Résolution de problèmes d'optimisation combinatoire par auto-organisation chez l'espèce de fourmis *Pachycondyla apicalis*

Mustapha Redouane Khouadjia[1]–Salim Chikhi[2] – Mohamed Batouche[3]

Faculté des sciences de l'ingénieur
Département d'Informatique
Université Mentouri
25000, Constantine
ALGERIE
[1]mkhouadjia@yahoo.fr, [2]slchikhi@yahoo.com,[3]batouche@ccis.edu.sa

RÉSUMÉ. Dans cet article nous présentons une nouvelle métaheuristique basée sur des comportements de recherche de nourriture chez les fourmis. Ces comportements s'inspirent de l'auto-organisation chez l'espèce de fourmis Pachycondyla apicalis. Ils se résument en des interactions simples, et parallèles de l'ensemble des fourmis de la colonie. La métaheuristique proposée est implémentée sur la plate-forme dédiée à la conception et à l'implémentation de métaheuristiques ParadisEO. Elle est appliquée sur le problème du voyageur de commerce. Les résultats sont très encourageants, et mettent en évidence la théorie de l'auto-organisation comme nouvelle approche pour la conception de métaheuristiques.

MOTS-CLÉS : Métaheuristiques bio-inspirées, algorithmes des colonies de fourmis, fourmis Pachycondyla aplicalis, auto-organisation, Programmation à mémoire adaptative, plate-forme ParadisEO.

ABSTRACT. In this paper we present a new metaheuristic based on artificial ants foraging behaviors. These behaviors are inspired from the self-organization of Pachycondyla apicalis ant species. They are summarized in simples and parallels interactions between ants. The proposed metaheuristic is implemented using ParadisEO which is a dedicated platform for the design and implementation of metaheuristics. Our metaheuristic is applied on travelling salesman problem. The results are very encouraging, and bring to light self-organisation theory as a novel approach for the design of metaheuristics.

KEYWORDS: Bio-inspired metaheuristics, Ant colony algorithms, Pachycondyla apicalis Ants, self-organisation, Adaptative memory programming, ParadisEO framework.

1. Introduction

La biologie est la source d'inspiration de nombreuses métaheuristiques. Une contribution importante de la biologie dans ce domaine vient de la *théorie de l'auto-organisation* [4]. Cette théorie étudiée dans et en dehors de la biologie décrit les conditions d'apparition de phénomènes complexes à partir de systèmes distribués, dont les agents font l'objet d'interactions simples, mais nombreuses. La théorie met en avant des concepts tels que la communication, les rétroactions, l'amplification des fluctuations et l'émergence. D'autre part, l'observation faite que les métaheuristiques récentes tendaient à devenir proches a conduit à mettre un cadre général du mode de fonctionnement de ces métaheuristiques. Ce cadre est proposé par la *programmation à mémoire adaptative* (PMA) [8]. Ce concept tente de décrire les points forts des métaheuristiques modernes, en insistant notamment sur le rôle de la mémoire et des mécanismes d'intensification et de diversification.

Le travail présenté dans cet article met en relation la théorie de l'auto-organisation et le concept de PMA. Cette combinaison donne des clefs pour concevoir les composants de base de métaheuristiques relevant de l'intelligence en essaim. Dans ce but, nous proposons une métaheuristique s'inspirant du comportement de fourragement chez l'espèce de fourmis *Pachycondyla apicalis*. Nous avons proposé plusieurs modèles comportementaux s'appuyant sur l'auto-organisation chez cette espèce de fourmis. Ces comportements permettent selon le concept de PMA de faire basculer la stratégie de recherche de notre métaheuristique entre intensification et diversification. Les résultats obtenus sur le problème du voyageur de commerce sont meilleurs que ceux obtenus par d'autres métaheuristiques modélisant la même espèce de fourmis tel que l'algorithme API [5].

