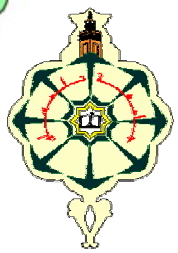

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid
Faculté des sciences de l'ingénieur
Département d'automatique

Mémoire de magister en « Automatique, Productique et Informatique »
Option « Productique et Informatique »
Intitulé :

Métaheuristiques pour la manipulation de routages alternatifs en temps réel dans un Job Shop

Présenté par :

Mehdi Souier

Devant le Jury :

Président :			
Lyes Benyoucef	Chargé de recherche	INRIA	France
Examinateurs :			
Khaled Belkadi	Maître de conférence	USTO	Algérie
Latéfa Ghomri	Maître assistante	UABB	Algérie
Encadreur :			
Zaki Sari	Maître de conférence	UABB	Algérie
Co-encadreur :			
Ahmed Hassam	Maître assistant	UABB	Algérie

Table des matières

Remerciements

Je tiens tout d'abord à remercier monsieur Zaki Sari, maître de conférence à l'université de Tlemcen et le chef de l'équipe productique au laboratoire d'automatique de Tlemcen, pour m'avoir accueilli au sein de son équipe. Je le remercie profondément pour la confiance et l'honneur qu'il m'a accordé en acceptant de diriger ce mémoire. Son attention, sa bienveillance et son appui sans faille ont été des encouragements décisifs pour mener à terme ce travail. Il a été toujours présent pour commenter et corriger tous mes documents et articles. Sans ses qualités rares tant au niveau humain que scientifique le développement et l'achèvement de ce travail n'auraient été possibles. Je lui exprime ma vive et respectueuse gratitude.

Je voudrais également remercier monsieur Ahmed Hassam, maître assistant à l'université de Tlemcen, d'avoir accepter d'être mon co-encadrant de mémoire, pour les conseilles qu'il m'a donné pour commencer ce travail, pour sa rigueur du travail, sa disponibilité, pour avoir collaborer à mes travaux. Il m'a fait partager sa motivation, et également le plaisir de cette collaboration. J'ai eu beaucoup de plaisir à travailler avec lui.

Ce mémoire ne se serait pas passer dans d'aussi bonnes conditions sans les autres membres de l'équipe de productique. J'adresse mes remerciements à eux pour leur patience et leur bonne humeur, qui ont su faire régner dans le laboratoire une atmosphère studieuse et chaleureuse.

Je tiens également à remercier les personnes qui ont accepté de participer au jury de mon mémoire :

- Le président de ce jury monsieur Lyes Benyoucef, maître de conférence à l'université Paul Verlaine, chargé de recherche au laboratoire INRIA Lorraine, à Metz et monsieur Khaled Belkadi, maître de conférence à l'université des sciences et de la technologie d'Oran pour l'intérêt qu'ils ont porté à mon travail et pour le temps qu'ils ont consacré en acceptant d'être rapporteurs malgré les contraintes de distance, je suis très honoré par leur présence dans ce jury.
- Un très grand merci à mademoiselle Latéfa Ghomri, maître assistante à l'université de Tlemcen qui a accepté d'être rapporteur et de participer au jury. Je la remercie pour tous ses conseils, son aide et sa modestie.

Malgré que tous les mots restent faibles pour exprimer mes sentiments, qu'ils trouvent à travers ce travail les fruits et la récompense de leurs efforts. Je remercie mes parents pour leur soutien moral, spirituel et leur tolérance durant toutes mes années d'études. J'espère que le dieu me donne la force et le courage pour que je puisse rendre leurs sacrifices.

Mes remerciements vont aussi aux mes proches et membres de ma famille en particulier mes deux sœurs, dont les encouragements et le soutien ont été indispensables à l'aboutissement de mes études.

Je voudrais remercier tout ceux qui ont apporté une contribution dans la réalisation de ce travail et m'ont aidé dans mes études, et que je n'ai pas pu citer.

Je voudrais saluer toutes celles et tous ceux qui me tiennent à cœur et qui ont quitté cette vie, paix à leur âme.

Enfin, je remercie Dieu tout puissant de la patience et de la volonté qui m'a donné pour réaliser ce travail.

Introduction générale

Aujourd'hui, les entreprises doivent faire face à l'augmentation de la concurrence, à la pression de plus en plus forte de leur environnement (clients et concurrents) pour un renouvellement rapide des produits, mais aussi une augmentation de l'offre vis à vis de la demande. Les systèmes de production et de fabrication actuels offrent une grande flexibilité, tel que les systèmes flexibles de production (FMS) qui fournissent des avantages divers comme l'augmentation de l'utilisation des ressources, l'augmentation de la productivité, la réduction des encours…etc.

Afin de tirer pleinement partie de la flexibilité offerte par ces systèmes, les décisions d'allocation et d'ordonnancement des opérations et des plans de processus sont généralement prises dynamiquement et à très court terme, en fonction de l'état du système de production (disponibilité des ressources, disponibilité du système de manutention, présence de goulots d'étranglement), des caractéristiques du plan de production (date échues des ordres de fabrication) et des objectifs de production (augmentation du taux de production, réduire les en-cours).

