Métaheuristique des Lucioles Discrète pour les Flow Shop Hybrides

Latifa DEKHICI, Khaled BELKADI

Laboratoire LAMOSI

Département d'informatique
Université U.S.T.O.M.B.
31000 Oran
ALGERIE

dekhicilatifa@gmail.com, belkadi1999@yahoo.com

RÉSUMÉ. L'algorithme des lucioles est une métaheuristique récente bio-inspirée. Dans ce travail, nous visons à fournir une version discrète de l'algorithme des lucioles pour l'ordonnancement d'un atelier de type flow shop flexibles (FSH). Nous comparons l'algorithme des lucioles discret proposé avec la recherche locale et l'optimisation par essaims de particules (PSO) que nous concevons avec les même principes et paramètres. La mise en œuvre avec différents nombres de lucioles sur un flow shop flexible typique qui est le flow shop flexible à deux étages, montre que la métaheuristique proposée trouve les meilleurs makespan mais il prend un temps d'exécution inconsidérément plus élevé que l'algorithme PSO.

ABSTRACT. . The Fireflies algorithm is a recent bio-inspired metaheuristic. In this work, we aim to provide a discrete version of the algorithm fireflies for scheduling a hybrid flow shop (FSH).We compare the discrete fireflies algorithm proposed with local search and particle swarm optimization (PSO) that we design with the same principles and parameters. Implementation with different numbers of fireflies on a typical flexible flow shop which is the two stage two hybrid flow shop, shows that the proposed metaheuristic find the best makespan but it takes an execution time unduly higher than the PSO algorithm.

MOTS-CLÉS : Optimisation discrète, Algorithme des lucioles, Flow shop Hybride, Optimisation par essaim de particules, Cmax.

KEYWORDS: Discrete optimization, firefly algorithm, Hybrid Flow Shop, particles swarm optimization, Cmax

1. Introduction

Actuellement, un ensemble de métaheuristiques bio-inspirées basées sur le comportement naturel des essaims, des abeilles, des oiseaux, et des fourmis avaient émergé comme une alternative pour surmonter les difficultés présentées par les méthodes classiques dans l'optimisation.

Dans le présent document, nous adaptons l'algorithme des lucioles au problème d'ordonnancement de flow shop flexibles. Pour cela, nous décrivons dans la deuxième section la métaheuristique, son origine et ses paramètres. Dans la troisième section, nous présentons le problème de flow shop flexible ou hybride (FSH) et sa notation. Dans la quatrième section, nous proposons une discrétisation de l'algorithme des lucioles au problème de flow shop hybride. Dans la dernière section, nous discutons la comparaison entre l'algorithme des lucioles discret et les autres algorithmes que nous avons implémentés. Enfin, nous donnons une conclusion et des perspectives.

2. Algorithme Des Lucioles (FireFly Algorithm)

2.1. Inspiration

Les lucioles(en anglais firefly) sont de petits coléoptères ailés capables de produire une lumière clignotante froide pour une attraction mutuelle. Les femelles peuvent imiter les signaux lumineux d'autres espèces afin d'attirer des mâles qu'elles capturent et dévorent. Les lucioles ont un mécanisme de type condensateur, qui se décharge lentement jusqu'à ce que certain seuil est atteint, ils libèrent l'énergie sous forme de lumière. Le phénomène se répète de façon cyclique. L'Algorithme de lucioles développé par [17] est inspiré par l'atténuation de la lumière sur la distance et l'attraction mutuelle mais il considère toutes les lucioles comme unisexes.

2.2. Algorithme

L'algorithme des Lucioles (Firefly algorithm) est la plus récente métaheuristique. Elle a été développée par Yang [17][18][19][20]. On peut trouver quelques articles concernant l'algorithme continu des lucioles [1][8][14]. Une validation de l'algorithme continu des lucioles sur l'optimisation des fonctions stochastiques est donnée dans [19]. Sayadi et al. [15] ont proposé la première version discrète pour problème de permutation de flow shop en utilisant un codage binaire de la solution et une formule de probabilité pour la discrétisation. A notre connaissance, peu d'articles ont abordé la discrétisation de cet algorithme pour des problèmes économiques [2][9].

