0 = 0.75$ est le paramètre qui détermine l'importance relative de l'exploitation versus l'exploration. Avant qu'une fourmi ne visite le prochain sommet, q est générée aléatoirement. Si $q \le q_0$ alors l'exploitation est encouragée, sinon le processus de l'exploration est encouragé,
- $\tau_0 = 10^{-6}$ représente la quantité initiale de la phéromone déposée sur tous les arcs.
- $T = 100$ représente la valeur initiale de la température,
- Initialement le paramètre de la température T est T_0 $(T \leftarrow T_0)$,
- $\theta = 0.9$, à chaque itération la température est modifiée par la formule $(T \leftarrow \theta * T)$.

4.2 Résultats des tests

Dans ce paragraphe, nous présentons une étude des tests effectués de l'algorithme ACS hybride que nous avons proposé. Les résultats des tests pour les instances détaillées dans le tableau 4.1 sont reportés dans le Tableau 4.2. La première colonne du tableau représente le nom de l'instance traitée. La deuxième colonne précise la meilleure solution connue trouvée par les heuristiques (Dist : distances et Nbv : Nombre de véhicules). La colonne avec l'entête H_ACS donne la solution trouvée par notre approche. Enfin, la dernière colonne donne la déviation de notre solution par rapport à la meilleure solution publiée trouvée par d'autres heuristiques. Elle est calculée par la formule suivante :

$$déviation = \frac{Solution(H\_ACS) - Solution(meilleure\ connue)}{Solution(meilleure\ connue)} * 100$$

A partir du Tableaux 4.2, nous remarquons que les résultats trouvés par notre approche ont une déviation moyenne de (0.022%) par rapport aux meilleures solutions connues sur l'ensemble des tests. Cela, s'explique par le fait que les solutions trouvées par notre ACS Hybride sont très proches des meilleures solutions publiées. De plus, notre méthode permet même d'améliorer certaines solutions pour certaines instances (B-n68-k9, B-n78-k10, B-n66-k9, A-n60-k9, P-n51-k10).

Instance	Meilleure connue		H_ACS		déviation
	Dist.	Nbv.	Dist.	Nbv.	
B-n68-k9	1304	9	**1300.91**	9	-0,23%
B-n78-k10	1266	10	**1238.28**	10	-2,19%
B-n66-k9	1374	9	**1371.67**	9	-0,17%
B-n50-k8	1313	8	1317.34	8	0,30%
B-n57-k9	1598	9	**1598.0**	9	0%
B-n63-k10	1537	10	1540.71	10	0,19%
B-n67-k10	1033	10	1035.46	10	0,19%
A-n60-k9	1408	9	**1357.70**	9	-3,57%
A-n63-k9	1634	9	1636.94	9	0,18%
A-n63-k10	1315	10	1322.93	10	0,60%
A-n69-k9	1168	9	1177.36	9	0,80%
A-n80-k10	1764	10	1787.05	10	1,30%
P-n51-k10	745	10	**743.26**	10	-0,23%
P-n76-k5	631	5	631.0	5	0%
E-n76-k10	832	10	836.37	10	0,52%
E-n76-k15	1032	15	**1030**	15	-0,19 %

Tableau 4.2 Résultats des expérimentations pour le CVRP

Nous comparons également notre approche avec celle de [Bullnheimer et al. 1999] pour évaluer la qualité de notre solution et le temps de calcul. Les tests de comparaison, effectués sur les benchmarks de [Christofides et al. 1979], sont présentés dans le tableaux 3.1 (C1, C2, C3,

C4, C5, C11, C12). À l'initialisation de cette approche, N fourmis sont placées sur les sommets du graphe, N représente ainsi le nombre de clients. Pour chaque instance du problème, l'algorithme est exécuté $2 * N$ itérations.

Les résultats de comparaison sont reportés dans le tableau 4.3. La première colonne du tableau représente le nom de l'instance traitée. La colonne « best publ. » représente la meilleure solution connue dans la littérature. La troisième colonne précise la meilleure solution trouvée par l'AS de [Bullnheimer et al. 1999] (COST : coût de la solution, CPU : le temps de calcul en secondes et Dev. : la déviation par rapport à la meilleure solution connue). La colonne avec l'entête H_ACS donne la solution trouvée par notre approche après 500 itérations pour chaque instance.