Cet article est structuré comme suit : Nous décrivons d'abord dans la section 2 les principaux concepts mis en avant dans les théories de l'auto-organisation et de la programmation à mémoire adaptative, et nous dégageons les relations entre ces deux théories du point de vue des métaheuristiques. Dans cette démarche, nous proposons dans la section 3 la métaheuristique PAO (*Pachycondyla apicalis Ant Optimization)*. La section 4 donne les résultats des expérimentations menées sur différents benchmarks du voyageur de commerce. Enfin, on termine par une conclusion et des perspectives s'inscrivant dans la continuité de notre travail.

2. Théories de l'auto-organisation et le concept de programmation à mémoire adaptative

2.1. Auto-organisation

L'auto-organisation caractérise un processus au cours duquel une *structure* émerge au niveau global uniquement d'un grand nombre d'interactions entre les composants du *niveau local* du système. Elle désigne un arrangement organisé d'entités dans l'espace et dans le temps. De plus, les règles spécifiant les interactions entre les composants du système sont suivies en utilisant uniquement des informations locales [4].

L'auto-organisation est mise en place à l'aide de plusieurs mécanismes. Parmi eux les processus de rétroaction, et la gestion des flux d'information.

Les rétroactions positive*s* sont des processus dont le résultat renforce l'action, par exemple par amplification, auto-catalyse. Les rétroactions positives sont capables d'amplifier les fluctuations du système, permettant la mise à jour d'informations peu apparentes. De tels processus peuvent facilement entraîner une divergence du système, s'ils ne sont pas maintenus sous contrôle par des rétroactions négatives, qui jouent ainsi le rôle de stabilisateurs du système. Les flux d'information permettent d'expliquer la façon dont l'information circule au sein du système. Lorsqu'ils sont couplés, de tels processus sont de puissants générateurs de modèles auto-organisés [4].

2.2. Programmation à mémoire adaptative

Le concept de programmation à mémoire adaptative se veut une forme de généralisation du mode de fonctionnement des métaheuristiques efficaces qui tentent d'équilibrer la balance entre diversification et intensification en s'appuyant sur une mémoire qui oriente la recherche vers l'optimum global [8].

La *mémoire* représente ici l'information récoltée par l'algorithme sur laquelle il va s'appuyer pour effectuer sa recherche. Cette mémoire s'adapte aux nouvelles connaissances acquises en cours de recherche.

L'*intensification* consiste en l'utilisation des informations disponibles pour améliorer la pertinence de celles-ci. Du point de vue des métaheuristiques, il s'agit généralement tout simplement de recherche locale.

La *diversification* est la recherche de nouvelles informations afin d'augmenter la connaissance du problème.

2.3. Bases communes

L'auto-organisation nous renseigne sur la structure à employer pour concevoir des métaheuristiques flexibles capables de s'adapter à un problème donné.

La programmation à mémoire adaptative nous renseigne quant à elle sur les méthodes employées par les métaheuristiques efficaces. Du point de vue de la conception des métaheuristiques, il existe une relation simple entre elles : la PMA décrit le "but" à atteindre, et la théorie de l'auto-organisation un "moyen" pour atteindre ce but. Ainsi, une métaheuristique efficace devrait selon la programmation à mémoire adaptative, mettre en place des mécanismes de mémoire, d'intensification et de diversification, reste la question des moyens à utiliser pour mettre en place ces mécanismes. L'auto-organisation propose un modèle de réalisation : un algorithme à base de population définissant des interactions simples au niveau local, permettant au niveau global la structuration de la population dans l'espace de recherche.

3. Une métaheuristique inspirée de l'auto-organisation chez l'espèce de fourmis Pachycondyla apicalis

3.1. Auto-organisation et stratégie de fourragement

Pachycondyla apicalis est une ponérine néotropicale que l'on rencontre en Amérique du Sud. Ces fourmis vivent en petites colonies comportant quelques dizaines d'individus (40 à 100 ouvrières) [2,6]. Leur stratégie d'exploration globale se résume ainsi :

De sorties en sorties, les fourmis s'éloignent du nid en essayant de couvrir au mieux leur espace de recherche que constitue le voisinage du nid.

