Algorithme génétique pour le problème d'ordonnancement de type job-shop.

Kheireddine MERHOUM , Messaoud DJEGHABA
Laboratoire d'Automatique et de Signaux de Annaba. -LASA-
Université Badji Mokhtar, BP12, Annaba, Algérie.
merhoum_kheirddine @ yahoo.fr djeghaba_mes @ yahoo.com

Résumé : **Dans ce papier nous développons une application qui permet de minimiser le makespan pour un problème d'ordonnancement job-shop flexible en utilisant les Algorithme génétique. Cependant le problème d'ordonnancement job-shop flexible dans la littérature est considéré comme une problématique difficile à résoudre dans le domaine de l'optimisation combinatoire, sa complexité est de type NP-complet au sens fort. Notre objectif est de montrer les performances des algorithmes génétiques (métaheuristque) dans la résolution de ce genre de problème.**

Mots clés : **Système de production flexible, Ordonnancement, Job-shop, Méta-heuristique, Algorithmes génétiques, Makespan.**

I. INTRODUCTION

Les problèmes d'ordonnancement sont en général difficiles et d'une grande utilité pratique. La réalisation d'un projet suppose l'exécution préalable de multiples opérations soumises à de nombreuses contraintes. Résoudre un problème d'ordonnancement consiste à déterminer l'ordre et le « calendrier » d'exécution de ces opérations en leur attribuant des ressources et des dates de début, de manière à réaliser le projet.

Nous nous intéressons plus particulièrement aux problèmes d'atelier, où les ressources sont des machines et les tâches à ordonnancer des opérations à réaliser sur ces machines. Les principaux problèmes sont de type Flow shop, job-shop ou open shop [4] [3].

La plupart des problèmes industriels sont d'une telle « complexité » que le nombre de solutions potentielles croît exponentiellement avec la taille du problème. Les méthodes exactes deviennent rapidement inutilisables pour des instances de grande taille. Il s'avère alors nécessaire d'utiliser des méthodes heuristiques pour les résoudre en un temps raisonnable.

Les méthodes heuristiques offrent l'avantage de ne parcourir qu'une fraction de l'espace de recherche pour parvenir à une solution acceptable. Dans le cadre de cette étude, nous nous intéressons aux méta Heuristique [5] [7]. Il existe un grand nombre de méta heuristiques dont certaines sont inspirées de processus naturels (recuit simulé, algorithmes génétiques, colonies de fourmis,…).

Dans notre travail, nous nous limitons aux algorithmes génétiques. L'objectif essentiel de cette étude concerne l'évaluation des performances des algorithmes génétiques sur la résolution d'un problème d'ordonnancement de type job-shop simple.

II. LES SYSTEMES FLEXIBLES DE PRODUCTION

La compétition entre les entreprises a conduit à chercher à améliorer la production tout en respectant le délai de livraison. Pour satisfaire ce t objectif, les systèmes de production doivent être flexibles et c'est ainsi que sont apparus les Systèmes Flexibles de production (SPF) ou FMS (Flexible Manufacturing system).

Un système de production est dit flexible s'il peut assurer la production simultanée de plusieurs types de pièces avec des quantités variables et s'il est capable de s'adapter à la production de nouveaux produits pour lesquels le système n'a pas été étudié[2][3].

Les SFP sont des systèmes dynamiques. Ils sont constitués d'un ensemble d'entités dont l'état peut changer en temps réel par exemple une ressource qui est en état de production d'une pièce peut passer à état non productif après la fin d'usinage de cette pièce. La dynamique est aussi réalisée par l'interaction entre ces entités. Pour garantir l'objectif de la production, le système de production doit être doté, en plus de la flexibilité, de trois autres caractéristiques qui sont la réactivité, la pro activité et la robustesse. On peut décomposer un système de production en deux parties complémentaires, la partie opérative et la partie conduite.

La partie opérative appelée encore système de fabrication est constituée des entités appartenant à trois populations : la population des produit, la population des moyens de production (machines) et la population des opérateurs de production. Les produits visitent les machines pour subir les opérations nécessaires à leur transformation. La description des opérations à effectuer sur le produit, ainsi que les machines nécessaires, constituent la gamme de fabrication (gamme d'usinage). Cette dernière est fournie par l'ordonnancement prévisionnel qui représente l'objectif développer dans cette communication.

