Adaptation d'un algorithme mémétique pour l'amélioration des performances d'un système flexibles de production

Y. Houbad[1], A. Hassam[2], M. Souier[3], Z. Sari[4]

[1,2,3,4]Univ. Abou-Bakr Belkaid B.P. 119 Tlemcen, Algérie.

[1]houbadyamina@yahoo.fr

{[2]a_hassam, [3]m_souier, [4]z_sari}@mail.univ-tlemcen.dz

***Résumé*- Dans ce papier, nous présentons l'adaptation d'une métaheuristique à base de population de solutions de type hybride dite l'algorithme mémétique qui combine un algorithme génétique et une méthode de recherche locale pour l'amélioration de performances d'un atelier flexible de production avec des machines identiques extrait de la littérature. L'atelier est composé de sept machines et de deux stations, une station de chargement et une station de déchargement. Les critères de performances considérés sont le taux de production du système, et le taux d'utilisation des machines. Pour valider les résultats trouvés, nous avons effectué une comparaison avec ceux trouvés par l'algorithme génétique.**

***Mots clefs*- Systèmes flexibles de production, métaheuristiques hybrides, algorithme mémétique, métaheuristiques.**

***Abstract*- In this paper we present the adaptation of a hybrid metaheuristic based on solutions population, which is the memetic algorithm that combines a genetic algorithm with a local search procedure, for the improvement of a flexible manufacturing system performances with identical machining centers extract from literature. The flexible manufacturing system (FMS) studied is composed from seven machines and two loading and unloading stations. The performances indicators considered are the production rate, and the machining utilization rate. We made a comparison between found results and those obtained by the genetic algorithm.**

***Key words*- Flexible manufacturing system, hybrid metaheuristics, memetic algorithm, metaheuristics.**

I. INTRODUCTION

Le contrôle et l'ordonnancement temps réel des systèmes flexibles de production (FMS) sont devenus un domaine de recherche populaire depuis le début des années 80, période dans laquelle les systèmes flexibles de production ont été adoptés par les pays industriels [1], [2], mais beaucoup d'études dans le contrôle et l'ordonnancement des FMS en temps réel ne prennent pas en considération la flexibilité de routages alternatifs [3], et la plupart des études qui prennent en compte ce point, règlent le problème de la sélection de routages avant le début de la production [4], [5].

Cette approche n'est pas applicable pour les systèmes flexibles de production aléatoires [6], où on ne peut pas prévoir l'arrivée ou l'entrée des pièces dans le système avant le début de la production. Car les routages des pièces peuvent être différents même pour des pièces de même type [6]. Ainsi le système de commande d'un FMS aléatoire est obligé d'utiliser effectivement et efficacement la flexibilité des opérations et de routages en temps réel pour avoir la capacité de s'adapter, avec l'arrivée aléatoire des pièces et des événements imprévus [7], [6]. Pour tirer pleinement partie de la flexibilité offerte par ces systèmes on doit proposer un système de gestion souple, capable de piloter en temps réel, et de prendre dynamiquement et à très court terme les décisions d'allocation et d'ordonnancement des opérations en fonction de l'état du système de production et des objectifs de production.

Dans la majorité des cas, les problèmes d'ordonnancement sont classés NP-difficiles. L'appartenance à la classe des problèmes NP-difficiles signifie qu'on ne pourra probablement jamais résoudre les problèmes de grande taille de façon optimale. Les méthodes de résolution des problèmes d'ordonnancement représentent une grande partie des problèmes d'optimisation combinatoire. En effet pour de tels problèmes, les méthodes exactes requièrent un effort calculatoire qui croit exponentiellement avec la taille du problème (explosion combinatoire). Les méthodes approchées, dites aussi heuristiques ou encore métaheuristiques deviennent l'unique moyen d'obtenir une bonne solution en un temps raisonnable.

Les heuristiques sont beaucoup plus adaptées au contexte industriel, où il ne s'agit pas tant de satisfaire complètement tel ou tel objectif que de fournir une solution d'ordonnancement réaliste, qui procure une satisfaction jugée acceptable d'un ensemble d'objectifs. Pour cette raison, les logiciels d'ordonnancement disponibles sur le marché font tous appel à des méthodes heuristiques [8]. De telles méthodes fournissent des solutions de bonne qualité en un temps raisonnable.

