Traitement d'un problème de type FJSP (Flexible Job Shop scheduling problem) à l'aide d'algorithme génétique

Zahia Khaldouna et Messaoud Djeghaba

Laboratoire d'automatique et de signaux d'Annaba (LASA),
Université Badji Mokhtar, BP12, Annaba, Algérie.

Accepté le 28/07/2010

ملخص

في هذه الورقة، درسنا المشكلة الأمثل في التصنيع الخلايا المرنة، والتي تعتبر عملية معقدة جدا. فمن بين العديد من التقنيات والأساليب المستخدمة لدراسة هذا النوع من المشاكل، هناك البحوث الميدانية والاستدلال وغيرها من الفوقية وهناك الاستدلال القائم على الذكاء الاصطناعي (الشبكة العصبية، خوارزميات التطورية، المنطق الضبابي، من المحرمات، وما إلى ذلك)، ففي حالتنا اخترنا نهج يقوم على الخوارزمية الجينية، ومن ثم مقارنتها مع التي وضعت في [2]، والذي يستخدم نفس الأداة ولكن يكمن الفرق بين الطريقتين في اختيار وظائف التقييم للأفراد والعمليات الوراثية. وللتحقق من صحة النتائج، استخدمنا نفس قاعدة البيانات التي استخدمت في [2]. ومن ثم وبالمقارنة بين الطريقتين تظهر أن النتائج تكون أفضل بكثير باستعمال نهج الموضوع.

الكلمات المفتاحية: *خلية التصنيع المرنة؛ جدولة؛ السلوك؛ الفوقية الاستدلال؛ الخوارزمية الجينية.*

Résumé

Dans cet article, nous avons étudié le problème d'optimisation d'une cellule de production flexible de type FJSP (flexible job-shop scheduling problem), dont le contrôle est très complexe. Parmi les multiples techniques et méthodes utilisées pour l'étude de ce type de problème, il y a celles qui relèvent de la recherche opérationnelle, et d'autres d'heuristiques et méta-heuristiques basées sur l'intelligence artificielle (réseau de neurones, algorithmes évolutionnaires, logique floue, tabou, etc.). Dans notre cas, nous avons opté pour une approche basée sur l'algorithme génétique, pour ensuite pouvoir la comparer avec celle développée dans [2], qui utilise le même outil. La différence réside dans le choix des fonctions d'évaluation des individus, et les opérateurs génétiques. Afin de valider les résultats, nous avons exploité la même base de données que celle utilisée dans [2]. Cette comparaison entre les deux méthodes a fait ressortir des résultats sensiblement meilleurs pour l'approche soumise.

Mots clés : *Cellule de production flexible; Optimisation ; Contrôle ; Méta-heuristique, Algorithme génétique*.

Abstract

In this paper, we studied the optimization problem of a flexible manufacturing cell type FJSP (flexible job shop scheduling problem), whose control is very complex. Among the many techniques and methods used to study such problems, there are those within operational research, and other heuristics and meta-heuristics based on artificial intelligence (neural network algorithms evolutionary fuzzy logic, taboo, etc...)]. in our case, we opted for an approach based on genetic algorithm, and then to compare it with that developed in [2], which uses the same tool. The difference lies in the choice of evaluation functions of individuals and genetic operators. To validate the results, we used the same database as that used in [2]. The comparison between the two methods shows significantly better results for the subject approach.

Key words*: Flexible manufacturing cell; Optimization; Control; Meta-heuristics; Genetic algorithm***.**

Auteur correspondant: zayakhaled@yahoo.fr (Zahia Khaldouna)

1. INTRODUCTION

La conduite d'une cellule de production flexible qui est composée de machines, de robots, de systèmes de transport, de magasin d'outils, offre beaucoup d'avantages en flexibilité, mais pose aussi plusieurs problèmes complexes, notamment l'affectation des tâches aux robots dans un environnement dynamique et aléatoire (distribution des pièces et matière première, défaillances des équipements, modification de la production, etc.), et la minimisation des temps de production globaux (Makespan: temps nécessaire pour compléter tous les travaux).