La stratégie de recherche locale privilégie quant à elle une exploration individuelle. Cette dernière se déroule ainsi :

– Si la fourmi capture une proie, elle retourne directement au nid en mémorisant le chemin emprunté;

– Après une capture, la fourmi utilise le chemin qu'elle a mémorisé pour retourner au site de capture ;

– Le retour sur le site de chasse se soldant par un échec peut se produire plusieurs fois de suite ;

– Un site de chasse ne produisant plus le renforcement que constitue la capture d'une proie est abandonné par la fourrageuse ;

– La fourmi a tendance à s'attacher à son meilleur site de chasse (i.e. celui qui lui a rapporté plusieurs captures de proies durant ses recherches). Il est à noter que même si celui-ci ne lui rapporte plus de proies, elle aura tendance à y retourner périodiquement.

Le nid est installé dans de vieilles souches ou des arbres morts en décomposition qui offrent ainsi un habitat instable pour des fourmis qui ne savent pas construire de fourmilière. Quand la vétusté de leur nid devient trop importante, elles doivent déménager et chercher un nouveau refuge. Des fourmis éclaireuses partent alors à la recherche d'un nouvel abri. Le déménagement s'effectue grâce à des mécanismes de recrutement en tandem, où une fourmi se fait guider par l'une de ses congénères jusqu'au nouvel emplacement. Ce changement de nid a pour effet de réinitialiser la recherche des fourrageuses.

3.2. Modélisation de la colonie pour traiter les problèmes d'optimisation

Nous allons considérer une population de n fourmis fourrageuses $a_1, \ldots, a_n$ de l'espèce *Pachycondyla apicalis*. Ces agents sont positionnés dans l'espace de recherche, noté S, et vont tenter d'optimiser une fonction d'évaluation f définie sur S.

Chaque point $s \in S$ est une solution valide du problème, ce qui signifie que f est définie en tout point de S. Cette métaheuristique n'impose aucune limitation sur l'espace de recherche.

Nous introduisons deux opérateurs d'exploration :

– Un opérateur O_g d'exploration globale. Cet opérateur permet de générer un point s à partir de l'emplacement du nid $\mathcal{N}$. Ainsi, on peut générer tous les points possibles de l'espace de recherche à partir de $\mathcal{N}$. Il est toute fois possible de limiter l'exploration au voisinage de $\mathcal{N}$. La taille du voisinage de $\mathcal{N}$ est alors paramétrée par une amplitude d'exploration notée $A_{globale}$, telle que $A_{globale} \in [0,1]$.

– Un opérateur de voisinage O_l. Cet opérateur nous permet de générer un point s' dans le voisinage d'un point s. La taille du voisinage de s est paramétrée par une amplitude, notée A_{locale} telle que $A_{locale} \in [0,1]$. Cette amplitude fixe la portée de l'exploration autour de s relativement à la taille de l'espace.

Chaque fourmi a_i est caractérisée par son site de chasse courant $s(a_i)$, ainsi qu'une liste de p sites de chasse notée $Ls(a_i)$. Par ailleurs, chaque fourmi garde en mémoire le meilleur site de chasse qu'elle a trouvé le long de ses recherches. Ce site est noté $s_b(a_i)$. Si un site ne rapporte plus de proies, la fourmi effectue un déplacement $d(a_i)$. Elle s'oriente en fonction de l'emplacement du nid $s_{\mathcal{N}}$ ainsi que l'emplacement de son meilleur site de chasse $s_b(a_i)$.

3.3. Exploration globale

D'un point de vue global, la métaheuristique proposée PAO (Pachycondyla apicalis Ant Optimization) place le nid $\mathcal{N}$ en une position $s_{\mathcal{N}}$ de l'espace de recherche S. Elle procède par la suite à l'exploration de S autour de $s_{\mathcal{N}}$. L'exploration est déterminée par le comportement local des fourmis. A chaque pas de l'algorithme, les n fourmis sont simulées en parallèles. A l'initialisation, le nid est placé dans $s_{\mathcal{N}}$ de manière uniformément aléatoire. Puis le nid est déplacé toutes les $P_{\mathcal{N}}$ itérations correspondant à des sorties du nid. Il est alors placé sur le meilleur site de chasse s^* trouvé au sein de la colonie depuis son dernier déplacement, si jamais il y a eu amélioration. A chaque déplacement du nid les fourmis reprennent leur exploration à partir de la nouvelle position du nid.

