Approche Multi-agents Basée sur le Recuit Simulé pour le Problème d'Ordonnancement Distribué

A.Kouider, B.Bouzouia
Division Robotique et Productique
Centre de Développement des Technologies Avancées CDTA, B.P n° 17
BP n° 17, Baba Hassen, Alger 16303
Email: akouider@cdta.dz, ahmed.kouider@voila.fr

***Résumé*- Dans ce papier nous proposons une approche Multi-agents utilisant la méthode de recuit simulé pour la résolution du problème d'ordonnancement d'atelier de type Job Shop. Ce problème est fortement combinatoire et sa résolution de manière optimale dans un temps raisonnable s'avère impossible, pour de tels problèmes, les méthodes exactes requièrent des calculs qui croient exponentiellement avec la taille de données, ce qui place ce problème dans la classe NP-difficile au sens fort. L'objectif de la résolution est de minimiser la durée totale de l'ordonnancement ou le makespan. Le modèle que nous proposons est constitué de deux classes d'agents: les agents ordonnanceurs responsables de la satisfaction des contraintes sous leurs responsabilité, et l'agent superviseur contenant le noyau de la méthode de recuit simulé.**

I. INTRODUCTION

Un problème d'ordonnancement consiste à allouer dans le temps des jobs à des ressources existantes en quantité limitée tout en satisfaisant un ensemble de contraintes. Parmi les problèmes d'ordonnancement les plus difficiles on cite le problème d'ordonnancement d'atelier connu comme un cas particulier du problème général d'ordonnancement qui préoccupe le domaine de la gestion de production. Dans les problèmes d'ordonnancement d'atelier, il faut comprendre par tâche, opération élémentaire de fabrication (ex: usinage, assemblage, fraisage,…) et par ressources, les machines sur lesquelles ces opérations sont effectuées. Un travail correspond à une gamme d'un produit et une gamme correspond à une liste d'opérations élémentaires nécessaires à la réalisation d'un produit. Une machine de production est considérée comme une ressource renouvelable car une fois utilisée elle redevient disponible, à l'opposé des matières première qui sont des ressources consommables. Suivant la manière dont la gamme d'un produit est organisée, trois types de problèmes d'atelier apparaissent: Flow Shop, Job Shop, Open Shop.

Dans ce papier nous nous intéressons au problème d'atelier de type Job Shop avec l'hypothèse qu'une machine peut présenter plusieurs fonctionnalités, c'est-à-dire que plusieurs types d'opérations peuvent être exécutés sur une même machine.

La résolution de ce problème par des méthodes exactes requiert un immense coût calculatoire, ce coût croit de manière exponentielle avec l'augmentation linéaire de la taille du problème. Parmi les plus connues des méthodes exactes, on trouve les méthodes d'énumération basées sur la programmation en variables mixtes, cette dernière se décompose en deux phases, la première consiste à relaxer les contraintes d'intégralité et résoudre le programme relaxé par le simplexe ou bien par les points intérieurs, dans la deuxième phase, on réintègre les contraintes d'intégralité, et on utilise après, les méthodes de coupe ou les méthodes arborescentes, comme *Branch and Bound* qui fonctionne selon un principe de recherche général de solution dans une perspective d'optimisation combinatoire. Par contre les méthodes approchées s'attachent à obtenir une solution satisfaisante plutôt qu'optimale, ces méthodes doivent vérifier un certain nombre de caractéristique avec un coût calculatoire raisonnable. Parmi les méthodes approchées dédiées à la résolution de ce type de problème, il y a les méthodes par construction progressive, les méthodes par voisinage, les méthodes par décomposition, les méthodes par relaxation des contraintes, et en fin les méthodes liées à l'intelligence artificielle, comme les système multi-agents.

L'approche que nous proposons dans ce papier repose sur un modèle multi-agents basé sur la méthode du recuit simulé.