L'ordonnancement dynamique des systèmes de production se fait en deux étapes: La première consiste à ordonnancer l'exécution des tâches dans le temps en donnant une priorité (qui peut être dynamique) à ces différentes tâches. Cette première étape se fait suivant des règles dites règles de séquencement.

La gestion des files d'attente par des règles de priorité constitue une des approches les plus simples et les plus utilisées pour ordonnancer en temps réel les opérations à traiter. Les règles de priorité ont suscité de nombreux travaux de recherche depuis plus de vingt ans, et plus particulièrement pour tenter de proposer des solutions au problème d'ordonnancement temps réel. Ces règles sont définies par Blackstone et al **[Blackstone 82]** comme "des règles utilisées pour sélectionner parmi tous les ordres de fabrication en attente de traitement, le prochain ordre à traiter".

La deuxième étape consiste à repartir l'exécution des tâches sur les ressources disponibles, en affectant à chaque job un routage parmi ses routages alternatifs possibles, cette deuxième phase se fait suivant des règles dites règles de routage. Parmi les règles de routages existantes dans la littérature on trouve : EPL (Equal Probability loading), elle consiste à sélectionner une machine au hasard ; FA (First Available), dont le principe est de sélectionner la première machine à être libérée, SQ (Shortest Queue) dans cette règle on sélectionne la machine dont la longueur de la file d'attente est la plus petite; LWQ (Least Work in Queue), cette règle est une amélioration de la règle SQ, qui consiste à donner la priorité à la machine dont la file d'attente est la moins chargée non pas en terme de nombre de pièces mais en terme de temps opératoire, DMM (Dissimilarity Maximisation

Method), cette méthode a pour principe de maximiser les dissimilitudes entre les routages adoptés par les différentes pièces présentes simultanément dans le job shop.

Les problèmes d'ordonnancement dans les systèmes de production sont généralement de type NP-Complet. Ces problèmes sont donc extrêmement difficiles et il n'existe pas de méthodes universelles permettant de résoudre efficacement tous les cas.

Les métaheuristiques sont des algorithmes de type stochastique visant à résoudre une large gamme de problème d'optimisation difficile, pour lesquels on ne connaît pas de méthodes classiques plus efficace. Souvent inspirées d'analogies avec la réalité comme la physique (recuit simulé, diffusion simulée...) la biologie (les algorithmes évolutionnaires, la recherche tabou...) et l'éthologie (les colonies de fourmis, les essaims particulaires...). Elles sont généralement conçues au départ pour résoudre des problèmes discrets, mais peuvent s'adapter aux autres types de problèmes.

Dans notre travail nous nous intéressons à certaines métaheuristiques (les colonies de fourmis, les algorithmes génétiques, le recuit simulé, la recherche tabou, les essaims particulaires, électromagnétisme métaheuristique). Pour cela nous avons proposé des algorithmes sur la base de ces techniques pour la résolution du problème de sélection de routages alternatifs en temps réel et nous allons présenter une étude comparative entre ces métaheuristiques et les règles DMM, et DMM modifiée afin d'avoir une idée sur l'efficacité de ces techniques et de sélectionner la meilleure pour ce problème.

Pour valider les résultats de notre étude, nous avons simulé les méthodes DMM et DMM modifiée dans un FMS par le logiciel de simulation ARENA, et les algorithmes des métaheuristiques étudiées ont été programmées en Java et exécutées dans un Core (TM) 2 Duo CPU avec 2.2 GHZ et 1 GO de RAM.

Le plan de ce travail s'articule autour de quatre chapitres. Les deux premiers représentent des généralités sur les FMS, les problèmes d'ordonnancement et les métaheuristiques, ils n'apportent rien de nouveau, mais ils sont utiles à la compréhension du sujet pour les non spécialistes.

Le premier chapitre contient quatre parties principales la première est consacrée aux généralités concernant les systèmes flexibles de production, la deuxième décrit les problèmes d'ordonnancement de manière générale et traite la problématique d'ordonnancement de type job shop et la troisième présente l'ordonnancement en temps réel, la notion d'ordonnancement dans un FMS est ensuite développée en faisant un état de l'art qui regroupe les travaux les plus importants dans ce domaine. Des travaux concernant l'ordonnancement interactif, coopératif, les différentes techniques d'intelligence artificielle, et les problèmes de sélection de routages alternatifs ont été présentés. Malheureusement aucun de ces articles ne concerne l'utilisation de métaheuristiques pour résoudre les problèmes de sélection de routages dans un job shop en temps réel.

Dans le deuxième chapitre nous nous intéressons de manière approfondie aux métaheuristiques. Il contient les éléments nécessaires à la compréhension de ces approches. Il est composé de deux parties principales, la première traite de manière générale ces techniques, leurs définitions, leurs domaines d'utilisation, leurs caractéristiques et leur classification. La seconde partie et réservé à la présentation détaillée de chaque métaheuristique étudiée, en incluant sa définition, son origine et son algorithme de base.