La partie conduite a pour rôle d'élaborer les ordres nécessaires à la partie opérative, de guider et piloter afin de maintenir la cohérence d'un système dans un environnement donné et de satisfaire les objectifs de la production. La conduite du système assure la mise en œuvre du plan de fabrication, le respect de l'objectif de production issus de l'étape de planification et le respect de l'exécution des gammes d'usinage des produits fournis par l'étape de l'ordonnancement prévisionnel.

III. FORMULATION DU PROBLEME JOB SHOP

On désigne sous le terme « job-shop JxM », une instance du problème de job-shop simple constituée de J produits et M machines. Un job job-shop est dit simple si chaque produit est constitué d'une seul gamme de fabrication, et si chaque opération de la gamme ne peut être effectuée que sur une seule machine. Le coût global de production est lié au temps nécessaire à la fabrication des différents produits, l'objectif consiste à réduire la durée globale de fabrication, appelée « makespan » [4] [10] [11].

Le problème est défini comme suite :

- On considère la réalisation de *N* jobs (produits) indexé par j, les jobs sont indépendant l'un de l'autre.
- Chaque job j est défini par une gamme de fabrication noté Gj qui représente un ensemble d'opérations n sur lequel une relation de précédence est définie, Gj = ({O1j, O2j,…, Oij,..., Onj}, <).
- La réalisation de chaque opération Oi,j nécessite une ressource (machine) k affecter parmi l'ensemble des machines M, (k =1…m avec m= le nombre total de l'ensemble des machines).
- On note Pijk la durée opératoire de l'opération Oij sur la machine k.
- L'opérateur de précédence < : définit un ordre total sur les opération d'un gamme opératoire.
- Une machine ne peut exécuter qu'une seule opération à la fois (contrainte de disponibilité).
- L'exécution d'une opération ne peut pas être interrompu (condition de non préemption).
- Soit $D^k(x)$, la date de fin de la dernière opération réalisée sur la machine k selon l'ordonnancement x. Le makespan $C_{max}(x)$ de l'ordonnancement x est défini par :

$$C_{\max}(x) = \max_{1 \le k \le m} D^k(x)$$

  Le makespan est la durée totale nécessaire à la fabrication de tous les produits de l'instance considérée selon l'ordre de fabrication imposé par l'ordonnancement x.
- Une solution qui vérifie les contraintes du problème est dite admissible. Notre objectif est de trouver un ordonnancement admissible qui permet de minimiser la valeur du makespan.

IV. ALGORITHMES GENETIQUES

Les Algorithmes génétiques s'attachent à reproduire l'évolution naturelle d'individus en respectant la loi de survie énoncée par darwin. Les principes de base des Algorithmes génétiques développés initialement par Holland pour répondre à des besoins spécifiques en biologie, ont été rapidement appliqués pour résoudre et avec succès les problèmes d'optimisation combinatoire en recherche opérationnelle et les problèmes d'apprentissage dans le domaine de l'intelligence artificielle [8] [9] [12]. Portmann, introduit et explique comment appliquer les algorithmes génétiques aux problèmes d'ordonnancement. Dans le cadre de l'application des algorithmes génétique dans un problème d'optimisation combinatoire, une analogie est développée entre un individu dans une population et une solution d'un problème dans l'espace de solutions globale.

Dans le cadre de l'optimisation et par analogie, une solution (individu) est codée par une structure de données qui représente son empreinte génétique, on associe a chaque individu une valeur de la fonction critère, celle-ci permet de lui attribuer sa force d'adaptation « fitness » pour simuler la sélection naturelle, de sorte que les solutions pour lesquelles la valeur du critère est plus éloignée de la valeur optimale soient les plus facilement éliminées.

L'exploration de l'espace de solution s'effectue grâce à l'application des mécanismes des opérateurs génétique le croisement et la mutation sur une population initiale définit aléatoirement. Leur déroulement sur plusieurs générations dont le nombre est fixé par le critère d'arrêt, nous permet d'espérer obtenir une solution proche de la solution optimale, ou même une solution optimale.

Le déroulement de l'AG standard peut être résumé comme suit : - Génération de la population initiale.

- Sélection.
- Reproduction (croisement et mutation).
- Remplacement par la nouvelle population.