L'adaptation des métaheuristiques pour la résolution de quelques problèmes dans l'industrie peut être justifiée par le développement des machines avec des capacités calculatoires énormes.

Pour être performante, une méthode de recherche doit associer judicieusement exploration et exploitation de l'espace de recherche. Or, une méthode de recherche est rarement aussi efficace pour exploiter que pour explorer l'espace de recherche.

Une solution consiste à ajouter des mécanismes complémentaires dans une méthode de recherche donnée. De ce fait, il peut être extrêmement bénéfique d'associer une méthode de recherche dont la capacité d'exploration est très élevée à une méthode de recherche dont le point fort est l'exploitation de l'espace de recherche, on parle alors des méthodes hybrides.

L'hybridation revient à combiner plusieurs algorithmes de nature différente.

Les métaheuristiques hybrides sont apparues en même temps que le paradigme lui-même, mais la plupart des

chercheurs n'y accordaient que peu d'intérêt [9]. Elles deviennent maintenant populaires, car les meilleurs résultats trouvés par métaheuristiques pour plusieurs problèmes d'optimisation combinatoire ont été obtenus par des algorithmes hybrides [10].

Les algorithmes génétiques ont montré leur efficacité dans un nombre de problèmes d'horizons très divers. Dans le travail de [11] les meilleurs résultats ont été obtenu par l'algorithme génétique, c'est-à-dire, que l'algorithme génétique a permet de bien améliorer certaines performances de système comparés à cinq autres métaheuristiques. Notre idée consiste à améliorer encore les performances du système en introduisant un mécanisme de recherche locale au niveau de l'algorithme génétique pour le rendre plus performant, on parle alors de l'algorithme mémétique.

Cet article s'articule autour de quatre sections. La première est un état de l'art qui regroupe les travaux les plus importants dans la sélection de routages alternatifs. Dans la deuxième nous définissons le contexte de travail présenté. Dans la troisième, nous définissons le mode de fonctionnement de l'algorithme mémétique adapté. La dernière section est consacrée aux résultats trouvés et à leurs interprétations.

II. ETAT DE L'ART

La flexibilité de routages offre au système les moyens d'un aiguillage plus souple, de façons à servir les différents segments de procédés libres ou sous engagés.

Dans la littérature, plusieurs travaux se sont intéressés à l'influence de la flexibilité de routages sur les performances d'un FMS.

Dans [12] les auteurs ont démontré l'effet de variation des niveaux de flexibilité de routages (le nombre de routages alternatifs pris par un type de pièce) sur les performances d'un FMS jugées par le makespan, ils ont proposé dans [13] une stratégie d'ordonnancement prenant en considération la flexibilité de routages basée sur la logique floue dans un FMS.

On peut citer les travaux de [14] qui ont étudié par simulation l'impact de la flexibilité des machines et de routages en variant différentes conditions de l'atelier sur le temps de cycle moyen des pièces, on trouve aussi les travaux de [15] qui ont étudié l'effet des règles de priorité et des niveaux de flexibilité de routage sur les différentes performances d'un FMS.

Citons aussi les travaux de [16] qui ont proposé une plateforme qui intègre le processus de planification flexible et l'ordonnancement prédictif, ils ont présenté un concept nommé Dissimilarity Maximization Method (DMM) pour minimiser la congestion dans un FMS, l'idée de cette règle est de maximiser les dissimilitudes entre les routages occupés, dans leur algorithme, chaque routage ne contient qu'une seule pièce à la fois. L'efficacité de cette règle dans la résolution des problèmes de sélection de routage en temps réel a été démontré dans [17] et [18] où elle a surpassé deux autres règles de sélection de routages alternatifs FIFO/FA (First-In First-Out/ First-Available) et EPL (Equal Probability Loading) si chaque machine utilise la règle FIFO comme règle de séquencement.

Dans l'étude de [19] sur la règle DMM, dans laquelle ils ont proposé la règle DMM-modifiée qui est une modification de la règle DMM visant à garder le même principe qui dépend de la maximisation des coefficients de dissimilitude pour la sélection des différents routages alternatifs mais en affectant plusieurs pièces à un seul routage.

Durant ces dernières années les métaheuristiques connaissent un intérêt croissant dans le domaine d'optimisation combinatoire, la résolution de plusieurs problèmes d'ordonnancement est basée sur ces techniques.