Le pseudo-code de l'algorithme de lucioles ou Firefly peut se montrer comme suit:

Algorithme 1. Métaheuristique FF

```
Procedure FF Meta-heuristic(Nbr_iter: nombre d'itérations)
Début
   Générer une population initiale de lucioles xi (i=1..., n)
   Déterminer les intensités de lumière Ii at xi via f(xi)
Tant que (t<Nbr_iter) faire
Pour i = 1à n       //toutes les lucioles
Pour  j=1 à n       //toutes les lucioles
si (Ij> Ii) alors
   Attractivité βi,j varie selon la distance ri,j
   Déplacer  luciole I vers  j avec l'attractivité βi,j
Sinon  déplacer  I aléatoirement
fin si
   Evaluer la nouvelle  solution
   Mettre à jour l'intensité Ii
   Vérifier si luciole i est la meilleure.
Fin j ,Fin i
   Trouver la meilleure luciole en fonction d'objective
   t++
Fait
Fin procédure
```

2.3. Paramètres

Dans l'algorithme des lucioles, il y a 4 points importants:

2.3.1. Intensité de lumière

Dans le cas le plus simple pour les problèmes de minimisation, la luminosité, ou l'intensité lumineuse d'une luciole à un endroit particulier x peut être choisie comme : I (x) α 1/ f (x).

2.3.2. Attractivité

Dans l'algorithme des lucioles, la principale forme de la fonction d'attractivité peut être n'importe quelle fonction monotone décroissante telle que la forme générale suivante:

$$\beta_{i,j} = \beta_0^* e^{-\gamma r_{i,j}^m} \qquad (1)$$

Où r est la distance entre deux lucioles, β_0 est l'attractivité à r = 0 et γ est un coefficient constant d'absorption de lumière.

2.3.3. Distance

La distance entre 2 lucioles i et j au x_i et x_j peut être la distance cartésienne comme suit:

$$r_{i,j} = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \qquad (2)$$

où $x_{i,k}$ est la $k^{ème}$ composante de la $i^{ème}$ luciole.

2.3.4. Mouvement

Le déplacement d'une luciole i attirée par une plus lumineuse (attrayante) luciole j , est déterminé par

$$x_i = (1 - \beta_{i,j})x_i + \beta_{i,j}x_j + \alpha(rand - {}^{1}/_{2}) \qquad (3)$$

où le premier terme et la second est due à l'attraction. Le troisième terme est la randomisation. α est le paramètre aléatoire et peut être constant. "rand" est un générateur de nombre aléatoire uniformément distribuée dans [0, 1].

3. Flow Shop Hybride

3.1. Présentation

Un flow shop hybride(FSH) est un système composé d'un ensemble d'étages, où chaque étage est composé d'une ou plusieurs machines en parallèle. Les différentes tâches visitent les étages dans le même ordre. Sur chaque étage, une tâche est traitée par une machine. Entre chaque deux étages, les tâches peuvent attendre ou non dans des stocks illimités ou limités [16][5]. Par ailleurs, toutes les tâches sont supposées être disponibles à l'entrée du système avec les dates 0 [4].

L'ordonnancement consiste à trouver une séquence adéquate des tâches et leurs affectations aux machines. Pour l'optimisation d'un critère de performance. On peut citer le makespan ou Cmax(le délai d'achèvement du dernier travail sur le dernier étage), le temps d'écoulement max (Fmax) [10] et des critères liés aux retards[2][12] . Ce problème peut avoir plusieurs contraintes [6]. Une discussion de la littérature est donnée dans [13].