Les deux derniers chapitres représentent notre contribution proprement dite, deux cas de sélection de routages alternatifs en temps réel sont considérés : avec et sans ré-ordonnancement des pièces contenues dans la station de chargement. Dans le troisième chapitre, nous présentons le FMS étudié pour tester nos algorithmes et l'adaptation de ces techniques pour résoudre le problème de sélection de routages alternatifs en temps réel sans ré-ordonnancement des pièces contenues dans la station de chargement ainsi les résultats obtenus après plusieurs simulations avec et sans présence de pannes, leurs interprétations et la comparaison de ces techniques avec les règles DMM et DMM modifiée.

Le dernier chapitre est réservé à la résolution du problème en temps réel avec ré-ordonnancement des pièces contenues dans la station de chargement.

Enfin, on termine par une conclusion générale qui fera la synthèse de notre travail en ouvrant de nouvelles perspectives.

Chapitre 1

Les problèmes d'ordonnancement dans les systèmes flexibles de production

1.1. Introduction

Le système de production rassemble l'ensemble des moyens qui permettent la transformation de ressources en produits ou en services **[Christian 98]**

On peut classer les systèmes de production en deux types, La première typologie sépare les systèmes fonctionnant à la commande de ceux basés sur une production pour stock, la deuxième classification est centrée sur la nature et le volume des produits fabriqués (les systèmes à production continue, les systèmes à production discrète).

Les systèmes de production peuvent être classé suivant le déplacement du produit :

- selon le type de produit (product layout) où on peut faire une très grande quantité de produits mais moins de variété.
- ou selon le traitement (process layout), ces types de systèmes consistent à regrouper les machines de traitement identique ce qui nous permet de produire de grandes variétés de produits mais en petite quantité.

L'automatisation flexible est ainsi largement due à l'incorporation de la flexibilité propre à l'informatique dans les schémas de pilotage des systèmes de production. Cette évolution peut être décomposée en trois étapes majeures **[Hatchuel 92]** :

- Dans un premier temps l'apparition de l'informatique industrielle a permis la création de processus automatiques plus complexes et plus souples que ceux supportés antérieurement (automatismes mécaniques et automates programmables de la première génération). L'application de ces processus aux systèmes de production a conduit à l'apparition des machines outils à commande numérique (MOCN) qui permettent d'approcher la productivité et la flexibilité de l'opérateur humain.
- Dans un second temps, ces mêmes processus automatiques ont permis la réalisation de systèmes robotisés de plus en plus performants. Les MOCN ont alors pu être regroupées en cellules (petits groupes de machines regroupés autour d'un mécanisme robotisé et orienté vers la fabrication d'une gamme de pièces) puis pour des productions en plus grandes séries, en lignes flexibles. Cette structure permet en effet de conserver la cadence propre aux lignes

de transfert (en limitant l'appel aux procédures complexes d'ordonnancement) tout en produisant, grâce aux MOCN, des pièces différentes au sein d'une même famille.

- Enfin, dans un dernier temps, l'introduction directe de l'ordinateur dans la gestion du cycle de production a favorisé l'utilisation des techniques avancées de l'informatique (système d'information, intelligence artificielle...). Celles-ci ont alors apporté à l'automatisation flexible les processus de décision qui lui manquaient pour décentraliser au maximum les processus de décision et se libérer des structures linéaires ou parallèles caractéristiques des lignes ou des cellules flexibles.

L'atelier flexible constitue ainsi un compromis efficace entre la flexibilité des machines servies manuellement et la productivité des lignes de transfert.

Dans un système flexible de production, le plan de charge permet de vérifier si la charge occasionnée dans l'atelier par les commandes, n'est pas supérieure à la capacité des ressources de l'atelier. Dans le cas contraire, des réajustements de charge ou de capacité (heures ou équipes supplémentaires) peuvent être faits. Les données sont transmises à la fonction d'ordonnancement.

La fonction ordonnancement des entreprises manufacturières devant permettre à la fois l'optimisation de l'utilisation des ressources et la réactivité de l'atelier, elle est un des points clés de la rentabilité et est ainsi elle-même en pleine évolution.

Ce chapitre est consacré aux problèmes d'ordonnancement dans les systèmes flexibles de production. La prochaine section de ce chapitre vise à décrire les systèmes flexibles de production. Pour cela, nous rappelons la définition de flexibilité, nous étudions ensuite les éléments des systèmes flexibles de production et leurs différents types.

L'ordonnancement de type job shop objet principal de notre travail, est décrit dans la troisième section de ce chapitre où nous présentons dans un premier temps l'ordonnancement, ses objectifs et ses contraintes, et dans un second temps un bref aperçu des principales méthodes, directes ou itératives, permettant de résoudre le problème d'ordonnancement global et la problématique d'ordonnancement de type job shop.

La quatrième section de ce chapitre est consacrée à l'ordonnancement en temps réel où nous présentons la problématique et diverses approches d'ordonnancement temps réel.

Les autres approches sont traitées dans la dernière section de ce chapitre, où un état de l'art détaillé sur l'ordonnancement dans un FMS sera présenté.

1.2. Les systèmes flexibles de production

Dans le domaine de l'industrie où la concurrence est féroce, les grands unités de production deviennent de moins en moins compétitives, parce que très peu adaptables aux variations du marché. Ces unités disparaissent progressivement laissant place à de nouvelles structures industrielles aux concepts différents : les systèmes flexibles de production (Flexibles Manufacturing Systems « FMS »). Ces systèmes sont conçus pour produire en petites et moyennes quantités, à des coûts minimums, une variété de produits avec des temps de préparation des machines et de

changement d'outils minimums. Leurs structures leur permettent de produire une très large gamme de produits et donc d'être moins sujet à un crash économique.