Dans [20] l'auteur fait appel à l'algorithme des colonies de fourmis pour l'ordonnancement d'un d'atelier flow shop hybride monocritère et multicritère. [21] proposent une nouvelle structure d'un système multi agent en utilisant un algorithme de colonies de fourmis (ACO) pour mieux ordonnancer un job shop dynamique. Dans [22] les auteurs ont hybridé un algorithme génétique avec une recherche à voisinage variable pour résoudre le problème d'ordonnancement d'un jobshop flexible. Dans [11] les auteurs ont adapté plusieurs métaheuristiques (les colonies de fourmis, les algorithmes génétiques, les essaims particulaires, le recuit simulé, l'électromagnétisme, et la recherche tabou) pour résoudre le problème de sélection de routages alternatifs en temps réel dans un système flexible de production. Les meilleurs résultats ont été trouvés par l'algorithme génétique.

III. PRESENTATION DU MODELE FMS ETUDIE

Le système étudié est composé de sept machines et deux stations : l'une de chargement et l'autre de déchargement, définies comme suit :

- Deux fraiseuses verticales (FV).
- Deux fraiseuses horizontales (FH).
- Deux tours (T).
- Une toupie (TP).
- Une station de chargement(SC).
- Une station de déchargement (SD).

Chacune des machines a une file d'attente d'entrée et une file d'attente de sortie, la station de chargement a aussi une file d'attente d'entrée. Le système traite six types de pièces différentes. Chaque type de pièces a un certains nombre de routages.

Les opérations sur le système de production étudié sont basées sue les suppositions suivantes :

- Chaque type de pièce a des routages alternatifs connus avant le début de la production.
- Le temps de traitement de chaque type de pièce sur une machine est déterminé, et il comprend le temps de changement des outils et le temps d'exécution de la machine.
- Le temps de traitement d'une opération est le même sur les machines alternatives identifiées pour cette opération.
- Chaque machine ne traite qu'une seule pièce à la fois.

Le système de production étudié peut être présenté comme suit :

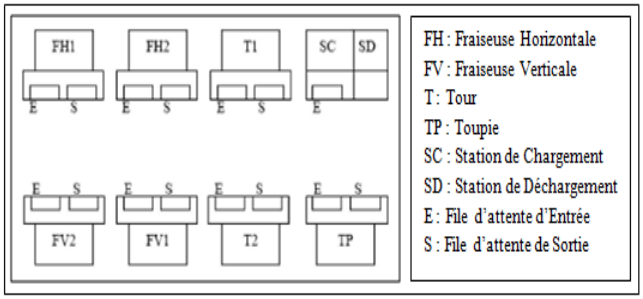

Fig. 1. Configuration du modèle FMS étudié [17].

Le tableau 1 présente les différents types de pièces et leurs taux d'arrivée au système, ainsi que leurs routages possibles avec le temps de traitement sur chaque machine.

TABLEAU I

Les routages alternatifs et les temps de traitement des pièces [17].

Types des pièces	Taux d'arrivée	Routages alternatives et temps de traitement (min)
A	17 %	SC-T_1(30)-FV_1(20)-SD
		SC-T_1(30)-FV_2(20)-SD
		SC-T_2(30)-FV_1(20)-SD
		SC-T_1(30)-FV_2(20)-SD
B	17%	SC-T_1(20)-TP(1)-FV_1(15)-SD
		SC-T_1(20)-TP(1)-FV_2(15)-SD
		SC-T_2(20)-TP(1)-FV_1(15)-SD
		SC-T_2(20)-TP(1)-FV_2(15)-SD
C	17%	SC-T_1(40)-FV_1(25)-SD
		SC-T_1(40)-FV_2(25)-SD
		SC-T_2(40)-FV_1(25)-SD
		SC-T_2(40)-FV_2(25)-SD
D	21%	SC-T_1(40)-TP(1)-T_1(20)-FH_1(35)-SD
		SC-T_1(40)-TP(1)-T_1(20)-FH_2(35)-SD
		SC-T_1(40)-TP(1)-T_2(20)-FH_1(35)-SD
		SC-T_1(40)-TP(1)-T_2(20)-FH_2(35)-SD
		SC-T_2(40)-TP(1)-T_1(20)-FH_1(35)-SD
		SC-T_2(40)-TP(1)-T_1(20)-FH_2(35)-SD
		SC-T_2(40)-TP(1)-T_2(20)-FH_1(35)-SD
		SC-T_2(40)-TP(1)-T_2(20)-FH_2(35)-SD
E	20%	SC-T_1(25)-TP(1)-T_1(35)-FH_1(50)-SD
		SC-T_1(25)-TP(1)-T_1(35)-FH_2(50)-SD
		SC-T_1(25)-TP(1)-T_2(35)-FH_1(50)-SD
		SC-T_1(25)-TP(1)-T_2(35)-FH_2(50)-SD
		SC-T_2(25)-TP(1)-T_1(35)-FH_1(50)-SD
		SC-T_2(25)-TP(1)-T_1(35)-FH_2(50)-SD
		SC-T_2(25)-TP(1)-T_2(35)-FH_1(50)-SD
		SC-T_2(25)-TP(1)-T_2(35)-FH_2(50)-SD
F	8%	SC-FH_1(40)-SD
		SC-FH_2(40)-SD