Le partage de ressource est une question des plus critiques dans la planification du système de production. Il dépend de l'environnement et des contraintes sur le processus. Dans le problème de type FJSP étudié, chaque tâche (gamme opératoire) est formée par une séquence d'opérations, et chaque opération nécessite une machine opérationnelle. La décision qui concerne le séquencement d'opérations sur la machine devra être optimisée afin d'améliorer la performance du système. Le traitement de toutes ces questions par des méthodes exactes s'est avéré inadapté, par rapport à la difficulté de modélisation des processus et surtout par l'exigence de décisions en temps réel. C'est pourquoi, les chercheurs se sont orientés vers des méthodes heuristiques et méta-heuristiques qui ne garantissent pas toujours des résultats optimaux, dans les différentes situations, mais offrent le plus souvent des résultats très acceptables, en tenant compte de la fonction objectif, et fournit un meilleur partage de ressource dans un temps raisonnable.

C'est dans ce cadre que cet article a été rédigé. Nous proposons, une méthode d'optimisation à l'aide d'algorithmes génétiques qui s'affirment peu à peu comme des techniques d'optimisation des plus robustes. Ces dernières peuvent être appliquées à des problèmes très divers car elles sont indépendantes du processus à optimiser et n'utilisent pas les dérivées. Un nouvel algorithme génétique (GA) appliqué au cas FJSP, utilisant de nouvelles stratégies, a été développé par F.Pezzella [2]. Il nous a donc paru utile de pouvoir confronter les deux approches, en traitant les mêmes données, et ainsi les situer l'une par rapport à l'autre à travers les résultats obtenus.

2. MÉTHODES DE RÉSOLUTION

Un très grand nombre de méthodes de résolution existent pour l'optimisation combinatoire et l'affectation sous contraintes. Ces méthodes font partie de deux groupes de nature différente. Le premier groupe comprend les méthodes exactes d'arborescence qui garantissent la complétude de la résolution : c'est le cas de SEP (séparation & évaluation progressive), A* et CSP (constraint satisfaction problems). Le temps de calcul nécessaire de telles méthodes, augmente en général exponentiellement avec la taille du problème à résoudre (dans le pire des cas). Le second groupe comprend les méthodes approchées, qui permet de trouver une solution de bonne qualité en un temps de calcul raisonnable sans garantir l'optimalité de la solution obtenue. Ces méthodes approchées sont fondées principalement sur diverses heuristiques, souvent spécifiques à un type de problème.

Les méta-heuristiques constituent une partie importante des méthodes approchées, et offrent des voies très intéressantes en matière de conception de méthodes heuristiques pour l'optimisation combinatoire. Parmi elles, les algorithmes génétiques.

3. PROBLÉMATIQUE

Soit un ensemble T de n tâches T = {T_1, $T_{2,}$…, T_n}, chaque tâche T_i est définie par une séquence de k opérations, Ti = {O_{i1}, $O_{i2,}$…, O_{ik}}. Soit M, l'ensemble des m machines sur lesquelles s'exécutent les opérations, M = {M_1, $M_{2,}$…, M_m}. Chaque opération i_k n'est exécutée que sur une machine M_j à la fois. Le temps d'exécution de chaque opération est lié à la machine choisie, soit t_{ikj}, le temps de réalisation de l'opération k de la tâche T_i, sur la machine j.

Les opérations sont non préemptives. Les machines ne pouvant exécuter qu'une seule opération à la fois.

$T = \{T_1, T_{2,}\ldots, T_n\}$

$T_i = \{O_{i1}, O_{i2,}\ldots, O_{ik}\}$

$M = \{M_1, M_{2,}\ldots, M_m\}$

t_{ikj}= temps de réalisation de O_k de T_i sur M_j.

C_i = temps de réalisation de T_i.

Il s'agit alors d'affecter les opérations des tâches aux machines, et le séquencement des opérations sur chaque machine, de la façon la plus proche de l'optimale tout en respectant les contraintes structurelles (contraintes de précédences), dans le but de minimiser le temps de réalisation global (Makespan).
Le temps nécessaire pour effectuer toutes les tâches est définie par $M_k=max_i$ [C_i], avec C_i, le temps nécessaire pour effectuer la tâche T_i.

Ce problème est connu, sous l'appellation de FJSP (flexible job shop scheduling problem).