Un système de production flexible a pour but d'aboutir, non seulement, à une productivité importante, mais aussi à une grande flexibilité de production lui permettant de suivre les variations du marché.

Dans cette section, nous nous proposons dans un premier temps de faire un rappel sur la définition de la flexibilité, puis dans un second temps nous nous intéressons aux éléments des systèmes flexibles de production et leurs différents types.

1.2.1. La flexibilité

La flexibilité est la notion principale utilisée dans la conception des systèmes automatisés modernes de fabrication comme les systèmes flexibles de production.

On peut définir la flexibilité par l'ensemble des propriétés et qualités d'un système manufacturier qui peut supporter des changements dans les types de produits et la capacité de production. Ces changements peuvent être :

- Internes : comme les pannes des équipements, pannes des systèmes informatiques (logiciels de gestion), absentéisme des travailleurs, variations des temps de fabrication…etc.
- Externes : comme le changement dans la conception des produits, la complexité de conception des produits, la variation de la demande ...

Pour absorber les incertitudes causées par les changements de conception des produits, le système de fabrication doit être flexible et capable de fabriquer différents types de produits avec un temps de fabrication et un coût minimum. Donc on peut dire que la flexibilité est la capacité des systèmes de fabrication de répondre à la fois aux changements internes et externes. **[Adamou 97]**

1.2.2. Les éléments des systèmes flexibles de production

Un système flexible de production (flexible manufacturing system) est constitué d'un ensemble de machines à commande numérique, de stations de travail connectées par un système de transport automatisé, le tout commandé par ordinateur.

Les systèmes flexibles de production peuvent être caractérisés par les points suivants :

- Machines reprogrammables à commande numérique.
- Changement d'outils automatisé
- Système de transport automatisé
- Chargement et déchargement automatisé
- Contrôle coordonné.

Les systèmes flexibles de production nécessitent des machines complexes et chères, mais leur nombre est réduit par rapport à d'autres systèmes de production. Globalement une économie dans le coût et l'espace de l'ordre de 30% peut être réalisée **[Askin 93]**.

Les systèmes flexibles de production sont constitués d'un certain nombre d'équipements qui peuvent être divisés en quatre grandes familles :

- Les stations de travail : chaque station de travail est composée d'une machine à commande numérique, d'un magasin d'outils avec un système automatique de changement d'outils, d'un magasin à palettes et supports, d'un système automatique de chargement / déchargement des palettes dans la machine, d'un système automatique d'inspection et d'un système de contrôle.
- Les systèmes de manutention et de transport, comprenant les systèmes de transport, les robots manipulateurs…, servent à déplacer les produits et éventuellement les outils entre les machines suivant les chemins planifiés pendant la conception du FMS. Il existe différents moyens de transport et différentes stratégies. On peut par exemple transporter les produits d'une aire de stockage vers une station de travail, puis vers une aire de stockage, ou par contre faire les déplacements entre les stations de travail, ce qui diminue le nombre de déplacements. On peut aussi déplacer les produits par petits lots au lieu que ce soit par unité. Le choix de l'une de ces méthodes dépend du système global. On favoriserait le déplacement à l'unité lorsque les distances entre les stations sont petites. Par contre, pour les grandes distances, le déplacement par lots sera plus intéressant. De plus, la taille et le poids des produits comparés au système de transport peuvent limiter le choix d'une méthode ou d'une autre.
- Les systèmes de stockages qui renferment les aires de stockages, les systèmes de chargement déchargement…, peuvent en plus du stockage, servir à d'autres activités, comme l'emballage, le contrôle de qualité...
- Les systèmes de contrôle et de communication : chaque système de contrôle peut être composé d'un ordinateur central de commande, il donne des instructions et reçoit des états des situations de chaque équipement du FMS. Il peut garder en mémoire les gammes d'usinage des produits si les ordinateurs subalternes sont saturés. C'est le système de contrôle qui décide quand et/ ou comment les produits doivent se déplacer entre les différentes machines et être chargés ou déchargés. **[Sari 03]**.

1.2.3. Types de systèmes flexibles de production

Le développement des FMS a permis d'en installer un nombre important dans différents types d'environnements industriels. Cependant, une confusion perdure dans le domaine de classification et de la définition des différents types de FMS.

Widmer **[Widmer 91]**, pense que la classification des systèmes flexibles de production peut se faire, en se basant sur le nombre de machines à commande numérique (CN) et leur agencement, de la manière suivante :

- Le module flexible (MF) : est une machine à commande numérique avec une aire de stockage, un changeur de pièces et un changeur automatique d'outils.
- La cellule flexible (CF) : représente plusieurs modules reliés par un véhicule filoguidé permettant l'alimentation des machines en pièces.
- Le groupe flexible (GF) : est un ensemble de cellules et de modules formant la même zone de production (fabrication, usinage, ou assemblage) joints par des véhicules filoguidés, le tout est géré par un ordinateur central.