La présence des machines identiques offre la possibilité à chaque type de pièces d'avoir plusieurs routages, le problème sera donc, à chaque fois qu'une pièce arrive à la station de chargement on doit lui affecter l'un des routages possibles, on doit donc prendre en considération, à chaque fois, l'état du système.

Dans notre travail, le but étant d'avoir un bon équilibre entre les routages dans le système.

IV. L'ALGORITHME MEMETIQUE

Un algorithme mémétique est une hybridation entre un algorithme génétique et une méthode de recherche locale.

En effet, la puissance des algorithmes génétiques provient du fait qu'ils sont capables de balayer de manière globale l'espace des solutions contrairement aux méthodes de descente ou aux heuristiques qui explorent une petite zone de cet espace [23].

Un inconvénient d'un algorithme génétique est que les opérateurs standards de croisement et de mutation ne permettent pas d'intensifier suffisamment la recherche [24]. L'opérateur de mutation apporte une légère modification à l'individu. Son rôle est de favoriser la diversification des individus alors que la sélection se charge de conserver les meilleurs. C'est pourquoi les algorithmes génétiques sont souvent hybridés avec des méthodes de recherche locale [25]. C'est le cas des algorithmes mémétiques de Moscato [26].

En 1989 Moscato [26] introduit pour la première fois les algorithmes mémétiques. Un algorithme mémétique est donc une combinaison, ou encore une coopération entre un algorithme génétique et une méthode de recherche locale.

Ces deux méthodes sont complémentaires car l'une permet de détecter de bonnes régions dans l'espace de recherche alors que l'autre se concentre de manière intensive à explorer ces zones de l'espace de recherche [26]. Ainsi, on peut explorer rapidement les zones intéressantes de l'espace de recherche pour les exploiter en détail [25].

L'idée de Moscato est donc d'ajouter une procédure de recherche locale qui peut être une méthode de descente ou une recherche locale plus évoluée (recuit simulé ou recherche tabou par exemple). Cette recherche locale sera appliquée à tout nouvel individu obtenu au cours de la recherche. Cette simple modification entraine de profonds changements dans le comportement de l'algorithme lui-même. Après avoir créé un nouvel individu à partir de deux parents sélectionnés, on applique une recherche locale et sous certaines conditions on applique un opérateur de mutation à cet individu [27].

Dans cet algorithme, l'opérateur de mutation assure la diversification de la méthode, d'autre part, l'intensification est produite par l'application de la méthode de recherche locale.

Les différentes étapes de l'algorithme mémétique sont données comme suit :

(1) Initialiser : générer une population initiale P de solutions
(2) Appliquer une procédure de recherche locale sur chaque solution de P
(3) Répéter
(4) Sélection : choisir deux solutions *x* et *x'*

(5) Croisement : combiner deux solutions parents *x* et *x'* pour former une solution *y*
(6) Recherche locale : appliquer une procédure de recherche locale sur *y*
(7) Mutation : appliquer un opérateur de mutation sur
(8) Choisir un individu *y'* pour être remplacé dans la population
(9) Remplacer *y'* par *y* dans la population
(10) Jusqu'à satisfaire un critère d'arrêt.