3.4. Exploration locale

A l'initialisation, et à chaque déplacement du nid, chaque fourmi a_i quitte le nid pour se constituer une liste de p sites de chasse qu'elle mémorise. Un site de chasse est un point de s construit par l'opérateur O_g avec une amplitude $A_{globale}(a_i)$ dans le voisinage de $s_{\mathcal{N}}$. La fourmi a_i va ensuite procéder à une exploration locale autour d'un de ses sites de chasse.

Initialement, la fourmi choisit un site s au hasard parmi les p sites dont elle dispose. L'exploration locale consiste à construire un point s' dans le voisinage de s grâce à l'opérateur O_l avec une amplitude $A_{locale}(a_i)$. La fourmi a_i capture une proie si cette exploration locale a permis de trouver une meilleure valeur de f, ce qui revient à avoir $f(s') < f(s)$, si on se place dans une minimisation de la fonction objectif. Une amélioration de f modélise donc la capture d'une proie. A chaque fois qu'une fourmi parvient à améliorer $f(s)$ elle mémorise s' à la place de s, et sa prochaine exploration locale aura lieu dans le voisinage de s'. Si l'exploration locale est infructueuse, pour la prochaine exploration, la fourmi choisira un site au hasard parmi les p sites qu'elle a en mémoire. Quand un site a été exploré successivement plus de P_{locale} fois sans avoir rapporté de proie, il est définitivement oublié et sera remplacé par un nouveau site à la prochaine itération. Le paramètre P_{locale} représente une patience locale. La création d'un nouveau site de chasse à la suite d'échecs successifs se traduit par un compromis psycho-social. Lorsque une fourmi a_i vient à abandonner un site, on tient en compte lors de la création d'un nouveau site pour chaque fourmi : son déplacement $d(a_i)$, son meilleur site de chasse $s_b(a_i)$, son site courant $s(a_i)$ qu'elle doit abandonner, et de l'emplacement $s_{\mathcal{N}}$ du nid qui correspond au meilleur site de chasse trouvé au sein de la colonie. La création d'un nouveau site $s'(a_i)$ après échec de l'exploration locale à l'itération t, se traduit comme suit :

$$d(a_i)_{t+1} \leftarrow d(a_i)_t + c_1\big(s_b(a_i) - s(a_i)\big) + c_2\big(s_N - s(a_i)\big) \quad \text{(I)}$$

$$s'(a_i) \leftarrow s(a_i) + d(a_i)_{t+1} \quad \text{(II)}$$

Où les paramètres c_1, c_2 sont des variables aléatoires tirées de manière uniforme dans l'intervalle [0,1] et qui ont pour objectif de pondérer les rôles relatifs à l'expérience individuelle (c_1) et la communication sociale (c_2).
Le déplacement d'une fourmi correspond à un mouvement permettant de passer d'une solution à une autre.

L'algorithme de la métaheuristique proposée est donné dans la figure1. La condition d'arrêt peut être l'une des suivantes : l'emplacement du nid $s_{\mathcal{N}}$ n'a pas été amélioré depuis un certain nombre d'itérations, un certains nombre d'évaluation de la fonction objectif a été atteint, ou un certain nombre d'itérations a été franchi.

(1) Choisir aléatoirement l'emplacement du nid.
(2) Les fourmis sont simulées en parallèle, pour chaque fourmis a_i, $i \in [1..n]$:
(3) Créer une liste de p sites de chasse à partir de l'emplacement du nid à l'aide de l'opérateur O_g.
(4) Assigner au meilleur site individuel le meilleur site de la liste de sites chasse seulement s'il y a initialisation.
(5) Se positionner sur un site choisi aléatoirement dans la liste des p sites de chasse.
(6) Faire une exploration locale sur le site courant à l'aide de l'opérateur O_l.
(7) **Si** l'exploration précédente a rapporté des proies **alors**
- Retourner à ce site à la prochaine sortie du nid.