- Le système flexible (SF) : représente plusieurs cellules flexibles reliées entre elles par des véhicules filoguidés composant les divers zones de production.
- La ligne flexible (LF) : est un ensemble d'instruments attribué aux diverses machines comme une ligne de véhicules filoguidés, de robots, de convoyeurs, de navettes…

Askin et Standridge **[Askin 93]**, quant à eux présentent une classification, en 5 niveaux, basée principalement sur le nombre de machines à CN :

- L'équipement (equipement) : représente une machine à CN, un robot manipulateur…
- La station de travail (workstation) : représente une machine à CN avec son aire de stockage, son système de chargement/déchargement, son changeur d'outils…
- La cellule flexible (cell), représente une ou deux stations de travail.
- L'atelier flexible (shop), représente un ensemble de cellules reliées par un système de transport automatique.
- L'unité industrielle (facility) : est l'ensemble de plusieurs ateliers.

Ils pensent que dans la plupart des cas un système flexible de production représente, en fait, un atelier flexible.

Maccarthy et Lui **[Maccarthy 93]** dressent une classification pouvant être appliquée à un grand nombre de systèmes flexibles de production. A partir de la littérature, ils formulent les remarques suivantes :

- Le terme FMS est général, il ne peut donc pas faire partie d'une classification, mais plutôt il définit toute structure de production se disant flexible.
- Un certain nombre de divergences réside dans la plupart des systèmes de classifications. Par exemple dans le système présenté par **[Widmer 91]**, il n'existe pas des relation de classification entre les lignes flexibles et les autres structures, de plus le terme système flexible qui est très général est considéré comme un niveau de classification.

Dans la classification de Maccarthy et Lui, le terme système flexible de production (FMS) représente n'importe quelle structure flexible automatisée. La classification se fait suivant les caractéristiques de fonctionnement et de contrôle des FMS. Elle consiste en quatre structures inter-reliées :

- La machine flexible unique (Single flexible machine, SFM) : est une machine à commande numérique (CN) avec changeur d'outils, système de transport et aire de stockage.
- La cellule flexible de production (flexible manufacturing cell : FMC) : est constituée d'un groupe de SFM reliées par un système unique de transport.
- Le système flexible de production à plusieurs machines (multi-machine flexible manufacturing systems : MMFMS) est constitué de plusieurs SFM connectées par un système de transport capable de desservir plusieurs machines à la fois (transport multiple).

- Le système flexible de production à plusieurs cellules (multi-cell flexible manufacturing system, MCFMS) : est constitué de plusieurs FMC et éventuellement de SFM connecté par un système de transport automatique.

Aussi, ils définissent un système de production flexible (FMS) comme étant un système capable de produire une grande variété de produits. Il est constitué d'un ensemble de machines à commande numérique, connectées par un système de transport automatisé, le tout commandé par ordinateur **[Sari 03]**.

1.3. L'ordonnancement

Dans la gestion de production, l'ordonnancement de la production a pour rôle de produire les quantités (lots) fixés dans le plan de production (plan directeur) dans les délais. Ces lots, appelés jobs dans l'atelier (ou Ordres de Fabrication), doivent traverser celui-ci en subissant diverses opérations dans un ordre fixe ou non, afin d'obtenir des produits finis **[Portmann 87]**, **[Carlier 88]**. Une opération (ou tâche) correspond au traitement d'un job par une ressource.

1.3.1. Définition

Ordonnancer la fabrication d'un ensemble de produits, c'est en répartir la réalisation dans le temps et l'espace **[Doumeingts 83]**. Pour cela, il importe :

1. d'une part, de déterminer la séquence des opérations dans le temps,
2. d'autre part, de déterminer la répartition des ressources de production (à capacité limitée) aux différentes opérations prévues.

Ces deux phases doivent être réalisées dans le but de minimiser une fonction objective, en respectant des contraintes.

1.3.2. Contraintes sur l'ordonnancement

On peut distinguer trois grandes catégories de contraintes : temporelles, technologiques et de ressources. Le premier type concerne les délais de fabrication imposés. Le deuxième type correspond aux contraintes technologiques, en général décrites dans les gammes de fabrication des produits. On y trouve des contraintes d'enchaînement temporel, mais aussi l'obligation d'utilisation de certaines ressources. Le dernier type de contraintes concerne la limitation de la quantité de ressources de chaque type. Ces contraintes distinguent les différents types de ressources, qui peuvent être disjonctives (exécuter une seule tâche à la fois), ou cumulatives (exécuter plusieurs tâches en parallèle). Les ressources n'ont pas la même disponibilité et la même capacité, celle-ci pouvant être modulée par la modification des calendriers d'utilisation ou l'emploi de ressources externes. Une ressource peut aussi être consommable, lorsqu' après sa libération, elle n'est pas disponible en même quantité. Dans le cas contraire, elle est dite renouvelable.

On peut aussi distinguer les contraintes suivant qu'elles soient strictes ou pas. Les contraintes strictes sont des exigences à respecter alors que les contraintes dites «relâchables» (contraintes de préférences) peuvent éventuellement ne pas être satisfaites**[Letouzey 01]**.