La méthode de recherche locale utilisée dans un algorithme mémétique n'est pas unique, on peut utiliser une méthode de recherche locale simple telle que les méthodes de descente, ou des méthodes plus évoluées telles que le recuit simulé ou la recherche tabou. Dans notre travail, nous avons utilisé une méthode de descente, dans la section suivante, nous allons présenter le principe des méthodes de descente.

A partir d'une solution trouvée par heuristique par exemple, on peut très facilement implémenter des méthodes de descente. Ces méthodes s'articulent toutes autour d'un principe simple. Partir d'une solution existante, chercher une solution dans le voisinage et accepter cette solution si elle améliore la solution courante.

Le squelette de la méthode de descente est donné par les étapes de l'algorithme suivant :

(1) initialisation : trouver une solution initiale x
(2) Répéter
(3) Recherche dans le voisinage : trouver une solution $x' \in N(x)$
(4) Si $f(x') < f(x)$ alors
(5) $x \leftarrow x'$
(6) Fin si
(7) Jusqu'à $f(y) \geq f(x), \forall y \in N(x)$

A partir d'une solution initiale *x*, on choisit une solution x_0 dans le voisinage *N(x)* de *x*. Si cette solution est meilleure que *x*, ($f(x_0) < f(x)$) alors on accepte cette solution comme nouvelle solution *x* et on recommence le processus jusqu'à ce qu'il n'y ait plus aucune solution améliorante dans le voisinage de *x*.

V. ADAPTATION DE L'ALGORITHME MEMETIQUE

Dans cette section nous présentons notre adaptation de l'algorithme mémétique pour la résolution de notre problème.

Toutes les métaheuristiques tentent de trouver une solution visant à minimiser ou maximiser (selon le problème à résoudre) une fonction dite 'fonction objectif'.

Les métaheuristiques sont des algorithmes itératifs, qui tentent d'améliorer une ou plusieurs solutions (une solution dans le cas des méthodes à solution unique, et plusieurs dans le cas des méthodes à population de solutions) au cours des itérations. Il est très difficile de trouver la solution optimale dès la première itération, ce qui justifie l'échec d'adaptation de ces techniques pour la résolution des problèmes complexes en ligne.

Pour pouvoir résoudre notre problème en ligne, nous avons pris en considération à chaque itération des critères concernant l'état du système, à chaque fois qu'une nouvelle pièce entre dans la station de chargement.

Dans notre travail, la fonction objectif est le produit de charges des routages, le but étant de maximiser cette fonction, afin d'équilibrer les charges des routages, on parle de charges non pas en terme de nombre des pièces mais en terme du temps opératoire. La maximisation de cette fonction (le produit des temps opératoires) donnera donc un bon équilibre entre les routages. Pour ce faire, nous avons pris en considération les types des premières pièces existantes dans la station dans la file infinie (*n* est égale à la capacité de la file d'attente de la station de chargement). Par la suite, chaque pièce sera affectée à un routage selon son type.

Cet algorithme est basé sur le même principe des algorithmes génétiques : une population d'individus (solutions) qui évoluent en même temps, chaque solution est représentée sous forme d'un chromosome, on parle alors du codage des solutions. Chaque solution possible du problème est représentée par un chromosome artificiel (généralement sous forme de chaîne). Dans notre cas, un chromosome artificiel représente les routages choisis des pièces.

Le pseudo code de l'algorithme mémétique adapté est donné comme suit :

(1) S'il y'a une place libre dans la station de chargement **alors**
(2) Génération d'une population aléatoire (cela revient dans notre cas à affecter les *n* premières pièces de la file infinie à des routages d'une façon aléatoire en respectant les types des pièces)
(3) Faire la procédure de recherche locale sur chaque individu de la population
(4) Tant que le critère d'arrêt n'est pas atteint faire
(5) Appliquer un opérateur de sélection (c'est-à-dire choisir deux solutions $x\ et\ x'$)
(6) Appliquer l'opérateur de croisement sur les deux parents x et x' pour former deux solutions $y\ et\ y'$
(7) Appliquer la procédure de recherche locale sur $y\ et\ y'$
(8) Appliquer l'opérateur de mutation sur $y\ et\ y'$ (ceci revient à modifier les routages de certaines pièces aléatoirement en respectant les types des pièces)
(9) Appliquer l'opérateur de remplacement
(10) Fin tant que

Dans l'algorithme mémétique, la procédure de recherche locale s'effectue sur chaque individu de la population générée initialement, puis elle s'applique sur chaque nouvel individu obtenu par le croisement, c'est-à-dire sur chaque nouvelle génération de solutions, cela veut dire qu'on tente toujours de choisir les meilleurs individus de la population initiale ou de la génération construite.