Algorithme génétique

```
Début
  Générer population_initiale.
  Calculer fitness_individu.
      Tant que « critère d'arrêt non satisfait » Faire
        Générer  Population_Sélectionner.
        Générer  Population_Croisement.
        Générer  Population_ Muté.
        Générer  Population_Remplacement.
        Calculer  fitness_individu.
      Fin tant que.
Fin.
```

V. JOB SHOP ET ALGORITHMES GENETIQUES

A. Génération de la Population Initiale

Plusieurs méthodes existent pour définir le mécanisme de la génération de la population initiale qui représente le point de départ pour la constitution des générations futures : le tirage aléatoire, les heuristiques ou une combinaisons de solution heuristiques et aléatoire [4] [10] [11].

Table I
Représentation du codage d'un individu i

Machines → Opérations

m1	Oi,j	….
m2	Oi',j'	……
…..	……	……

Une solution ou l'individu i qui compose cette population est codée par une matrice de séquence d'opérations qui définie l'ordre des opérations que doit effectuer chaque machine comme le montre la table I. Les solutions constituants la population initiale sont choisies aléatoirement dans cette étude.

Algorithme génération de la population initiale

Début
Fixer la taille n de la population initiale.
Générer une première solution.
k=1
Tant que « k ≠ n» **Faire**
Tester la réalisabilité :
Si « solution sans cycle » **Alors**
Si « solution non existante » **Alors**
- Solution k acceptée (solution réalisable).
k = k + 1.
Sinon
- Solution refusée (solution existe déjà).
Fin si ;
Sinon
- Solution refusée (solution non réalisable).
Fin si ;
Générer une autre solution.
Fin tant que ;
Fin.

B. Sélection.

La phase de sélection consiste à choisir parmi les N individus i de la population courante les plus forts individus à partir desquels la génération suivante sera créée. Soit pour un ordonnancement i réalisable, on calcul la valeur de la fonction objectif fi « Makespan » pour notre problème c'est évaluer le plus long chemin sur le graphe disjonctif. On associe à chaque solution une fonction d'évaluation pour calculer sa force d'adaptation Fi, dans notre cas le problème à pour but de minimiser la fonction objectif min f (i), alors la force de l'individu i (fitness) peut être exprimer tout simplement par *Fi = 1/fi*, Les individu ainsi sélectionnés constituent une population intermédiaires.

Algorithme de sélection

Début
pour i = 1 jusqu'à N **Faire**
Générer un individu i (ordonnancement réalisable).
Calculer Makespan _ individu i.
Calculer fitness_individu i.
Rangement.
Calculer Probabilité_sélection Pi
Générer un nombre x aléatoirement x ∈ [0, 1].
Si x < Pi **alors**
Sélectionner l'individu i.
Fin si,
Fin pour,
Fin.

Il existe différente stratégie de sélection, tel que la sélection par la Roue de Loterie (Roulette Wheel Selection) ou La sélection par la méthode de Tournois.

Dans cette étude nous avons utilisé la méthode de sélection par Rang (Rankine) de classement : on affecte à chaque individu un rang *Ri* qui dépend de son fitness, les individus d'une population sont ensuite classés dans une liste selon l'ordre croissant de leur fitness, on associe pour chaque individu i une probabilité Pi d'être sélectionné, la sélection sera proportionnelle à leur rang dans la liste de la population :

$$pi = R_i \Big/ \sum_{i=1}^{N} R_i$$

C. Croisement (crossover)

L'opérateur de croisement est le plus important dans les AGs, il permet d'explorer efficacement l'espace de recherche. L'opérateur de croisement appliquer sur deux individu parent1 et parent2 choisis parmi la population sélectionnée, permet de générer deux nouvelles solutions enfant1 et enfant2 par combinaison des propriétés des parents 1 et 2.

Algorithme de croisement

Début
Sélectionner aléatoirement deux parents *indiv1* et *indiv2*.
Sélectionner aléatoirement une machine k parmi m.
Croisement :
Pour « i = 1 jusqu'à m » **Faire**
Si « i ≠ k » **Alors**
enf1 reçoit les mêmes affectations que *indiv1*.
enf2 reçoit les mêmes affectations que indiv2.
Sinon
enf1 reçoit les affectations de *indiv2*.
enf2 reçoit les affectations de *indiv1*.
Fin si ;
Fin de pour ;
Tester la réalisabilité :
Si « *enf1*, *enf2* sans cycle » **Alors**
- *enf1 et 2* acceptés (solutions réalisable)
Sinon
- *enf1 et 2* refusés (solutions non réalisables)
Fin si ;
Fin .