II. LE PROBLEME D'ORDONNANCEMENT

Le problème Job Shop consiste à réaliser un ensemble de n jobs $\{J_1, \ldots, J_n\}$ sur un ensemble de m ressources $\{R_1, \ldots, R_m\}$. Chaque Job J_i, i=1, …, n, correspond à une gamme. Une gamme de production est définie par une suite de n_i opérations devant être exécutées sur les différentes ressources de l'atelier selon un ordre bien défini. Cet ordre traduit les contraintes de précédence ou de succession entre les différentes opérations du job. D'autre part, chaque opération possède une durée opératoire connue à l'avance et elle ne peut être exécutée que par une et une seule machine, ceci définit l'hypothèse de non préemption, puisque une fois commencé, une tâche ne peut être interrompue, autrement dit, il n' y aura pas de recours à une segmentation de l'exécution d'une tâche. De plus, une machine ne peut exécuter qu'une seule opération à un instant donné, ce qui définit les contraintes disjonctives des ressources. De ces contraintes de disjonction, il résulte qu'un job ne peut être effectué que sur une seule machine à la fois, puisque l'exécution des opérations d'un même job doit être séquentielle et non parallèle. Une solution du problème consiste alors à

définir pour chaque opération une date de début satisfaisant l'ensemble des contraintes citées.

III. RECUIT SIMULE

Le recuit simulé est un algorithme stochastique itératif qui progresse vers l'optimum par échantillonnage d'une fonction objectif, cet algorithme est inspiré d'un processus utilisé en métallurgie. Après avoir subir des déformations au métal, on réchauffe celui-ci à une température, de manière à faire disparaître les tensions internes causées par la déformation, puis on laisse refroidir lentement. L'énergie fournie par le réchauffement lent fige peu à peu le système dans une structure d'énergie minimale.

L'application de ce raisonnement dans les problèmes d'optimisation combinatoire débute par la sélection d'une solution initiale X_n, cette solution peut être soit tirer au sort de l'espace des solutions possibles, soit construite à l'aide des méthodes de construction progressive, on fait correspondre à cette solution une énergie initiale $F(X_n)$ calculée en fonction du critère qu'on cherche à optimiser, on tire au sort ensuite une solution voisine X* de l'ensemble des voisins de la solution courante.

- Si $F(X^*) \leq F(X_n)$ alors X* devient la nouvelle solution courante qu'on notera X_{n+1}.
- Sinon on décide par tirage au sort si X* devient la nouvelle solution courante, ou bien X_n reste une solution courante, et on tirera au sort un autre voisin de X_n.

Initialisation

Choisir une solution initiale $S \in X$;
$S^* \leftarrow S$;
$K \leftarrow 0$; (compteur d'itération global)
Nouveau_Cycle ← vrai;
$T \leftarrow T_0$; (T_0 température initiale du système)

Processus itératif

Tant que (Nouveau_Cycle=vrai) **faire**
Nbiter ← 0 ; Nouveau_Cycle ← faut ;
Tant que (Nbiter <nb_iter_cycle) **faire**
$K \leftarrow K+1$; Nbiter ← Nbiter+1 ;
Générer aléatoirement une solution $S' \in N(S)$;
$\Delta F \leftarrow F(S')-F(S)$;
Si ($\Delta F < 0$) **alors**
$S \leftarrow S'$; Nouveau_Cycle ← vrai;
Sinon
Prob (ΔF, T) ← exp ($-\Delta F/T$);
Générer q uniformément dans l'intervalle [0,1[;
Si (q< prob (ΔF, T)) **alors**
$S \leftarrow S'$;
Nouveau_Cycle ← vrai;
Si (F(S) < F(S*)) alors
$S^* \leftarrow S$;
$T = \alpha T$; ($0 < a < 1$ coefficient de refroidissement)

Algorithme1. Le recuit simulé

Bien que cet algorithme a montré son efficacité dans le domaine de l'optimisation combinatoire, il reste néanmoins difficile à être adapté au Job Shop. Cela est lié au nombre de paramètres à définir:

- Solution initiale,
- Fonction voisinage,
- Evaluation de la solution courante,
- La manière dont la probabilité P dépend de la température T et de ΔF,
- Critère d'arrêt.