Nous avons choisit comme méthode de recherche locale la méthode de descente. C'est une méthode basée sur l'exploration du voisinage d'une solution. C'est-à-dire qu'à partir d'une solution trouvée on fait une recherche qui passe d'une solution à une autre solution voisine, et à chaque fois qu'on trouve une solution améliorante on l'utilise comme nouvelle solution de départ, et la nouvelle recherche s'effectue dans le voisinage de la nouvelle solution trouvée.

Le pseudo code de la procédure de recherche locale appliquée sur les individus de la population initiale, ou résultants par croisement est donné comme suit :

(1) Pour chaque individu (qui représente une condition initiale *x* pour cette méthode)
(2) Tant que la condition d'arrêt n'est pas vérifiée faire
(3) Modifier les routages de certaines pièces (trouver une autre solution voisine x' de x)
(4) Si la fonction objectif est améliorée (produit des charges des routages) : $f(x') > f(x)$ **alors**
(5) Remplacer x par x'
(6) Finsi
(7) Fin tant que

VI. RESULTATS ET DISCUSSIONS

Dans cette section nous présentons les résultats obtenus par l'algorithme mémétique adapté pour la résolution de notre problème. Les résultats obtenus sont comparés à ceux de l'algorithme génétique adapté dans [11] pour la résolution du même problème.

Les expérimentations ont été mises en œuvre sur un PC équipé d'un Intel (Core TM 2 Duo CPU avec 2.2 Ghz et 2 Go de RAM).

Les résultats concernent deux critères de performances, le premier étant le taux de production, et le deuxième étant le taux d'utilisation des machines.

A. Le taux de production

Le taux de production se définit dans ce qui suit comme étant le taux de sortie des pièces, il se calcule en divisant le nombre de pièces sortantes du système par le nombre de pièces entrantes.

Les simulations sont effectuées en fixant le taux de création des pièces (taux d'arrivée des pièces) et en variant la capacité de file d'attente.

Les résultats sont donnés pour un taux d'arrivée des pièces égale à 1/5min (c'est-à-dire les pièces arrivent toutes les 5 minutes) et 1/20min.

TABLEAU II

Le taux de production pour un taux de création des pièces =1/5min

Capacité de la file d'attente	2	4	6	8
Algorithme mémétique	24.59	24.72	24.84	24.82
Algorithme génétique	23.70	23.93	24.17	24.24

Le tableau II donne l'évolution du taux de production pour différentes valeurs de la capacité de file d'attente. Il est clair que l'algorithme mémétique donne de meilleurs résultats comparés à ceux de l'algorithme génétique. De plus nous remarquons que la variation de la capacité de file d'attente n'influx pas beaucoup sur le taux de production, c'est-à-dire, le taux de production varie d'une manière uniforme.

TABLEAU III

Le taux de production pour un taux de création des pièces = 1/20

Capacité de la file d'attente	2	4	6	8
Algorithme mémétique	99.99	99.99	99.99	99.99
Algorithme génétique	98.44	99.99	99.99	99.99

Le tableau III donne l'évolution du taux de production pour différentes valeurs de la capacité de file d'attente pour un système moins saturé (taux d'arrivée des pièces=1/20). Pour une capacité de file d'attente égale à 2 l'algorithme mémétique est plus performant, hors de cette valeur, les résultats sont identiques.

B. Le taux d'utilisation des machines

Le taux d'utilisation des machines a été définie par le pourcentage du temps d'utilisation des machines par rapport au temps de simulation.

Dans cette section nous présentons les taux d'utilisation des machines T, FV, et FH.

TABLEAU IV

Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente égale à 2

Taux de création des pièces	1/25	1/20	1/15	1/10	1/5
Algorithme mémétique	76.15	93.51	92.22	92.39	92.66
Algorithme génétique	75.67	92.72	90.98	90.85	90.96

Le tableau IV représente les taux d'utilisation des machines T_1 et T_2 pour différentes valeurs de taux de création des pièces pour une capacité de file d'attente égale à 2. Les résultats sont proches, mais l'algorithme mémétique est plus performant.