Sinon
- Sélectionner un autre site aléatoirement parmi les p sites de chasse en mémoire pour la prochaine sortie.

Finsi
(8) **Si** un site a été exploré plus de P_{locale} fois sans rapporté de proie **alors**
- Créer un nouveau site de chasse en se déplaçant et en se positionnant selon les équations (I) et (II).
- Abandonner et effacer de la liste le site ne constituant plus de renforcement.

Finsi

(9) **Si** parmi les p sites en mémoire, il existe un site rapportant plus de proies que le meilleur site individuel de la fourmi **alors**
- Mettre à jour le meilleur site par le site nouvellement trouvé.

Finsi
(10) **Si** le critère d'arrêt est vérifié **alors**
- Arrêt de l'algorithme.

Sinon
Si $P_{\mathcal{N}}$ sorties du nid ont eu lieu **alors**
- Déménager le nid vers le meilleur site trouvé par les fourmis durant leurs recherches.
- Effacer pour chaque fourmi la liste des sites de chasse.
- Aller à (3).

Sinon
- Aller à (6).

Finsi
Finsi

Figure 1. Algorithme de PAO (*Pachycondyla apicalis* Ant Optimization)

Du point de vue de la programmation à mémoire adaptative, la mémoire est structurée en plusieurs niveaux : mémoire globale qui correspond à l'emplacement du nid, mémoire individuelle qui comprend la liste des sites de chasses, mais également le meilleur site de chasse qu'a pu trouver chaque fourmi lors de ses recherches.

4. Expérimentation

La métaheuristique PAO (Pachycondyla apicalis Ant Optimization) proposée a été implémentée sur la plate-forme dédiée à la conception et à l'implémentation de métaheuristique ParadisEO (PARallel and DIStributed Evolving Objects) [7]. Nous proposons ParadisEO-PAO comme nouveau composant du framework ParadisEO.

Afin de tester PAO nous nous sommes basés sur un ensemble de jeux de tests très utilisés dans le domaine. Les problèmes pris comme tests sont tirés de la librairie TSPLIB qui regroupe les benchmarks du voyageur de commerce [1].

Un site correspond à un chemin hamiltonien dans le graphe des villes à visiter. Pour l'exploitation d'un site, on a utilisé des heuristiques proches du problème permettant de modifier un chemin pour en trouver un plus court. La modification d'un chemin est effectuée par l'opérateur d'exploration locale O_l. Elle est réalisée par l'heuristique 2-opt. Cette heuristique consiste à supprimer deux arêtes non adjacentes et à reconnecter les segments restants par de nouvelles arêtes. L'amplitude A_{locale} dans ce cas est égale à 2. La génération d'un site à partir de l'emplacement du nid est réalisée à l'aide de l'opérateur O_g correspondant à une permutation simple : deux villes sont échangées au hasard dans la séquence. L'amplitude $A_{globale}$ correspond dans ce cas au nombre de permutations effectuées.

Nous avons comparé PAO à d'autres métaheuristiques basées sur les fourmis artificielles afin d'en mesurer les performances. Ces métaheuristiques sont l'algorithme API [5], et ACS (Ant Colony System [3]). Nous avons repris les résultats obtenus par API et ACS dans [6]. Dans un objectif d'équité, les paramètres de chaque algorithme ont été ajustés pour que le même nombre d'évaluation de la fonction objectif soit effectué. Les paramètres utilisés pour ACS sont les suivants: nombre de fourmis $n = 20$, visibilité de la piste $\beta = 2.0$, coefficient de recherche $q0 = 0.98$, coefficient d'évaporation $\rho = 0.1$, et la quantité initiale de phéromone τ_0 est égale à 0.2. Le critère d'arrêt est fixé à 200 évaluations de la fonction objectif. Pour notre métaheuristique PAO nous avons repris le même critère d'arrêt. Les paramètres utilisés sont : $P_{\mathcal{N}} = 12$, $P_{locale} = 2$, $p = 3$, $n = 20$, c_1=0.4, c_2=0.6, $A_{\text{globale}} = n/10$, $A_{\text{locale}} = 2$. Ces paramètres sont obtenus après l'étude de leur sensibilité sur la qualité des résultats.