3.2. Notation

Le nombre d'étages est M, le nombre de tâches est N et l_k est le nombre de machines en étage k. La figure 1 est un exemple d'un flow shop hybride avec 2 étages,3

machines sur le premier étage et 2 sur le second. Un stock de capacité infinie est incorporé entre les étages. Si l'on considère f_{ik} la date fin de la tâche i à l'étage k alors $Cmax = max\ f_{iM}$. Selon la notation de [31], le système est un FSH2 (3, 2) | | Cmax.

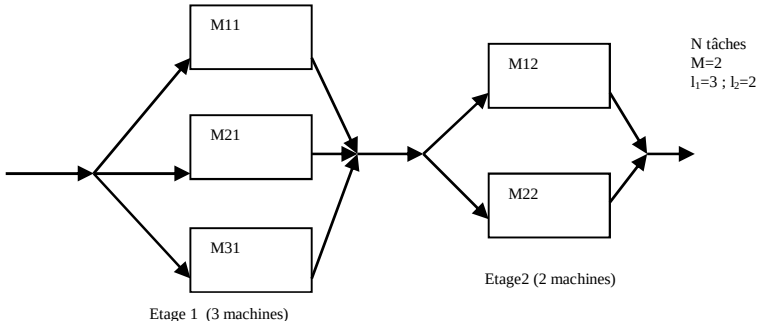

Figure 1 *Schéma de FSH2 (L_1 = 3, L_2 = 2) | | Cmax*

4. Discrétisation de l'algorithme des lucioles pour FSH

Notre fonction objective est le makespan (Cmax). La codification d'une solution d'ordonnancement en FSH est une concaténation de l'ordre des n travaux et leurs affectations aux machines l_k de chaque étage k. Pour cela, nous voyons que la distance Hamming entre les lucioles est la mieux adaptée pour une optimisation discrète. Elle se mesure par la dissemblance de chiffres entre deux solutions.

Comme la luciole passe lors du mouvement d'une valeur entière à une valeur réelle, nous utilisons une restriction de sa nouvelle position au domaine de la solution : [1..N] pour l'ordre et [1..l_k] pour les affectations.

5. Résultats et Discussion

Pour fournir une étude comparative de l'algorithme des lucioles, nous concevons 2 métaheuristiques: une recherche locale simple (SLS) avec un paramètre d'itérations et une optimisation par essaim de particules (Particle Swarm Optimization, PSO) [11]. Ce dernier algorithme a été utilisé dans [7] pour les FSH. Nous y utilisons la même codification et les mêmes principes de discrétisation que les lucioles.

L'étude expérimentale a été réalisée sur des flow shop hybride à deux étages avec machines indépendantes, ce qui signifie que les temps de traitement d'une tâche dans un étage dépendront de chaque machine. Dans les exemples ci-dessous, nous avons FSH(7,8) et FSH (4,3) pour 10 et 20 tâches. t_{ijk} qui est le temps de traitement de chaque

tâche i par la machine k à l'étage j est une valeur aléatoire entre 5 et 100 générée en début de chaque réplication. Nous générons 50 réplications pour chaque cas.

La figure 2 montre la comparaison des Cmax moyens pour des simulations à 100 itérations avec un nombre de lucioles égales au nombre de particles= 10. Nous remarquons que l'algorithme de lucioles (FF) montre des résultats comparables pour les deux cas: FSH(3,4) et FSH(7,8) et pour toutes les instances. Par exemple, le Cmax moyen atteint par l'algorithme des lucioles pour le FSH (4,3), N = 20 est de 194,92 alors qu'il est de 463 pour la recherche locale et de 272,62 par la PSO. (Fig. 2).