Le cas d'étude proposé par F.Pezzela, représente une cellule de production composée de machines, et de produit à réaliser. Les données sont organisées dans une table (table1), où les lignes correspondent aux opérations, les colonnes aux machines, et les entrées aux temps de traitement.

La population initiale est générée par l'approche par localisation de *Kacem et al* [5]. Cette approche consiste à trouver l'affectation initiale par ré-ordonnancement des tâches et des machines, et par la recherche du minimum global de la table d'instance. Elle prend en compte le temps de traitement, et la charge de travail sur la machine. La somme du temps de traitement des opérations est affectée à chaque machine. La procédure consiste à chercher pour chaque opération, la machine qui la réalise avec le minimum de temps. Une fois ce choix fixé, On additionne ce temps à chaque entrée de la même colonne (mise à jour de la charge de travail de la machine), comme le montre la table 2.

Cette approche dépend de l'ordre dans lequel les opérations et les machines sont données dans la table. Pour séquencer les opérations générées par la population initiale, elles seront modifiées légèrement, de deux manières :
Régle1 : chercher le minimum global dans la table de traitement
Régle2 : permutation aléatoire des opérations et des machines dans la table.
Une fois les deux règles appliquées, il restera à déterminer le séquencement d'opérations sur la machine**.** Trois règles sont adoptées par F.Pezzella [2],
Règle 1: "The most work remaining (MWR)",
Règle 2: "The most number of operations remaining (MOR)"
Régle 3: "Randomly select job (Random)".
Il sera réalisable s'il respecte les contraintes de précédence des opérations d'une gamme (l'opération $O_{i,j+1}$ ne doit pas passer avant l'opération $O_{i,j}$).

Table 1. *Temps de traitement*

	M_1	M_2	M_3	M_4
O_{11}	6	5	3	5
O_{12}	3	7	4	4
O_{13}	8	4	3	6
O_{21}	7	5	2	4
O_{22}	2	4	7	2
O_{31}	2	5	1	3
O_{32}	3	5	7	3
O_{33}	9	7	2	2

Table 2. *Approche par localisation (la mise à jour de la charge de travail de la machine est en gras)*

	M_1	M_2	M_3	M_4		M_1	M_2	M_3	M_4
O_{11}	6	5	[3]	4		6	5	[3]	4
O_{12}	3	7	**8**	5		[3]	7	**8**	5
O_{13}	8	4	**7**	6		**12**	4	**7**	6
O_{21}	7	5	**6**	4		**11**	5	**6**	4
O_{22}	2	4	**11**	2		**6**	4	**11**	2
O_{31}	2	5	**5**	3		**6**	5	**5**	3
O_{32}	3	5	**11**	3		**7**	5	**11**	3
O_{33}	9	7	**6**	2		**13**	7	**6**	2

4. L'ALGORITHME GÉNÉTIQUE (AG) POUR L'OPTIMISATION D'UN FJSP

4.1 Introduction

Les AGs ont été mise au point par Holland [4] puis développés par Goldberg [3]. C'est une technique inspirée de l'évolution d'un processus naturel, utilisée pour une optimisation globale de divers problème d'optimisation, dans le but d'obtenir une solution approchée dans un minimum de temps. Cette évolution s'effectue sur la base de transformation inspirée de la génétique, assurant de génération en génération, l'exploration de l'espace des solutions en direction des plus adaptées. Elle commence par des solutions choisies, appelées : La population, dont l'évolution biologique nous conduit à améliorer ses descendants. La population de solution est constituée d'un ensemble d'individus (solutions) ou chromosomes. Chacun d'eux contient un certain nombre de gènes, constitués sous forme de chaîne. Celle-ci code les fonctionnalités de l'organisme et représente un support de l'information génétique. A Chaque itération, appelée génération, il est crée une nouvelle population.

La création d'une nouvelle population à partir de la précédente se fait par l'application d'opérateurs génétiques qui sont : la sélection, le croisement et la mutation.

Le cas étudié dans cet article, s'appuie sur une base de données tirée de [2]. Cet auteur utilise une population initiale générée par la procédure de recherche du minimum global dans une table définissant les affectations des tâches aux machines mentionnée à l'étape 3.

Dans notre cas, le choix de la population initiale est déterminé de la même manière, mais le choix des règles pour la sélection et la reproduction de nouveaux individus sont, beaucoup plus déterministes afin de s'éloigner de l'aspect aléatoire.