Le tableau 1 donne une comparaison sur la qualité des solutions trouvées par PAO, API, ACS. s_m représente la longueur moyenne du plus court chemin trouvé sur 50 essais, et σ l'écart type correspondant. La meilleure solution trouvée s^* représente la longueur du plus court chemin trouvé. La distance entre les villes s'exprime en distance Euclidienne.

Problème	PAO(s_m,[σ], s^*)	API (s_m,[σ], s^*)	ACS (s_m,[σ], s^*)	Optimum connu
EIL51	437.72, [9.07], **415**	439.50, [9.07], **419**	430.76, [6.75], **419**	**415**
EIL76	571.96, [13.69], **540**	591.48, [14.54], **566**	554.88, [4.85], **542**	**538**
KROA100	22570, [214], **22380**	59581.20, [1332.12], **56879**	50995.44, [10098.60], **26501**	**21282**
D198	21775.02, [252.6], **16229**	21775.02, [1594.90], **18769**	16977.78, [144.18], **16606**	**15780**
LIN318	46839.2, [447.833], **45634**	75989.66, [3598.25], **67902**	47043.08, [369.42], **46455**	**42029**
RAT783	9787.4, [102.65], **9667**	18893.86, [846.62], **17081**	10103.17, [71.80], **9880**	**8806**

Tab.1– Comparaison des meilleures solutions trouvées par PAO avec ceux des métaheuristiques API et ACS.

Les résultats obtenus par notre métaheuristique sont meilleurs en qualité que ceux obtenus par API et ACS sur les problèmes traités ci-dessus (tableau 1). On note en moyenne une amélioration en qualité des solutions à hauteur de 53.52% pour PAO par rapport à API. Tandis que le gain moyen enregistré entre PAO et ACS s'élève à 4.57%.

Bien que ACS utilise l'information de distance entre les villes comme heuristique locale, PAO a donné de bien meilleurs résultats, offrant un compromis entre exploration de l'espace de recherche et exploitation des solutions intéressantes.

5. Conclusion

Les résultats obtenus sont très satisfaisants, et démontrent qu'en joignant la théorie de l'auto-organisation au concept de programmation à mémoire adaptative, on peut arriver à concevoir des métaheuristiques offrant une bonne balance entre intensification et diversification dans la recherche. La métaheuristique PAO paraît prometteuse. Il reste à la tester sur d'autres problèmes d'optimisation combinatoire et continue. Elle se présente comme un sérieux candidat pour les meilleures métaheuristiques déjà existantes.

6. Références

[1] Benchmarks du TSP: http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/

[2] A. Drogoul, D. Fresneau, Métaphore du fourragement et modèle d'exploitation collective de l'espace pour des colonies de robots autonomes mobiles, in the proceedings of JFIADSMA'98, (1998), Hermès, Paris, France.

[3] M. Dorigo, L. M. Gambardella, "Ant Colony System: A cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, No.1 (1) :53-66, 1997.

[4] J. Dreo, P. Siarry, "Métaheuristiques pour l'optimisation et auto-organisation dans les systèmes biologiques", *Technique et Science Informatiques*, 2006, Vol. 25, No. 2, pp. 221-240.

[5] N.Monmarché, G.Venturini, M.Slimane, "On how Pachycondyla apicalis ants suggest a new search algorithm", *Future Generation Computer Systems*, vol. 16, 2000, p. 937-946.

[6] N. Monmarché, Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation, Thèse pour l'obtention du grade de Docteur en Informatique, Tours, France, 2000.

[7] Plate-forme ParadisEO: http://paradiseo.gforge.inria.fr/

[8] E.D. Taillard, L.M. Gambardella, M. Gendreau, J.-Y. Potvin, "Adaptive Memory Programming: A Unified View of Meta-Heuristics", *European Journal of Operational Research*, vol. 135, No.1, 1998, p. 1–16.