Dans ce qui suit nous adaptons les paramètres de cet algorithme au problème de Job Shop avant de décrire après, le modèle multi-agents proposé.

A. Solution initiale

L'efficacité des solutions approchées ayant en procédure une solution initiale dépend du choix de cette dernière, ce choix peut être fait par une construction progressive à l'aide de règles de placement en respectant les contraintes du problème.

B. Fonction voisinage

Le voisinage d'une fonction est obtenu par une permutation de deux opérations consécutives sur le chemin critique et sur une même machine, cette permutation doit garder la satisfaction des contraintes de succession, et par conséquent la permutation concerne uniquement les opérations n'appartenant pas au même job.

C. Détermination des paramètres T_0, L et α

Les paramètres T_0, L et α doivent être déterminer expérimentalement en tenant compte de la taille du problème. T_0 est choisi de sorte qu'au début de l'exécution de l'algorithme, des solutions moins bonnes que la solution courante soient acceptées. On choisit souvent T_0 en fixant un taux d'acceptation initial moyen des solutions plus mauvaises que la solution courante. Par exemple, si ce taux est fixé à 50% on évalue alors par simulation la détérioration moyen (ΔF), on partant d'une solution initiale on calcul T_0 de sorte que $e^{-(\Delta F/T)} = 0{,}5$ c'est-à-dire $T_0 = (\Delta F/\text{Ln } 2)$. La longueur L du palier de température doit être déterminée en tenant compte de la taille de voisinages, elle devrait augmenter avec la taille du problème, ce qui signifie de laisser plus de temps pour explorer l'espace de solution quand ce dernier est plus vaste.

Le paramètre α associé à la décroissance géométrique de la température est fixé aléatoirement à 0,9 ou 0,95, c'est ce paramètre qui occupe de la lenteur du refroidissement sa valeur peut être liée avec la longueur L du palier de température, un refroidissement plus rapide peut être consenti si les paliers sont plus longs.

D. Variation de la température

La température initiale T_0 est maintenue constante pendant L itérations, après ces L itérations on passe à une température $T_1 = \alpha T_0$ ($0<\alpha<1$) que l'on réduit à $T_2 = \alpha^2 T_0$ pour L autres itérations et ainsi de suite, jusqu'à l'itération KL où la température sera $T_k = \alpha^k T_0$.

E. Détermination de la probabilité P

La probabilité P d'accepter X* quand cette solution est moins bonne que la solution courante décrois avec le temps de l'algorithme, cette probabilité P est une fonction P(T, ΔF) dépendant d'un paramètre T, appelé température, par analogie avec le recuit métallurgique, et de la dégradation de l'objectif $\Delta F=F(X^*)-F(X_n)$.

La forme de la loi de probabilité P est dictée par la distribution de Boltzmann $P(T, \Delta F)= e^{-(\Delta F/T)}$ et elle est comprise entre 0 et 1 comme une probabilité.

F. Critère d'arrêt

Le critère d'arrêt est le nombre maximal d'itérations, il peut être dynamique et déterminant le moment où le système est gelé, on dit que le système est gelé si, par exemple, la fonction F a diminuée de moins de 0.1% pendant 100 paliers consécutifs.

IV. LE MODELE MULTI-AGENTS

Le modèle multi-agents proposé est constitué de deux types d'agents cognitifs: agent ordonnanceur et agent superviseur, l'agent ordonnanceur est responsable de la satisfaction des deux classes de contraintes caractérisant le problème du Job Shop, à savoir les contraintes de précédence et les contraintes de disjonction, chaque agent ordonnanceur possède les connaissances nécessaires pour coopérer en vue d'établir le plan de l'ordonnancement qui représente une solution du problème, il possède également son propre comportement qui dépend de son état: satisfait ou insatisfait.