TABLEAU V

Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente égale à 2

Taux de création des pièces	1/25	1/20	1/15	1/10	1/5
Algorithme mémétique	24.94	33.29	32.26	32.43	32.48
Algorithme génétique	25.54	31.94	28.60	28.36	28.72

Le tableau V donne l'évolution du taux d'utilisation des machines FV_1 et FV_2 pour différentes valeurs du taux de création des pièces pour une capacité de file d'attente égale à 2. Il est nettement clair que les résultats obtenus par l'algorithme mémétique sont meilleurs que ceux de l'algorithme génétique.

TABLEAU VI

Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente égale à 2

Taux de création des pièces	1/25	1/20	1/15	1/10	1/5
Algorithme mémétique	34.72	40.45	40.37	40.36	40.28
Algorithme génétique	33.88	40.99	42.47	42.57	42.36

Le tableau VI montre les taux d'utilisation des machines FH_1 et FH_2 en fonction des taux de création des pièces pour une capacité de file d'attente égale à 2. Les résultats de l'algorithme génétique sont meilleurs que ceux obtenus par l'algorithme mémétique sauf pour un taux de création des pièces égale à 1/20. Malgré ça les résultats sont proches.

VII. CONCLUSION

Dans ce travail, nous nous sommes intéressés à l'adaptation de l'algorithme mémétique, qui représente une métaheuristique hybride combinant un algorithme génétique et une méthode de recherche locale, pour la résolution d'un problème d'ordonnancement dans un système flexible de production. Le but étant l'amélioration de certains critères de performance du système.

Les résultats obtenus sont comparés à ceux de l'algorithme génétique afin de montrer l'amélioration apportée à celui-ci en lui introduisant une procédure de recherche locale. Les résultats (sauf ceux du taux d'utilisation des machines FH) montrent la performance de l'algorithme mémétique.

Le fait d'associer une procédure de recherche locale à un algorithme génétique lui permet de bien intensifier la recherche dans les zones de l'espace de recherche les prometteuses. Donc, les deux méthodes, algorithme génétique et recherche locale sont complémentaires, car l'algorithme génétique permet de bien balayer l'espace de recherche, tandis que la recherche locale permet de bien intensifier la recherche dans ces zones pour trouver les meilleures solutions.

La méthode de recherche locale que nous avons utilisé est de descente très simple, il sera donc bénéfique et intéressant d'utiliser une autre méthode de recherche locale plus évolué qui sera donc plus efficace (une méthode de descente plus performante telle que la grande descente ou encore une métaheuristique à solution unique telle que le recuit simulé).

REFERENCES

[1] C. Saygin & S.E. Kilic, (1995). "Dissimilarity Maximization Method for Real-time Routing of Parts in Random Flexible Manufacturing Systems", *The International Journal of Flexible Manufacturing Systems*, 16, 1995, pp. 169-182.

[2] C. Peng & F.F. Chen, "Real-time control and scheduling of flexible manufacturing systems: a simulation based ordinal optimization approach", *International Journal of Advanced Manufacturing Technology*, 14(10), (1998), pp. 775-786.

[3] A. Kazerooni, F.T. Chan, & K. Abhary, "A fuzzy integrated decision-making support system for scheduling of FMS using simulation", *Computer Integrated Manufacturing Systems*, 10(1), 1997, pp. 27-34.

[4] S.K. Das & P. Nagendra, "Selection of routes in a flexible manufacturing facility", *International Journal of Production Economics*, 48, 1997, pp. 237-247.

[5] H. Cho & R.A. Wysk, "Intelligent workstation controller for computer-integrated manufacturing: problems and models", *Journal of Manufacturing Systems*, 14(4), 1995, pp. 252-263.

[6] R. Rachamadugu & K.E. Stecke, Classification and review of FMS scheduling procedures, Production Planning and control, 5, 1994, pp. 2-20.

[7] A.G. Mamalis, I. Malagardis, & E. Pachos, On-line Scheduling in Metal Removel Processing Using Variable Routing and Control Strategies, Computer Integrated Manufacturing Systems, 8, 1995, pp. 35-40.