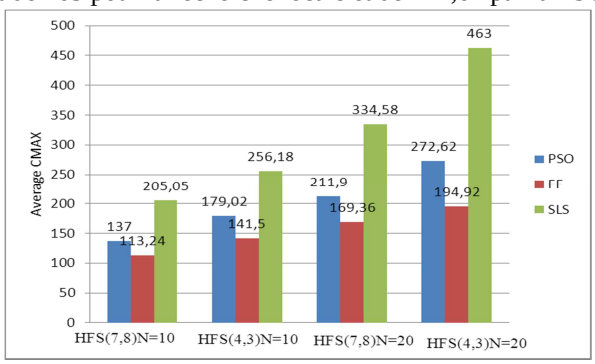

Figure 2 *Comparaison des Cmax moyens, nbr. de particules =10, nbr. d'itérat. =100.*

Pour tester les performances de l'algorithme de FF au sujet de temps CPU, nous schématisons sur la figure 3, le temps d'exécution moyen d'une réplication. Le temps d'exécution de l'algorithme des lucioles FF augmente avec le taux d'amélioration. L'algorithme des lucioles(FF) prend plus de temps que la PSO. Il peut durer de 60 millisecondes à 257 millisecondes par rapport à 4 millisecondes à 8 millisecondes pour la PSO. Mais il reste est inférieur à 1 secondes et donne une amélioration de plus de 56%.

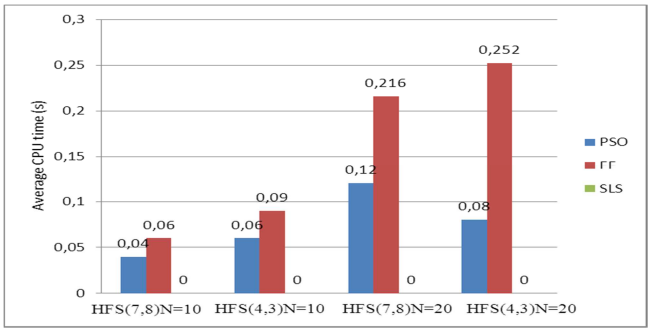

Figure 3 *Comparaison des temps CPU moyenne, nbr particules =10, nbr. d'itérat. =100.*

En raison de la comparaison des lucioles entre elles, qui est un principe de l'algorithme de base, l'exécution peut prendre plus de temps d'exécution que PSO. PSO semble être mieux en termes de temps de calcul. Ce pourrait être l'effet de la comparaison d'une particule seulement avec sa meilleure position historique. En termes d'amélioration, l'algorithme des lucioles est potentiellement plus puissant dans la résolution des problèmes FSH. L'algorithme de lucioles semble être un outil d'optimisation d'ordonnancement prometteur.

6. Conclusion

Le but de cet article était de résoudre l'ordonnancement de flow shop hybride en utilisant une version discrète de l'algorithme de luciole. Cet algorithme est métaheuristique bio-inspirée. Nous avons essayé dans cet article de donner une discrétisation de l'algorithme. Nous avons gardé le codage expressif de la solution de flow shop hybride. Nous avons opté pour la distance de Hamming entre les lucioles qui est plus signifiante pour les valeurs discrètes. La discrétisation et le bon choix de paramètre d'attraction a permis à notre version d'algorithme d'atteindre le meilleur taux d'amélioration sur la minimisation des makespan dans un temps d'exécution raisonnable. Pour affiner le travail décrit dans ce document, nous travaillons sur d'autres cas et variantes de système de production. Ce travail nous permettra de connaître les nombres de lucioles et d'itérations adéquats pour chaque instance. Nous avons prévu également la mise en œuvre de l'algorithme des lucioles parallèle afin de diminuer le temps d'exécution et d'améliorer l'efficacité de cette métaheuristique pertinente.

7. Bibliographie

[1] Aungkulanon, P., Chai-Ead N., and Luangpaiboon P.:"Simulated Manufacturing Process Improvement via Particle Swarm optimization and Firefly Algorithms". In Proc. International Multi Conference of Engineers and Computer Scientists IMECS'11,Hong Kong,.1123-1128. (2011)

[2] Bank, J. and Werner, F.: "Heuristic Algorithms for Unrelated Parallel Machine Scheduling with a Common Due Date, Release Dates, and Liner Earliness and Tardiness Penalties", Mathematical and Computer Modeling, Vol.33,pp.363-38.(2001).