L'agent superviseur est un agent cognitif qui contient le noyau de l'algorithme recuit simulé, et c'est à son niveau qu'on saisit les données du problème, la configuration de l'atelier et la fixation des paramètres de l'algorithme. Cet agent joue le rôle d'interface entre l'utilisateur et les agents ordonnanceurs qui supervise.

Dans la suite nous détaillons ces deux types d'agents.

G. Agent Ordonnanceur

L'architecture de l'agent ordonnanceur est inspirée des deux approches: AOP (Croyance, Compétences, Choix, Engagement) et BDI (Croyance, Décision, Intention), cette architecture se décompose en quatre modules: Connaissances, Communication, Décision, Expertise.

Cet agent est satisfait lorsque les contraintes de succession et de disjonction relatives aux opérations qui lui sont affectées sont satisfaites, sinon il est insatisfait et il essaye de résoudre le problème.

A.1 Module Connaissance

Le module de connaissance détient l'ensemble des informations statiques qui sont composées de la liste des opérations qu'il peut exécuter avec les durées opératoires respectives, les gammes de production, la liste des accointances et l'ensemble de jobs. Alors que ses connaissances dynamiques contiennent les opérations qui lui sont effectivement affectées avec leurs dates de début d'exécution, ces connaissances dynamiques sont traduites par un plan de production

A.2 Module de communication

Le module de communication s'occupe de l'envoie et la réception des messages à destination ou en provenance des autres agents, ce module sert d'interface entre l'agent et ses accointances et gère leurs interactions par le biais d'une boite de réception de messages.

A.3 Module d'Expertise

Le module d'expertise contient l'ensemble des plans comportementaux, chaque plan spécifie une séquence d'action élémentaire tels que l'envoi de messages, mise à jours de la base de connaissances, des calculs d'ordonnancement, etc.

Ce module ne fait que dérouler la séquence d'actions que ces plans spécifient.

Les comportements sont scindés en deux types: plan local qui décrit le comportement isolé d'un agent et ne nécessitant pas d'interaction avec les autres agents, et protocole de rôle qui décrit le comportement d'un agent en interaction avec d'autres agents et qui nécessite l'envoi et la réception des messages

A.4 Module de décision

Ce module active ou désactive les plans comportementaux du module expertise après la vérification de leurs conditions de déclenchement, il mit à jour également les connaissances dynamiques de l'agent (plan d'ordonnancement). Il gère également les événements perturbateurs provenant des autres agents