La méthode implémenté dans cette étude consiste à choisir aléatoirement une machine k et on applique un croisement entre la séquence des opérations associé à cette machine seulement, le séquencement des opérations sur les autres machines reste identique sur les fils, puis on teste la réalisabilité des solutions obtenu, ceux qui ne comporte pas de cycle seront acceptées.

Table I
Représente les des parents indiv1, indiv2 sélectionner pour le croisement sur la machine 2.

M1	$O_{2,2}$	$O_{1,1}$	$O_{3,3}$
M2	$O_{3,1}$	$O_{2,3}$	$O_{1,2}$
M3	$O_{1,3}$	$O_{2,1}$	$O_{3,2}$

M1	$O_{1,1}$	$O_{2,2}$	$O_{3,3}$
M2	$O_{2,3}$	$O_{1,2}$	$O_{3,1}$
M3	$O_{2,1}$	$O_{1,3}$	$O_{3,2}$

Table II
Les enfants *enf1 et enf2* obtenu après croisement

M1	$O_{2,2}$	$O_{1,1}$	$O_{3,3}$
M2	$O_{2,3}$	$O_{1,2}$	$O_{3,1}$
M3	$O_{1,3}$	$O_{2,1}$	$O_{3,2}$

M1	$O_{1,1}$	$O_{2,2}$	$O_{3,3}$
M2	$O_{3,1}$	$O_{2,3}$	$O_{1,2}$
M3	$O_{2,1}$	$O_{1,3}$	$O_{3,2}$

D. Mutation :

Le deuxième opérateur génétique important est la mutation, elle vient en deuxième place sur le plan d'importance par rapport au croisement. Une mutation est une perturbation introduite sur la composante de l'individu afin de garantir la diversité et élargir le champ d'exploration.

La démarche suivie dans ce travail consiste à choisir aléatoirement un individu et la machine qui doit subir cette perturbation, ensuite on choisit aléatoirement deux opérations, l'opération consiste à permuter l'ordre entre eux.

Algorithme de mutation

Début
Sélectionner aléatoirement un individu i.
Sélectionner aléatoirement une machine k parmi m.
Pour « h = 1 jusqu'à m » **Faire**
Si « $h \neq k$ » **Alors**
Indiv muté reçoit les mêmes affectations que *indiv i*.
Sinon
Sélectionner aléatoirement deux opérations $O_{i,j}$ et $O_{i',j}$
Permuter l'ordre des opérations $O_{i,j}$ et $O_{i',j'}$.
Fin si ;
Fin de pour ;
Tester la réalisabilité :
Si « solution sans cycle » **Alors**
Indiv muté acceptés (solution réalisable)
Sinon
Indiv muté refusés (solutions non réalisable)
Fin si ;

Fin.

Table III
Individu i avant permutation sur la machine 2

M1	$O_{2,2}$	$O_{1,1}$	$O_{3,3}$
M2	$O_{3,1}$	$O_{2,3}$	$O_{1,2}$
M3	$O_{1,3}$	$O_{2,1}$	$O_{3,2}$

Table IV
Individu i après permutation

M1	$O_{2,2}$	$O_{1,1}$	$O_{3,3}$
M2	$O_{2,3}$	$O_{3,1}$	$O_{1,2}$
M3	$O_{1,3}$	$O_{2,1}$	$O_{3,2}$

E. Remplacement

Cette étape constitue la population de la génération suivante à partir des parents et des enfants de la génération courante. Une fraction de la population est remplacée par sa descendance à chaque génération. L'écart entre les générations indique la proportionnel de parents remplacés par des enfants.

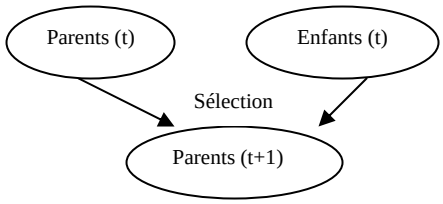

Fig. 2. Le Remplacement, cette phase permet de ne retenir que les meilleurs éléments entre enfants et parents.

Algorithme de remplacement

Début
Rangement de la population des parents.
Rangement de la population enfants.
Remplacement :
Choisir les N meilleurs individus parmi les enfants et les Parents.
Fin.