1.3.3. Objectifs de l'ordonnancement

Le traitement de l'ordonnancement dans la littérature s'est tout d'abord orienté vers une optimisation monocritère. L'environnement manufacturier évoluant rapidement et la concurrence devenant de plus en plus acharnée, les objectifs des entreprises se sont diversifiés et le processus d'ordonnancement est devenu de plus en plus multicritère. Les critères que doit satisfaire un ordonnancement sont variés. D'une manière générale, on distingue plusieurs classes d'objectifs concernant un ordonnancement **[Esquirol 99]**.

- **Les objectifs liés au temps :** On trouve par exemple la minimisation du temps total d'exécution, du temps moyen d'achèvement, des durées totales de réglage ou des retards par rapport aux dates de livraison.
- **Les objectifs liés aux ressources :** maximiser la charge d'une ressource ou minimiser le nombre de ressources nécessaires pour réaliser un ensemble de tâches sont des objectifs de ce type.
- **Les objectifs liés au coût :** ces objectifs sont généralement de minimiser les coûts de lancement, de production, de stockage, de transport, etc.

1.3.4. Les différentes approches de résolution

Selon Letouzey **[Letouzey 01]**, deux types de méthodes sont utilisées pour résoudre les problèmes d'ordonnancement : les méthodes directes, qui sont des approches résolvant le problème en une seule passe, et les méthodes itératives qui permettant de produire une solution puis de l'améliorer par itérations successives.

1.3.4.1. Aperçu sur les approches directes

Ce paragraphe présente quelques méthodes de résolution d'ordonnancement parmi les plus courantes et les plus connues. Comme pour toute activité de décision, on distingue en ordonnancement deux approches fondamentales pour la résolution de problème :

- Une approche dite « optimale » consistant à rechercher une décision optimale par rapport à un critère donné, à partir d'hypothèses fortement réductrices qui seules permettent d'établir et de prouver cette optimalité.
- Une approche orientée « rationalité limitée » ne cherchant pas à atteindre l'optimalité de la solution et préconise de poser un ensemble d'hypothèses moins réductrices qui vont favoriser la recherche d'une décision satisfaisante **[Simon 77].**

1.3.4.1.1. Méthodes de résolution optimales

Ces méthodes considèrent le problème d'ordonnancement comme un problème d'optimisation où l'on cherche à satisfaire une fonction objective généralement unique. Elles sont décrites dans de nombreux ouvrages traitant des problèmes d'ordonnancement, comme par exemple **[Esquirol 99]**. Les méthodes optimales permettent d'avoir des solutions optimisant un seul et unique critère, sous des conditions bien particulières. Ces conditions sont rarement satisfaites dans les problèmes industriels.

Les problèmes d'ordonnancement les plus courants sont en général des problèmes NP complets **[Maccarthy 93]** et le temps de résolution par des méthodes optimales augmente de façon exponentielle avec la taille du problème.

1.3.4.1.2. Méthodes de résolution non optimales

1.3.4.1.2.1. Placement des ordres de fabrication

Dans un premier temps, les OF sont classés suivant un critère quelconque (priorité, date de livraison, date de commande...). Une fois cette liste établie, toutes les opérations du premier OF de la liste sont planifiées sur les ressources quand celles-ci sont libres, puis toutes les opérations du deuxième OF... Une fois planifiées, ces tâches ne sont plus remises en cause.

Les opérations d'un OF peuvent être placées sur les ressources de deux manières : au plus tôt (les tâches sont planifiées de la première à la dernière dès que possible) ou au plus tard (les tâches sont planifiées en partant de la dernière date de livraison et remontant jusqu'à la première).

Des techniques « mixtes » sont aussi possibles. C'est une méthode simple, rapide et facile à comprendre. Peu consommatrice en temps de calcul, elle a été utilisée par les premiers logiciels d'ordonnancement. Néanmoins, ne permettant aucune optimisation, même partielle ou locale, elle n'est quasiment plus utilisée dans les logiciels d'ordonnancement.

1.3.4.1.2.2. Théorie des graphes

Le principe de cette technique d'ordonnancement repose sur l'utilisation d'un graphe dont les sommets représentent les tâches et dont les arcs modélisent les contraintes entre opérations. Au départ, seules les contraintes liées aux gammes sont représentées. A chaque étape, l'ensemble des opérations est séparé entre les tâches déjà ordonnancées, celles qui peuvent l'être et celles ni ordonnancées ni ordonnançables. On choisit une opération parmi celles ordonnançables ainsi que la ressource sur laquelle elle est affectée. A chaque conflit rencontré, on utilise une règle de résolution prédéfinie. Chaque résolution de conflit entraîne l'apparition de nouvelles contraintes sur le graphe.

1.3.4.1.2.3. Approches par satisfaction des contraintes

Le principe général de ces approches consiste à réduire l'espace dans lequel on recherche la solution en exploitant les contraintes que doit satisfaire cette solution, et à rechercher cette solution dans l'ensemble restreint des solutions admissibles obtenues.