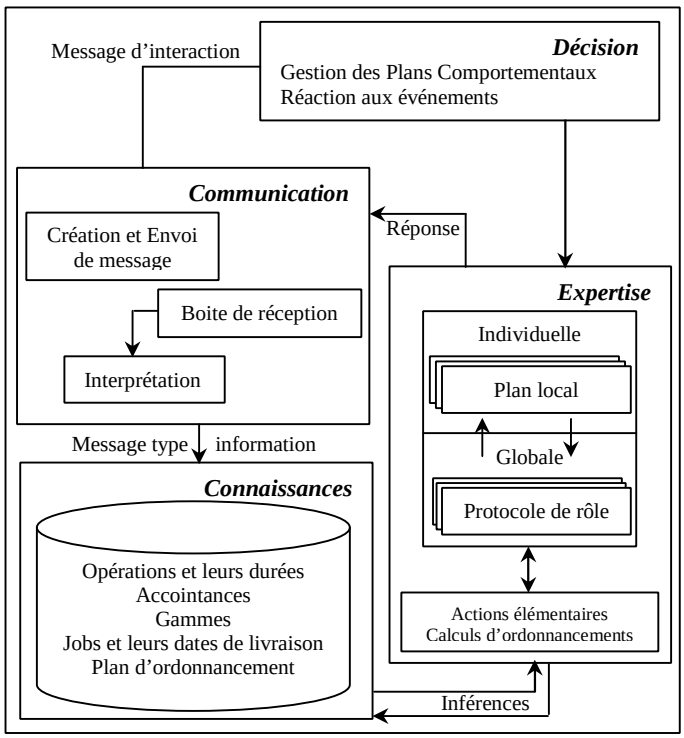

Fig. 1. Architecture de l'Agent Ordonnanceur

H. Agent Superviseur

L'architecture de l'agent superviseur se décompose en quatre modules: Connaissances, Communication, Configuration, Noyau de l'algorithme.

Cet agent contient le noyau de l'algorithme recuit simulé et il est insatisfait tant que le nombre d'itérations n'a pas atteint le nombre maximal.

B.1 Module Connaissance

L'ensemble de connaissance statique de l'agent superviseur contient la liste de toutes les opérations avec les durées opératoires respectives, les gammes de production, la liste des accointances, et la liste des jobs avec leurs gammes respectives. Il contient aussi, les paramètres de l'algorithme recuit simulé: la température initiale, la longueur L du palier de température, le paramètre α associé à la décroissance géométrique de la température. Alors que ses connaissances dynamiques contiennent le compteur d'itération globale, la meilleure solution rencontrée et son coût ainsi que la solution courante et son coût.

B.2 Module Communication

Ce module transmit aux agents ordonnanceurs l'ensemble des informations relatives à la configuration des ressources en précisant leurs compétences et accointances, ainsi que les données du problème d'ordonnancement. Il reçoit également une solution courante du problème et envoi en contrepartie une perturbation de cette solution vers un des agents dans le cas où la solution courante ne satisfait pas le critère d'arrêt.

B.3 Module Configuration

La configuration consiste à déterminer: le nombre d'opérations réalisables dans l'atelier avec leurs durées opératoires, le nombre de ressources et leurs compétences, le nombre de gammes et l'ordre des opérations pour chaque gamme, le nombre de jobs et la gamme associée à chaque jobs, ainsi que les paramètres de l'algorithme recuit simulé.

Ces informations seront stockées dans la base de connaissances de l'agent superviseur, puis transférées vers les agents ordonnanceurs.

B.4 Module Noyau algorithmique

Ce module supervise le processus de résolution qui est précisé dans "Algorithme. 1". Ces fonctionnalités comprend: l'évaluation du coût de la solution courante, le calcul de nombre d'itérations globale, la variation de la température, le calcul de la probabilité P d'accepter X* quand cette solution est moins bonne que la solution courante, et la demande ou proposition d'une perturbation de la solution courante si le critère d'arrêt n'est pas satisfait.

Cette perturbation est obtenue: soit par une génération aléatoire de l'ensemble de solutions voisines, soit par coopération entre les agents ordonnanceurs.

Dans la suite nous présentons la dynamique du système multi-agents.