F. Critère d'arrêt

Le processus est stoppé au bout d'un nombre fixé de génération, ou lorsque les individus ont convergé vers une ou plusieurs solutions satisfaisantes, dans notre étude nous avons opté pour la première démarche.

Les meilleurs individus de la population sont alors retenus comme solutions au problème.

VI. EXPERIMENTATION

Un système de production flexible FMS est souvent défini comme étant un système de production complexe. Plusieurs machines, robots et systèmes de convoyages collaborent à la réalisation d'un ensemble d'opérations comme l'assemblage l'usinage, le chargement, le déchargement et le stockage d'un produit. La cellule de production flexible pilote, développée au LAGIS (LILLE) a été pris comme site expérimental pour simulé le fonctionnement des algorithmes génétiques sur le problème job shop.

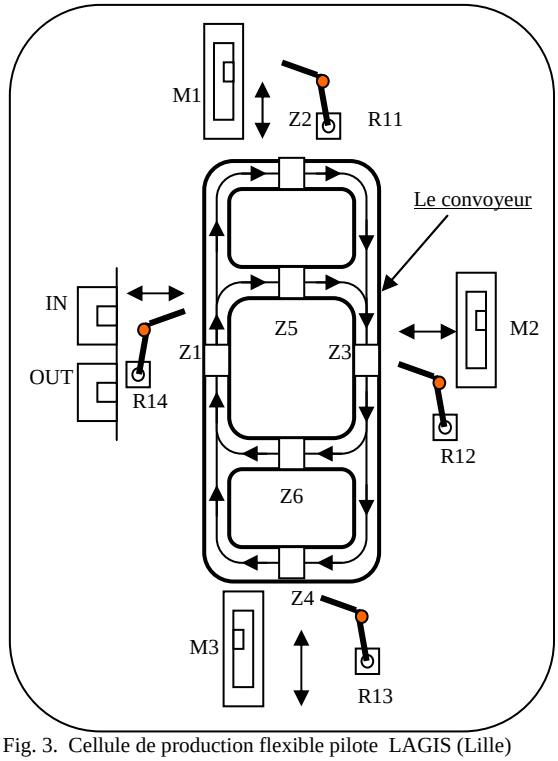

Fig. 3. Cellule de production flexible pilote LAGIS (Lille)

Les ressources principales de cette cellule sont trois machines appelées M1, M2 et M3. Elles exécutent respectivement les fonctions d'usinage : M1 (f1, f3), M2 (f1, f2) et M3 (f2, f3), avec :

f1 : programme pièce de l'opération de perçage.
f2 : programme pièce de l'opération de fraisage1. f3 : programme pièce de l'opération de fraisage2.

Le robot R14 permet les opérations d'alimentation de la cellule en pièces de la file d'attente FIFO IN vers la zone Z1 ainsi que des opérations d'évacuation de cellule par le transfert des pièces de la zone Z1 vers FIFO OUT.

Le même robot R14 permet le transfert des pièces de la zone Z1 vers la zone Z4 et vis versa. Les robots R11, R12 et R13 assurent le chargement et le déchargement des machines M1, M2 et M3.

Le convoyeur CV assure le transfert des pièces entre machines, dans les deux sens, entre les zones de Z1, Z2 et Z5 et aussi de la Z3, Z4 et Z6.

VII. RESULTATS NUMERIQUES

Teste n°1 :

Nous considérons le problème JSP 3x3 qui représente l'exécution de trois jobs sur trois machines, pour chaque job j on associe trois opérations, on numérote les opérations par des entiers, les durées opératoires et les machines utilisées pour exécuter les opérations sont fournies dans la table V.

Les paramètres des algorithmes génétiques utilisés sont les suivants :

- La taille de population initiale (Nbr_pop) = 20
- Probabilité de croisement (Pc) = 0.7
- Probablité de mutation (Pm) = 0.01
- Nombre de génération (NB_g) = 50

Table V
Les durées opératoires et les affectations machines des opérations

Jobs	Opérations	N°	M_1	M_2	M_3
J_1	$O_{1,1}$	1	6	x	x
	$O_{2,1}$	2	x	x	12
	$O_{3,1}$	3	x	9	x
J_2	$O_{1,2}$	4	x	16	x
	$O_{2,2}$	5	14	x	x
	$O_{3,2}$	6	x	x	5
J_3	$O_{1,3}$	7	x	x	22
	$O_{2,3}$	8	x	18	x
	$O_{3,3}$	9	9	x	x

Les graphes disjonctifs correspond à ce teste permet de représenter les opérations par des nœuds, les contraintes de précédence sont modélisées par les arcs conjonctifs et les contraintes de capacité des machines par les arcs disjonctifs sont données dans les figure (4, 5,6).