Ces différentes opérations sont explicitées dans ce qui suit :

4.1.1 Codage

Il utilise le partage symbolique par chaîne. Celle-ci est formée par le triplet (i, j, k) pour chaque opération.
i : gamme opératoire
j : numéro de l'opération sur la gamme i
k : la machine assignée à l'opération j
Ou ($O_{i,j}$, M_k) est donnée par la chaîne (i, j, k)

Les gènes du chromosome décrivent les opérations sur les machines, et l'ordre d'apparition dans le chromosome décrit les séquences d'opérations. Chaque

chromosome représente une solution du problème.

Dans notre étude on a utilisé le codage réel [6], [1] représenté par une chaîne de paramètres $O_{i,j,k}$, ou chaque paramètre est une valeur réelle qui représente le temps opératoire.

4.1.2 Fonction d'évaluation

La fonction d'évaluation du chromosome coïncide avec le MAKESPAN de la solution représentée par le chromosome.

Il s'agit d'évaluer la solution au cours de la recherche, et de décider des solutions qui seront retenues de celles qui seront rejetées.

L'évaluation générique préfère le chromosome avec une valeur de MAKESPAN minimale.

La différence d'application des opérateurs génétiques des deux stratégies est illustrée ci-dessous :

4.1.3 Sélection

Pour la 1ére stratégie :

La phase de sélection à pour but de choisir les chromosomes pour la reproduction, selon les trois méthodes de sélection suivante :

Binary tournament: deux individus sont sélectionnés aléatoirement parmi la population, le meilleur d'entre eux est sélectionné pour la reproduction

N-size tournament: les individus de la reproduction sont choisis parmi un nombre aléatoire d'individus

Linear ranking: les individus sont triés selon leur fonction d'évaluation, un rang $r_i \in \{1..N\}$ est assigné à chacun, avec N: taille de la population, le meilleur individu prend le rang N, tandis que le mauvais prend le rang 1. $p_i = 2.r_i/N.(N+1)$. C'est la probabilité de choisir l'ième individu.

Pour la 2éme stratégie (Approche proposée):

Nous avons choisi la sélection par décimation [7]. Les meilleurs individus de la population (les plus adaptés) sont sélectionnés et autorisés à se reproduire. Les individus dont le MAKESPAN est élevé seront éliminés.

4.1.4 Génération de survie

Pour la 1ére stratégie :

La nouvelle génération est obtenue par le changement d'affectation des opérations aux machines (croisement, mutation, mutation intelligente) et par le changement de séquencement des opérations (Precedence preseving order-based crossover POX and Precedence preseving shift mutation PPS). Ces règles préservent la flexibilité des nouveaux individus. Ces derniers sont générés jusqu'à ce qu'un nombre maximum d'individus soit fixé, enrichi, et forme la génération suivante à chaque étape de l'algorithme.

Pour la 2éme stratégie :

La nouvelle génération est obtenue par le croisement arithmétique, où deux individus parents produisent qu'un seul enfant à la fois (similaire à l'opération de multiplication),

4.1.5 Critère d'arrêt

L'algorithme s'arrête lorsque le nombre maximum de génération est satisfait et le meilleur individu avec le partage correspondant (c'est-à-dire la ressource partagée qu'il exploite, et qui est indiquée par l'information génétique portée par les gènes de cet individu) est donné et fourni, sinon l'algorithme répète les séquences de 3.

4.2 Description détaillée

4.2.1 Population initiale

Cette approche prend en compte les délais de traitement, le travail chargé par la machine, et la somme des temps de traitement des opérations attribuées à

chaque machine. La procédure consiste à trouver, pour chaque opération, la machine avec le minimum de temps de traitement, la fixation de cette affectation, puis d'ajouter ce temps à chaque séquence d'entrée de la même colonne.
Comme cette approche est strictement dépendante de l'ordre dans lequel les opérations et les machines sont données dans la table, on opère selon la règle suivante :

On cherche le minimum global dans la table des temps. On débute par l'opération qui correspond au minimum global de la table. Par conséquent, le temps du travail chargé par la machine est ajouté pour toute autre opération, pour toute autre entrée de la même colonne. Notre population initiale est formée de cinquante individus.