Instance	best publ.	AS Bullnheimer et al. 1999]			H_ACS		
		COST	CPU (s)	Dev.	COST	CPU (s)	Dev.
C1	524.61	524.61	6	0%	524.61	2	0%
C2	835.26	844.31	78	1.08%	838.86	20	0.43%
C3	826.14	832.32	228	0.75%	834.77	35	1.04%
C4	1028.42	1061.55	4048	3.22%	1035.23	80	0.66%
C5	1291.45	1343.46	5256	4.03%	1304.25	200	0.99%
C11	1042.11	1065.21	552	2.22%	1046.81	32	0.45%
C12	819.56	819.56	300	0%	819.56	30	0%

Tableau 4.3 Test de comparaison entre l'algorithme ACS hybride et l'algorithme AS de [Bullnheimer et al. 1999]

A travers le tableau 4.3, nous déduisons que la déviation moyenne de notre approche (0,51%), par rapport aux meilleures solutions connues, est plus petite que celle de l'approche de [Bullnheimer et al. 1999] (1.61%) . Ceci prouve que notre approche réalise de meilleures performances. De plus, les solutions sont trouvées dans des temps de calcul nettement plus petits que ceux de [Bullnheimer et al. 1999]. Cependant, nous ne pouvons pas effectuer une comparaison directe en terme du temps de calcul car les approches sont testées sur des machines différentes.

En résumé, notre approche trouve des solutions de bonnes qualités dans de temps de calcul raisonnables.

5. CONCLUSION

Dans cet article, nous avons présenté un nouvel algorithme pour le CVRP. L'algorithme est basé sur les colonies de fourmis hybrides avec le recuit simulé. Les résultats d'expérimentations que nous avons exposés montrent la qualité des solutions obtenues par notre approche. L'objectif de cette étude est de montrer l'efficacité des algorithmes de colonies de fourmis hybrides pour les problèmes des tournées de véhicules. Cette étude nous a montré que la combinaison des colonies de fourmis avec le recuit simulé permet de trouver des résultats compétitifs.

REFERENCES

Bouhafs, L., Hajjam, A., Koukam, A.: "A Hybrid Ant Colony System Approach for the Capacitated Vehicle Routing Problem", Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, volume 3172 de Lecture Notes in Computer Science, pages 414–415. Springer, September 2004.

Bullnheimer, B., Hartl, R. F. and Strauss, Ch.: Applying the ant system to the vehicle routing problem. In: Voss, S., Martello, S., Osman, I. H. and Roucairol, C. (Eds.): Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston (1999).

Bullnheimer, B., Hartl, R. F. and Strauss, Ch.: An im proved ant system algorithmfor the vehicle routing problem. Annals of Operations Research 89 (1999) 319–328.

Christofides, N., Eilon, S.: An algorithm for the vehicle dispatching problem. Operational Research Quarterly 20 (3), (1969)309–318.

Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (Eds.), Combinatorial Optimization, Wiley, Chichester, (1979) pp. 315-338.

Clark, G., and Wright, J. W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 14, (1964) 568-581.

Dorigo, M., Maniezzo, V., and Colorni, A.: "Ant system: Optimization by a Colony of Cooperating Agents". IEEE Trans. Sys., Man, Cybernetics 26 (1996) 1, pages 29-41.

Dorigo, M., Gambardella, L.: Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem. IEEE Transactions on Evolutionary Computation, (1997) 1(1):53–66.

Dorigo,M., Di Caro,G., Gambardella, L.M.: "Ant algorithms for discrete optimization". In Artificial Life 5, pages 137–172, 1999.

Doerner, K., Gronalt, M., Hartl, R.F., Reimann, M., Strauss, C., and Stummer, M. (2002): "SavingsAnts for the Vehicle Routing Problem". In *Proceedings. (Lecture Notes in Computer Science, LNCS 2279). Kinsale, Ireland, April 3--5, 2002. Berlin [u.a.]: Springer. ISBN 3-540-43432-1.* pp. 11--20.

Paessens, H.: The savings algorithm for the vehicle rout ing problem, Eur. J. Oper. Res. 34, (1988) 336-344.

Aarts, E. H. L., Korst, J. H. M., and Laarhoven, P. J. M. V.: Simulated annealing. In Local Search in Combinatorial Optimization, E. H. L. Aarts and J. K. Lenstra, Eds. Wiley- Interscience, Chichester, England, (1997)91–120.

Czech , Z. J. and Czarnas, P.: "A Parallel Simulated An nealing for the Vehicle Routing Problem with Time Windows," Proc. 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands, Spain, (2002)376-383