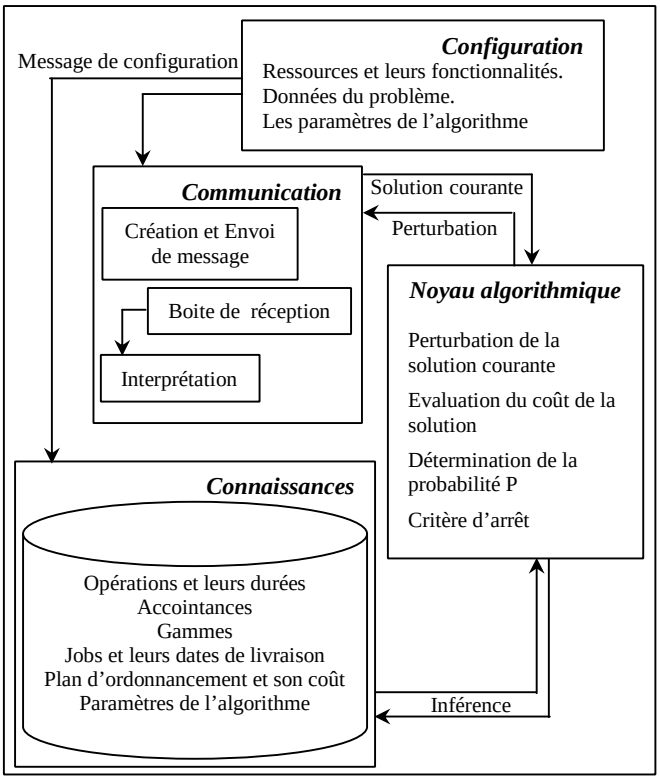

Fig. 2. Architecture de l'Agent Superviseur

V. INTERACTION ET MODE D'INTERACTION

Un agent peut agir sur d'autres agents par une intervention qui peut prendre la forme d'une modification de l'état des agents qu'ils rencontrent, cette modification opère soit au niveau de leurs connaissances (envoie d'information), soit au niveau de leur activité (demande d'engagement). Ainsi cet ensemble des actions réalisables par un agent définit l'interaction.

Le mode d'interaction s'occupe de la gestion des dépendances entre les activités, cette gestion doit définir les actions à opérer conjointement entre les agents qui participent à la réalisation de buts communs.

Dans notre cas, le mode d'interaction des agents ordonnanceurs est décomposé essentiellement en deux phases: la construction de la solution du départ par la méthode de construction progressive, et l'application de l'algorithme de recuit simulé pour la détermination de la solution optimale.

A. Détermination de la solution initiale

Après le transfère des données du problème de l'agent superviseur vers les agents ordonnanceurs, ces derniers coopèrent pour la construction progressive de la solution initiale.

Compte tenu qu'une opération est susceptible d'être exécutée par une seule ressource, ceci signifie que le problème d'affectation n'est plus posé, et chaque agent connaît l'ensemble des opérations que sa ressource doit exécuter.

La coopération des agents ordonnanceurs ne se limite pas uniquement à la détermination de la date de début d'exécution de toutes les opérations du problème avec la satisfaction des contraintes de succession et de disjonction, mais à déterminer des dates permettant également la réduction du coût de la fonction objectif.

A cet effet la coopération débute par le calcul de la date de début au plus tôt pour chaque opération, cette date notée d est calculée par la méthode suivante :

- S'il s'agit de la première opération du job (selon sa gamme opératoire) alors d est égale à la date de début au plus tôt du job (zéro)
- Sinon d est égale à la date de fin de son opération précédente.

D'après cette méthode, on remarque que les contraintes de succession sont satisfaites, reste donc à satisfaire les contraintes de disjonction, pour cela un algorithme coopératif est développé pour remédier le conflit de chevauchement entre les opérations appartenants à une ressource. Cet algorithme réduit les marges libres relatives aux temps libres (chômage inutile des ressources).

La philosophie de cet algorithme débute au niveau de l'agent ordonnanceur ayant la première opération d'un Job, cet agent ordonnanceur affecte la valeur zéro à la date de début de cette opération, puis il avise l'agent ordonnanceur ayant la deuxième opération de ce job, ce dernier avant qu'il affecte l'opération sollicitée par son contractant, il va tenter de remplir ses marges libres par des opérations appartenant à d'autres jobs (sans relaxer les contraintes de succession), puis il envoi un message à l'agent possédant la troisième opération de ce job.

Le processus susmentionné sera continu jusqu'à ce que toutes les dates d'exécutions des opérations de ce job soient déterminées. Et la même idée sera répétée pour tous les jobs du problème.