4.2.2 Codage

Dans l'ordre d'implantation de l'AG, on a besoin de représenter le partage symbolique formé par le triplet (i, k, j) pour chaque opération, où :
i : est le numéro de la tâche de cette opération
k : nombre progressif de cette opération dans la tâche i
j : machine affectée à cette opération.

Le codage réel est représenté par une chaîne de paramètres $O_{i,j,k}$, où chaque paramètre est une valeur réelle qui représente le temps opératoire.

4.2.3 Fonction d'évaluation (fitness)

Nous définissons respectivement t_{ikj}, att_j et F_{ij}, comme, la durée opératoire de l'opération k de la tâche i sur la machine j, la durée d'attente de disponibilité de la machine j pour effectuer l'opération k et la date de fin de chaque opération O_{ij} où F_{ij} est donné par l'expression : $F_{ij} = att_j + t_{ikj}$;

Le makespan est le maximum des dates de fin des dernières opérations pour chaque tâche:

C_{max} = Max $\{C_i\}$ avec C_i est le temps de réalisation de T_i défini par $C_i = \sum_j F_{ij}$

4.2.4 Sélection

Notre algorithme débute par une population initiale de N individus (N = 2 × V). La méthode utilisée est la sélection déterministe. Elle consiste à associer à chaque chromosome i de la population, une fonction C_i qui représente la fonction objectif pour l'individu i (proportionnelle à sa performance d'adaptation). Ainsi, les individus seront triés par ordre décroissant selon leur performance d'adaptation. En s'appuyant sur le principe de la sélection déterministe, les individus les mieux adaptés correspondant à la moitié supérieure, sont sélectionnés.

(N/2 = V individus si N (population) est un nombre paire, et N / 2 + 1 = V individus si N est impaire). Les reste des individus seront élimines. Cela permet d'accoupler les meilleurs individus et améliore globalement l'adaptation.

Phase de reproduction :

Le séquencement des opérations est préservé lors de la reproduction. Il doit respecter les contraintes de précédence entre les opérations de même gamme.

4.2.5 Croisement

Un croisement heuristique a été utilisé. Cette opération est particulière pour les trois raisons suivantes :
- elle utilise les valeurs de la fonction objectif,
- elle produit un seul enfant,
- elle peut aussi ne pas produire d'enfant,

Cet opérateur génère un enfant x_3 à partir du produit arithmétique de ces deux parents x_1 et x_2, ($x_3 = x_1 . x_2$), sous le respect des contraintes de précédence des opérations d'une même tâche. L'enfant généré x_3, acquiert une performance meilleure que ses parents x_1 et x_2.
Le croisement se fait entre les taches en question sur la même ressource et les

ressources à partager, pour produire les opérations futures sur ses ressources

4.2.6 Mutation

L'opérateur de mutation évite d'établir des populations uniformes incapables d'évoluer. Elle a un rôle secondaire par rapport à la recombinaison (croisement), qui est à la base des algorithmes génétiques. L'algorithme proposé, utilise la mutation réelle, il permute l'emplacement des opérations de deux tâches différentes Ainsi si un chromosome est choisi pour la mutation, avec une probabilité p_m, nous déterminons pour chaque machine j les chromosomes correspondant aux temps Max $\{C_i\}$ et Min $\{C_i\}$. Et dans une seconde phase, nous échangeons les emplacements des opérations de ces deux chromosomes sur la machine en respectant la gamme opératoire.

4.2.7 Sélection des survivants

Cette étape consiste à ne garder que les solutions les plus intéressantes. Il y aura donc autant de ''morts'' que de nouveau-nés. On choisit de garder seulement les enfants. Cela assure la diversité et l'évolution de la population. Nous avons utilisé, la méthode d'insertion par triage, qui consiste à ressortir le meilleur chromosome de la nouvelle population. A cet individu est alloué la ressource.

4.2.8 Critère d'arrêt

Afin d'éviter que les meilleurs chromosomes ne soient perdus après les opérations de croisement et de mutation, on se limite au nombre de générations g (g = 1), composée seulement d'enfants. Cela permet d'éviter d'exécuter des simulations inutilement puisque l'optimum global est déjà trouvé.