Parmi les travaux sur l'utilisation de la satisfaction de contraintes en ordonnancement, on trouve par exemple ceux présentés dans **[Erschler 76]**, ou plus récemment ceux relatés dans **[Schwalb 97]**, sur la propagation de contraintes temporelles. Lopez et al **[Lopez 92]** proposent des méthodes d'ordonnancement qui combinent les contraintes d'utilisation des ressources et les contraintes temporelles…

1.3.4.1.2.4. Méthodes issues de l'intelligence artificielle

L'intelligence artificielle a fourni de nouvelles approches et tendances en ordonnancement depuis deux décennies, notamment par l'utilisation de réseaux de neurones **[Sabuncuoglu 98]** ou de systèmes experts **[Smith 92]**.

L'utilisation de la logique floue dans l'ordonnancement n'a pas pour but de proposer une nouvelle méthode, mais d'apporter ponctuellement des techniques qui vont permettre d'être plus proche de la réalité. L'incertitude et l'imprécision caractérisant les données d'une entreprise, bien que fréquemment constatées, sont rarement prises en compte à court terme **[Smith 92]**. La gestion de cette incertitude et de cette imprécision est plus souvent faite par le biais de méthodes statistiques, mais l'utilisation de la logique floue et de la théorie de possibilités a connu un succès certain.

L'intelligence artificielle est susceptible d'être d'un grand intérêt pour l'ordonnancement, lui permettant notamment d'être plus proche de la réalité.

Les méthodes d'ordonnancement issues de l'intelligence artificielle sont toutefois peu implantées dans l'industrie car elles ne sont pas encore suffisamment matures ni suffisamment transparentes pour l'utilisateur.

1.3.4.2. Approches itératives

Il a été fréquemment constaté dans la littérature (**[Farhoodi 90]**, **[Bérard 98]**, **[Grabot 99]**) qu'un ordonnancement est rarement idéal « du premier coup » et qu'il est généralement nécessaire de procéder à des améliorations successives. On distingue deux approches d'amélioration d'un ordonnancement :

- Réparation de la solution par un processus automatique (amélioration par ordinateur).
- Intervention de l'opérateur dans l'amélioration de la solution.

Dans ce paragraphe, nous nous s'intéressons aux approches de réparation automatique (l'utilisation de l'intelligence artificielle, les métaheuristiques et les techniques de choix de la meilleure méthode de résolution), les approches d'intervention humaine seront traitées dans l'état de l'art.

1.3.4.2.1. L'intelligence artificielle

L'intelligence artificielle est fréquemment utilisée pour améliorer des ordonnancements déjà existants. Dans **[Bel 88]** et **[Bensana 88]**, les auteurs proposent de faire un ordonnancement par satisfaction de contraintes, puis d'améliorer le résultat grâce à un système expert. On trouve dans **[Artiba 97]** la présentation d'un système d'ordonnancement de systèmes complexes comportant un module « base de connaissances » qui en fonction des problèmes et des performances de l'ordonnancement en cours, re-modélise le problème et choisit la méthode ou l'heuristique de résolution la mieux adaptée au modèle. Partant d'un ordonnancement fait par gestion des files d'attente, il est proposé dans **[Lereno 01]** de capitaliser les résultats successifs de l'ordonnancement pour constituer une base de connaissances qui pourra choisir le réglage des meilleurs paramètres.

1.3.4.2.2. Métaheuristiques

Les méthodes dites « métaheuristiques » consistent à explorer « intelligemment » l'espace des solutions possibles pour trouver l'optimum ou une solution proche. Parmi les métaheuristiques les plus connues on trouve :

- Les colonies de fourmis.

- Les algorithmes génétiques
- Le recuit simulé
- La recherche tabou…

1.3.4.2.3. Choix de la meilleure méthode

On peut aussi classer dans cette catégorie les différentes approches qui proposent de sélectionner automatiquement la meilleure méthode de résolution en fonction des paramètres de l'atelier considéré. On trouve différentes approches de sélection comme les réseaux de neurones **[Boukachour 97]** ou les systèmes experts **[Romanowicz 97]**. Dans **[Riane 99]** les auteurs proposent une plate-forme qui évalue et compare les différentes heuristiques de résolution. On peut aussi rappeler **[Artiba 97]** dans lequel le système présenté comporte un module « base de connaissances » qui choisit la méthode ou l'heuristique de résolution la plus adaptée au modèle. Des tentatives de sélection des meilleures règles de priorité en fonction des paramètres de l'atelier ont été faites dans l'ordonnancement de job shop **[Boucon 91]**.

1.3.5. Problématique de l'ordonnancement en Job Shop

Suivant le type d'atelier, la nature des problèmes rencontrés peut accentuer la complexité de l'ordonnancement; par exemple, la prise en compte des temps de préparation, ou la possibilité de préempter une ressource pour l'affecter à un lot urgent.

On distingue trois grands types d'atelier, avec de nombreux types intermédiaires **[Graves 81]** :

- le *Flow Shop* dans lequel la séquence des opérations est toujours la même pour tous les produits de l'atelier
- l'*Open Shop* dans lequel chaque produit doit effectuer un nombre fixé d'opérations dans n'importe quel ordre.
- le *Job Shop* dans lequel la séquence des opérations peut varier d'un produit à l'autre, les séquences (ou gammes) étant définies par avance pour chaque produit.