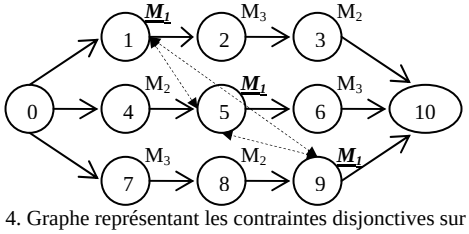

Fig. 4. Graphe représentant les contraintes disjonctives sur M1

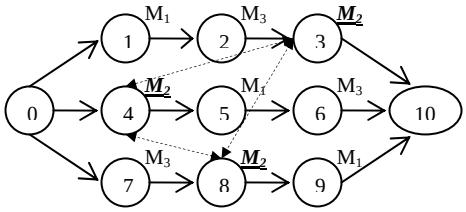

Fig. 5. Graphe représentant les contraintes disjonctives sur M2

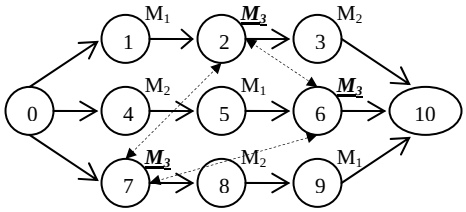

Fig. 6. Graphe représentant les contraintes disjonctives sur M3

Un ordonnancement réalisable doit être tel que une direction est choisie pour les arcs disjonctifs de façon à former un graphe acyclique. Son « makespan » représente la longueur du plus long chemin allant du nœud 0 au nœud 10.

La table VI représente la matrice des contraintes conjonctifs de chaque job, les opérations de chaque sont ordonné suivant la gamme opératoire du job, dans la matrice on associe un 1 si $< O_{i,j}$ précède $O_{i+1,j} >$ sinon un 0, chaque solution dite réalisable doit prendre en compte cette contrainte.

Table VI
Matrice de contraintes conjonctives

N°	1	2	3	4	5	6	7	8	9
1	0	1	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	0	0	0
5	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	0	0	0	0

La table VII représente la solution réalisable, c'est-à-dire l'ordre établi des opérations sur chaque machine et la table VIII spécifie les dates de début et la date de fin de chaque opération.

Table VII
Solution représentant un ordonnancement réalisable

machines	Opérations		
m1	1	5	9
m2	4	8	3
m3	2	6	7

Table VIII
Solution représentant un ordonnancement réa

machines	Opérations	Date de début	Date de fin
M_1	1	0	6
	5	16	30
	9	40	49
M_2	4	0	16
	8	22	40
	3	40	49
M_3	2	0	22
	6	22	34
	7	34	39

Le « Makespan » de cette solution est égale à 49 unités de temps.

Les testes que nous avons effectuer consiste à augmenté à chaque fois le nombre de jobs qui varie en gardant le nombre de machines fixer à trois qui correspond à la structure du site expérimentale, nous avons fixer le nombre d'opérations par job suivant le nombre de machines comme le montre les graphes disjonctifs.

Table XII
Les résultats des testes

Type de problème étudier	Nombres d'opérations par job	Le nombre de solution globale	Nombre de génération
3x3	(3,3,3)	216	50
4x3	(3,3,3,3)	13 824	100
5x3	(3,3,3,3,3)	1 728 000	200
6x3	(3,3,3,3,3,3)	373 248 000	400
7x3	(3,3,3,3,3,3,3)	128 024 064 000	600

Table XII (suite)

Nombre de solution optimale	Valeur du makespan (temps)	Temps d'execution (seconde)
2	49	00.00
2	59	06.40
2	73	11.00
3	99	17.00
2	122	25.00

VIII. CONCLUSION

Dans cette communication nous avons utilisé les algorithmes génétique pour traiter le problème d'ordonnancement job-shop, connu dans la littérature par leur appartenance à la classe des problèmes NP-complet, et considérer parmi les problèmes d'optimisation combinatoire les plus difficile. Le makespan représente notre objectif à minimiser.