5. RESULTATS

Dans la table 3, la 1ère colonne représente le nombre de cycles à atteindre NC, la deuxième colonne ($M_{akespan}AG$) représente la Makespan obtenu par la mise en œuvre de l'algorithme de [F. Pezzella]. On le compare avec la troisième colonne ($M_{akespan}AC$) qui représente la Makespan obtenus par notre approche, exploitant la même base de données de l'algorithme [F. Pezzella]. La quatrième colonne représente l'écart obtenu entre les deux méthodes ($E_{cart}M$) défini par:

$$E_{cart}=(M_{akespan}AC-M_{akespan}AG)/M_{akespan}AG*100\%.$$

Dans la table 4 la deuxième colonne (N_pAG) et la troisième colonne (N_pAC) représentent respectivement le nombre de produits fabriqués par l'algorithme de [2], et le nombre de produits fabriqués par notre algorithme. La quatrième colonne indique la comparaison des deux méthodes, et définit l'écart du nombre de produits ($E_{cart}N$), exprimée par :

$$E_{cart}N = (N_pAC - N_pAG) / N_pAG *100\%.$$

Table 3

NC	$M_{akespan}AG$	$M_{akespan}AC$	$E_{cart}M\%$
1	*6*	*5*	*-20*
25	*236*	*224*	*-5.35*
50	*455*	*434*	*-4.83*
75	*661*	*648*	*-2.00*
100	*871*	*861*	*-1.16*
125	*1078*	*1068*	*-0.93*
150	*1295*	*1286*	*-0.69*
175	*1497*	*1496*	*-0.06*
200	*1724*	*1704*	*-1.17*

Table 4

NC	N_pAG	N_pAC	$E_{cart}N$ %
1	*2*	*2*	*+1.03*
25	*194*	*196*	*+0.51*
50	*387*	*389*	*+2.84*
75	*566*	*581*	*+2.65*
100	*746*	*772*	*+3.48*
125	*925*	*963*	*+4.10*
150	*1109*	*1155*	*+4.14*
175	*1300*	*1346*	*+3.53*
200	*1494*	*1535*	*+2.74*

Il s'ensuit des résultats présentés dans les deux tableaux, que pour les cas étudiés et comparés à l'approche [2], la notre indique une amélioration significative des résultats, à la fois sur l'ensemble du temps total de réalisation (Makespan) et sur le nombre de produits réalisé.

6. CONCLUSION

Il est clair que les techniques s'appuyant sur les algorithmes génétiques peuvent être un outil très intéressant dans la prise en charge des problématiques liées aux cellules de production flexible. Nous avons essayé, dans ce travail de comparer notre approche avec celle proposée par d'autres auteurs, pour traiter un problème de type FJSP (flexible job shop scheduling problem), avec les mêmes données.

Le résultat obtenu montre une amélioration sensible des résultats obtenus par l'autre approche. Ce résultat pourrait être encore amélioré, par l'exploration d'autres voies sur le choix des fonctions d'évaluation des individus par exemple, ou par l'utilisation d'une combinaison de différents critères de sélection pour choisir les meilleurs individus d'une génération. Il serait aussi judicieux d'exploiter l'hybridation dans les méthodes de résolution qui peut ouvrir une voie intéressante à la résolution de ce problème.

Références

[1] J.M. Alliot et T. Schiex, *Intelligent Artificielle & Informatique théorique*, Cepadués- Editions, Toulouse, 1994, p. 441-460.

[2] F. Pezzella, G. Morganti, G. Ciaschetti, *A genetic algorithm for the flexible job-shop scheduling problem*, Science Direct, Computers & Operations Research, Vol. 35, 2008, p. 3202-3212.

[3] D.E. Goldberg, *Genetic Algorithm in Search, Optimisation and Machine Learning*, Addison Wesley, 1989.

[4] J.H. Holland, *Adaptation in natural and artificial systems*, University of Machington Press, Ann Arbor, 1975.

[5] K.l. Hammadi et S.P. Borne, *Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems*, IEEE Transactions on systems, Man, and Cybernetics, Part C, Vol. 32, Issue 1, 2002, p. 1-13.

[6] Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag, 1994.

[7] Y. Rahmat-Sami et E. Michielssen, *Electromagnetic Optimisation by Genetic Algorithm*, John Wiley & Sons, 1999.