[3] Basu B. , Mahanti G. K.: "Firefly and artificial bees colony algorithm for synthesis of scanned and broadside linear array antenna", Progress In Electromagnetics Research 32:169-190. (2011).

[4] Belkadi, K., Gourgand M. , Benyettou M.: "Resolution of scheduling problem of the production systems by sequential and parallel Tabu search". J. Applied Sci.,6:1534-1539.(2006).

[5] Billaut, J.C., Vignier A., Proust C. and Portmann, M.C.: "A Survey of Hybrid flow Shop Problems". ODIP, Aussois, France.(2000).

[6] Dekhici, L. and Belkadi, K.: “Operating theatre scheduling under constraints”. J. Applied Sciences, Vol. 10, pp.1380-1388. (2010)

[7] Ercan M.F, “Particle Swarm Optimization and Other Metaheuristic Methods in Hybrid Flow Shop Scheduling Problem”, Particle Swarm Optimization, Aleksandar Lazinica, 2009.

[8] Horng M-H. and T-W. Jiang: “The Codebook Design of Image Vector Quantization Based on the Firefly Algorithm”, Lecture Notes in Computer Science, Vol. 6423/2010,pp.438-447.(2010).

[9] Jati G. K. and Suyanto S.: “Evolutionary discrete firefly algorithm for travelling salesman problem”, ICAIS2011, Lecture Notes in Artificial Intelligence (LNAI 6943), pp. 393-403.(2011)

[10] Jin, Z., Yang, Z. and Ito, T.: “Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem”. International Journal of Production Economics, 100: 322–333.(2006).

[11] Kennedy, J., Eberhart, R.C.:” Particle swarm optimization”. In: Proc. of IEEE International Conference on Neural Networks, Piscataway, NJ ,pp.1942–1948. (1995).

[12] Kim, D.W., Kim, K.H., Jang, W., and Chen, F.F.: “Unrelated parallel machine scheduling with setup times using simulated annealing”, Robotics and Computer Integrated Manufacturing, 18:223-231.(2002).

[13] Linn, R., Zhang, W.: “Hybrid flow-shop schedule: a survey”, Computers and Industrial Engineering, 37:57-61 (1999).

[14] Lukasik, S. and Zak, S: ”Firefly Algorithm for Continuous Constrained Optimization Tasks”, Lecture Notes in Computer Science, 5796/2009: 97-106, (2009)

[15] Sayadi M. K., R. Ramezanian, N. Ghaffari-Nasab: “A Discrete Firefly Meta-heuristic with Local Search for Make span Minimisation in Permutation Flow Shop Scheduling Problems”, International Journal of Industrial Engineering Computations, Vol. 1,pp.1–10.(2010)

[16] Vignier A., Billaut, J.C. and Proust, C.: “Scheduling problems type hybrid flow-shop: State of the art.” RAIRO. Recherche opérationnelle, 33(2),pp 117-183. (1999)

[17] Yang X.S. :”Nature-Inspired Metaheuristic Algorithms”. Luniver Press, UK.(2008)

[18] Yang X.S.: “Firefly Algorithms for Multimodal Optimization, Stochastic Algorithms: Foundations and Applications”, SAGA 2009, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 5792:169-178.(2009)

[19] Yang X.S. : “Firefly algorithm, stochastic test functions and design optimization”, International Journal of Bio-Inspired Computation, 2(2):78-84.(2010)

[20] Yang X.S.: “Firefly algorithm, Lévy flights and global optimization”, in: Research and Development in Intelligent Systems XXVI (Eds M. Bramer, R. Ellis, M. Petridis), Springer London, 209-218.(2010).