Etape1 :
Réception du message ('$O_{i,j}$, $C_{i,j}$') ;
Etape2 :
Poser $t_{i+1,j}= C_{ij}$;
Etape3 :
Remplir les marges qui précède $O_{i+1,j}$;
(Permettre un léger décalage à droite s'il le faut)
Etape4 :
Envoyer le message ('$O_{i+1,j}$, $C_{i,j}$') vers $Ag_{i+1,j}$:

Algorithme. 2. Construction de la solution initiale

Sachant que:

- $J_1, J_2, \ldots, J_n$. l'ensemble des jobs.
- $Ag_{i,j}$: l'agent contenant la I-ième opération du Jj .
- $t_{i,j}$: date d'exécution de la I-ième opération du J_j .
- $O_{i,j}$: I-ième opération du J_j .
- $C_{i,j}$: date fin d'exécution de la I-ième opération du J_j .
- $B_{i,j}$: date de fin d'exécution de l'opération qui précède la I-ième opération du J_j .

Dans la procédure de remplissage de marges on note:

- Op: l'opération qui précède la marge à remplir;
- Existe=vrai, signifie qu'il existe des opérations non traitées par la procédure;
- Marge= vrai, signifie qu'il existe un temps libre après l'opération Op
- $d\_O_{i,j}$: date de début d'exécution au plus tôt de l'opération sélectionnée;
- t: date d'exécution de l'opération Op;
- $N\_O_{i,j}$: l'opération non traitée;
- $Ag_{i,j}$: l'agent contenant la I-ième opération du Jj .

Tant que (marge=vrai et existe=vrai) **alors**
Choisir l'opération $N\_O_{i,j}$;
Si (i=1) alors
Placer $N\_O_{i,j}$ dans la marge;
Envoyer le message ('$O_{i+1,j}$, $C_{i,j}$') vers $Ag_{i+1,j}$:

Si ((i > 1) et ($d\_O_{i-1,j}$ est déterminé)) **alors**
Si ($d\_O_{i-1,j}$ < t) alors
Placer $N\_O_{i,j}$ dans la marge;
Envoyer le message ('$O_{i+1,j}$, $C_{i,j}$') vers $Ag_{i+1,j}$:

Algorithme. 3. Construction de la solution initiale

Une fois la phase de construction de la solution initiale est terminée, elle sera envoyée à l'agent superviseur, ce dernier calcul son coût et lance le processus de l'algorithme recuit simulé pour l'améliorer

B. Construction d'une solution voisine

L'agent superviseur ayant reçu la solution initiale calcul son coût et demande aux agents ordonnanceurs de coopérer pour perturber cette solution et obtenir une solution voisine.

Le voisinage d'une solution est obtenu par des permutations particulières, ces permutations concernent uniquement les opérations du même job tout en gardant la satisfaction des contraintes du problème.

La permutation d'opérations consiste à inverser, sur une même machine les dates de début de deux opérations adjacentes appartenants à un chemin critique. Ce chemin possède la longueur égale à la longueur de l'ordonnancement, et est constitué par des opérations reliées soit par un lien de précédence, soit par un lien disjonctif.

C. Exemple d'illustration

Soit le problème de Job Shop illustré par le tableau des durées de traitement des opérations sur leurs différentes machines.

TABLE. 1. SOLUTION INITIALE

	M1	M2	M3
O11	1	X	X
O12	X	1	X
O13	X	X	1
O21	X	1	X
O22	1	X	X
O23	X	X	1
O31	2	X	X
O32	X	2	X
O33	X	X	2

La solution initiale élaborée par le système Multi-agents par la méthode de construction progressive est illustrée dans la figure "Fig. 3"

Nous remarquons :

- L'absence des marges libres sur la machine M1.
- L'existence d'une marge libre sur la machine M2, cette marge qui précède la deuxième opération du J2 est issue de la satisfaction de la contrainte de succession entre les opérations de J2
- Deux marges libres sur la machine M3, la première marge est relative à la contrainte de succession entre les opérations de J2, alors que la deuxième est relative à la contrainte de succession entre les opérations de J_1.

Ceci signifie que ces marges ne peuvent pas être remplis et le coût de la solution de départ est 8.