[8] E. Talbi, "Sélection et Réglage de Paramètres pour l'Optimisation de Logiciels d'Ordonnancement Industriel", *Thèse de doctorat, Institut National Polytechnique de Toulouse*, 2004.

[9] C. Cotta, E-D, Talbi, & E. Alba, Parallel hybrid Metaheuristics: A new Class of Algorithms, Wiley-Interscience, 2005, pp.347-370.

[10] S. Noël, "Métaheuristiques hybrides pour la résolution du problème d'ordonnancement de voitures dans une chaîne d'assemblage automobile", *Mémoire présentée comme exigence partielle de la maitrise en informatique, Université du Québec*, 2007.

[11] M. Souier, A. Hassam, Z. Sari, Metaheuristics for Real Time Routing Selection in FMS, Book chapter in: L. Benyoucef & B. Grabot (ED.), *Artificial intelligence techniques for networked manufacturing enterprises management*, Springer-Verlag, ch. 2010, pp.221-247.

[12] R. Caprihan & S. Wadhwa, "Impact of routing flexibility on the performance of an FMS-A simulation study", *International Journal of Flexible Manufacturing Systems* 9(3), 1997, pp.273-298.

[13] R. Caprihan, A. Kumar, K.E. Stecke, "A fuzzy dispatching strategy for due date scheduling of FMSs with information delays", *International Journal of Flexible Manufacturing Systems* 18(1), 2006, pp. 29-53.

[14] H. Tsubone & M. Horikawa, "Comparison between Machine flexibility and routing Flexibility", *The international journal of flexible manufacturing systems*, 11, 1999, pp. 83-101.

[15] F. Mahmoodi & C.T. Mosier, "The effects of sheduling rules and routing flexibility on the performance of a random flexible manufacturing system", *International journal of flexible manufacturing systems*, 1999, pp. 271-289.

[16] C. Saygin & S.E. Kilick, "Integrating Flexible Process Plans with Scheduling in Flexible Manufacturing System," *International Journal of Advanced Manufacturing Technology*, Springer, vol.15, 1999, pp. 268-280.

[17] C. Saygin, F.F. Chen, J. Singh, "Real time manipulation of alternative routings in flexible manufacturing systems: A simulation study", *International journal of advanced manufacturing technology*, 18, 2001, pp. 755-763.

[18] L. Ghomri, "Influence des contraintes et des perturbations sur les performances des règles de routages dans un FMS", *la conférence internationale Conception et Production Intégrées*, 2007.

[19] A. Hassam & Z. Sari, "Selection of alternative routings in real time: DMM and modified DMM rules*", International Journal of Product Development* 10(1/2/3), 2010, pp.241-258.

[20] S. Khalouli, (2010), "Métaheuristiques à base de modèles : applications à l'ordonnancement d'atelier flow-shop hybride monocritère*", Thèse de doctorat, Université de Reims Champagne-Ardenne*, 2010.

[21] Kang K, Zhang R, Yang Y, 2007. MAS Equipped with Ant Colony Applied into Dynamic Job Shop Scheduling. Book chapter in: Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Book series: Lecture Notes in Computer Science 4682, pp.823-835.

[22] G. Zhang, L. Gao, X. Li, P. Li, "Variable neighborhood genetic algorithm for the flexible job shop scheduling problems", Lecture Notes in Computer Science 5315, 2008, pp.503-512.

[23] P. Rebreyend, "Algorithmes génétiques hybrides en optimisation combinatoire", *Thèse de doctorat, l'école Normale Supérieure de Lyon*, 1999.

[24] H.H. Hoos & T. Stüzle, Stochastic local search: Foundations and applications, Morgan Kauffman, 2004, pp. 43-85.

[25] J.C.H. Hernandez, "Algorithmes Métaheuristiques hybrides pour la sélection de gènes et la classification de données de biopuces", *Thèse de doctorat, Université d'Angers, 2008.*

[26] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms", *Technical Report C3P 826, Cal-teech Concurrent Computation Program,* 1989.

[27] M. Sevaux, "Métaheuristiques Stratégie pour l'optimisation de la production de biens et de services", *Habilitation à diriger des recherches, Laboratoire d'Automatique, de Mécanique d'informatique Industrielles et Humaines du CNRS (UMR CNRS 8530) dans l'équipe systèmes de production*, 2004.