Le choix de cette méta-heuristique est conditionné par l'étude des problèmes de grande taille, elle permet de construire des solutions de bonne qualité avec un temps raisonnable du calcul.

Les testes effectuer consistaient à fixer le nombre de machines suivant le site expérimentales et pour chaque job on fixe le nombre des opérations suivants le nombre de machine utilisées où dans chaque job il doit réalisé au moins une opération sur chaque machine.

Le nombre de solutions globale pour chaque problème est lier seulement au nombre de job et au nombre de machines utilisée, dans notre étude nous avons seulement changé le nombre de job pour chaque teste avec un nombre fixe de machine que nous considérer à trois, on remarque que l'espace de recherche devient rapidement très grand.

Les résultats obtenues sont très satisfaisant par rapport au paramètre temps et le respect des contraintes de précédence et les contraintes sur machines (contraintes disjonctif), par contre l'optimalité reste toujours une solution approchées.

Comme perspective à ses travaux nous envisageons d'étudier ce type de problème avec une méthode hybride afin d'améliorer le paramètre d'optimalité et de se rapprocher à l'optimum globale.

REFERENCES

[1] Carlier. J, Chretienne. P, "Problèmes d'ordonnancement : modélisation / complexité / algorithmes" collection études et recherche en informatique, Masson ,1988.

[2] Bourdeaud'huy.T, Korbaa.O, "Un modèle mathématique pour la résolution du problème d'ordonnancement cyclique avec minimisation de l'en-cours" 6e Conférence de Modélisation et SIMulation, *MOSIM'06*, Rabat, Maroc, 2006.

[3] Esquirol.P, Lopez.P "L'ordonnancement " Série : Production et techniques quantitatives appliquées à la gestion, Collection Gestion, Economica, 1999.

[4] Chu.C, Proth J. M " L'ordonnancement et ses applications" série : sciences de l'ingénieur, Collection organisation industrielle, Masson, 1996.

[5] Irène Charon, Anne Germa, Olivier Hudry " Méthodes d'optimisation combinatoire " Collection pédagogique de Télécommunication, Masson, 1996.

[6] Mellouli.R, Sadfi.C, Kacem.I, Chu.C, "Ordonnancement sur machine parallèle avec contraintes d'indisponibilité" 6e Conférence de Modélisation et SIMulation, *MOSIM'06*, Rabat, Maroc, 2006.

[7] Hao.JK, Galinier.P, Habib.M, "Méta Heuristique pour l'optimisation combinatoire et l'affectation sous contraintes" Revue d'Intelligence Artificielle, 2000, vol. 13, n°. 2, pp. 283-324.

[8] Goldberg.D.E, "Algorithmes génétiques : exploration, optimisation et apprentissage automatique " Edition Addison Wesley, 1994.

[9] Boutregue.A, Mouali.N, Merhoum.K, " Application d'un algorithme génétique pour la résolution d'un problème d'ordonnancement à une machine " Conférence Internationale sur la Productique, *CIP'0*5 Tlemcen, Algérie, 2005.

[10] Mesghouni.K, Hammadi.S, Borne.P, " Evolution programs for job-shop scheduling " Int. J. Appl. Math. Comput. Sci, *IJAMCS'04*, vol.14, N°.1, pp 91-103, 2004.

[11] Kacem.I, Hammadi.S, Borne.P, " Approche évolutionniste modulaire contrôlée pour le problème du type job-shop flexible" Journées Doctorales d'Automatique, *JDA'01*, pp 25-27, Toulouse, France, 2001.

[12] Merhoum.K, Djeghaba.M, " Méthode Hybride Branch&Bound et Algorithme Génétique appliquée au problème d'ordonnancement Flow-Shop " International Conference on Control, Modelling and Diagnosis, *ICCMD'06*, Annaba, Algérie, 2006.

[13] Benbouzid-Sitayeb.F, Varnier.C, Zerhouni.N, "Proposition of New Genetic Operator for Solving Joint Production and Maintenance Scheduling : Application to the Flow Shop problem" International Conference on Service Systems and Service Management, *ICSSSM'06*, Troyes, France, 2006.

[14] Mellouli.R, Sadfi.C, Kacem.I, Chu.C, "Ordonnancement sur machine parallèle avec contraintes d'indisponibilité" 6e Conférence de Modélisation et SIMulation, *MOSIM'06*, Rabat, Maroc, 2006.