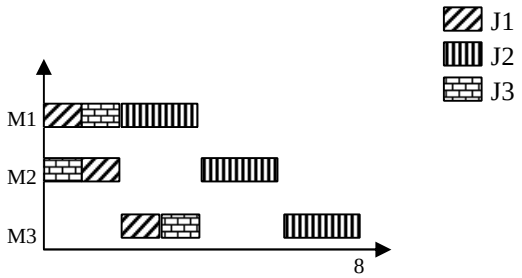

Fig. 3. Solution initiale

Les chemins critiques ayants des opérations adjacents sur une même machine sont les suivants:

- O_{11}, O_{23}, O_{13}, O_{23}, O_{33}.
- O_{13}, O_{21}, O_{13}, O_{23}, O_{33}.
- O_{11}, O_{23}, O_{21}, O_{33}, O_{23}, O_{33}.
- O_{13}, O_{21}, O_{21}, O_{33}, O_{23}, O_{33}.

La permutation des deux dernières opérations du premier chemin critique donne une solution voisine ayant le coût égale à 7 voir "Fig. 4"

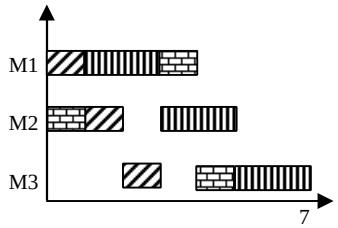

Fig. 4 Perturbation de la solution initiale

Cette nouvelle solution est envoyée vers l'agent superviseur, ce dernier substitue la solution initiale par la nouvelle solution, et le même processus continu tant que le test d'arrêt de l'algorithme n'est pas atteint.

VI. CONCLUSION

Dans le présent article, nous avons présenté une approche Muli-agents pour la résolution du problème de Job Shop.
Cette approche est basée sur un algorithme stochastique inspiré du principe de recuit métallurgique.

Le modèle Multi-agents proposé est composé de deux types d'agents: les agents ordonnanceurs, responsables de la satisfaction des contraintes du problème, et l'agent superviseur intégrant le noyau du recuit simulé.

Le travail présenté sur cet article porte sur la distribution de la méthode du recuit simulé au niveau des différents agents de système, ces agents interagissent en vue de résoudre conjointement le problème d'ordonnancement d'atelier de type Job Shop. La dynamique de fonctionnement entre les agents de système se résume par la construction de la solution du départ par la méthode de construction progressive, et l'application de l'algorithme de recuit simulé pour la détermination de la solution optimale.

REFERENCES

[1] Jacque Teghem, Marc Pirlot. "Optimisation approchée en Recherche Opérationnelle, Recherches locales, réseaux neuronaux et satisfaction de contraintes", paris hermès science publication, 2002, p 25 à 38.
[2] Pierre Lopez, François Roubellat "Ordonnancement de la production", paris hermès science publication, 2001, p 56 à 93.
[3] Patrick Pujo, Jean-paul Kieffer "Fondement de pilotage des systèmes de production", paris hermès science publication, 2002, p 94 à 106.
[4] Jean-pierre Briot, Yves Demazeau "Principes et architecture des systèmes multi-agents", paris hermès science publication, 2001, p 27 à 60.
[5] Patrick Pujo, Jean-paul Kieffer "Méthode de pilotage des systèmes de production", paris hermès science publication, 2002, p 61 à 95.
[6] Earwan Trouvez "IAD et ordonnancement : une approche coopérative du Réordonnancement par systèmes Multi-agents" 2002 ,01AIX30023.
[7] Gotha, "Les Problèmes d'ordonnancement, in Recherche Opérationnelle / Operations Research" Vol 27 n°1, 1, 1993, p 77 à 150.
[8] S Ourari, A Kouider, S Benabbas, "An heuristic approach for scheduling a one machine with random events" 10th IEEE MMAR Miedzyzdroje, pologn