Dans ce travail, nous nous intéressons exclusivement à l'ordonnancement Job Shop.

Dans un problème d'ordonnancement Job Shop n×m, n jobs doivent être effectués par m machines, sachant que chaque machine ne peut travailler que sur un seul job à la fois. Les notations couramment employées dans les problèmes d'ordonnancement sont les suivantes :

- la date de début au plus tôt du job J_i est notée r_i (release date). Cette date correspond à un minorant de la date de disponibilité du job J_i.
- d_i et C_i sont respectivement la date de fin au plus tard souhaitée (due date) et la date effective de fin de fabrication (completion date) du job J_i.

Le problème d'ordonnancement peut se décomposer en deux phases, une première phase qui tente d'attribuer des caractéristiques temporelles aux opérations à effectuer et une deuxième phase d'attribution des ressources aux différentes opérations.

- La première phase est décrite par des contraintes potentielles qui regroupent les contraintes de précédence entre tâches, prenant en compte la succession des opérations de la gamme

opératoire (la tâche *i* doit démarrer après la tâche *j*) et les contraintes d'une tâche dans le temps **[Carlier 88].**

- La deuxième phase est décrite par les contraintes disjonctives qui expriment le fait que deux opérations utilisant une même ressource, ne peuvent avoir d'intervalle de temps commun **[Carlier 88].**

Un ordonnancement qui satisfaisait aux contraintes potentielles et disjonctives est dit admissible **[Mebarki 95]**.

1.4. L'ordonnancement en temps réel

1.4.1. Problématique

La nécessité de gérer au mieux, c'est à dire en temps réel s'impose notamment dans les FMSs. Un FMS est un système de production fortement intégré et piloté par ordinateur, qui permet une production en petite et moyenne série. Ce type de systèmes est plus particulièrement implanté pour pouvoir produire une large variété de produits **[Froment 88]**. Ainsi, selon Talavage et Hannam **[Talavage 88]**, ses principaux avantages sont les suivants :

- réduction des temps de cycle, ce qui permet de réduire les en-cours et de mieux s'adapter aux besoins du marché.
- amélioration de la qualité.
- meilleure utilisation des équipements.

Malheureusement, les FMSs nécessitent des investissements élevés. Il importe donc de gérer au mieux de tels équipements. De plus, l'accroissement des variétés de produits susceptibles d'être traités dans un FMS impose des outils de gestion flexibles et performants. C'est pourquoi la définition d'un ordonnancement prévisionnel limite les capacités d'un tel système et ne permet pas de répondre aux attentes de ses responsables. Il faut donc proposer un système de gestion souple et capable de piloter un système aussi complexe qu'un atelier flexible. Le rôle de ce système est de fournir une réponse à la question suivante :

Chaque fois qu'une ressource devient libre, à quelle opération doit-on l'affecter, compte tenu de l'état présent de l'atelier et des objectifs de production ?

Pour pouvoir résoudre ce problème, il apparaît nécessaire de :

- s'adapter aux changements d'états de l'atelier, c'est à dire prendre en compte tous les changements d'état qui affectent le plan de production prévu (aléas ou changements d'état attendus).
- surveiller le déroulement de l'exécution des opérations, afin de pouvoir détecter les situations anormales, c'est à dire les situations où il apparaît une dérive entre les réalisations de l'atelier et les objectifs de production.

L'ordonnancement temps réel est une des réponses les mieux adaptées aux problèmes de gestion rencontrés dans les systèmes flexibles de production. On définit l'ordonnancement temps réel comme une démarche permettant, chaque fois qu'une décision d'affectation doit être prise, de

proposer une solution prenant en compte l'état réel du système de manière à optimiser une fonction objectif **[Mebarki 95]**.

1.4.2. Ensemble d'ordonnancements admissibles

Parmi les approches d'ordonnancement temps réel, on peut citer celle qui a été mise en place par le Laboratoire d'Analyse et d'Automatique des Systèmes de Toulouse, à partir de méthodes issues de l'analyse sous contraintes dont est issu le logiciel ORDO **[Roubellat 94]**. L'idée est de rechercher, pour exécuter un plan de production, non pas un seul ordonnancement, mais un ensemble d'ordonnancements. Pour cela, le processus de décision est décomposé en deux étapes :

A. Analyse : à l'aide d'un modèle d'atelier, prenant en compte les contraintes essentielles du problème, le but est de caractériser un ensemble d'ordonnancements admissibles, c'est à dire compatibles avec les contraintes du modèle. Pour cela, deux démarches sont possibles :

- la première démarche **[Erschler 76]**, **[Esquirol 87]** consiste à rechercher les conditions nécessaires d'admissibilité **[Zouhri 93]**, telles que tout ordonnancement solution doit satisfaire ces conditions. Cette méthode utilisant l'analyse sous contraintes, permet de caractériser un ensemble d'ordonnancements dans lequel est inclus l'ensemble des ordonnancements admissibles.
- la deuxième démarche **[Demmou 77]**, **[Thomas 80]**, **[Le Gall 89]** consiste à rechercher des conditions suffisantes d'admissibilité, telles que tout ordonnancement satisfaisant ces conditions est admissible. Cette méthode permet de caractériser un ensemble d'ordonnancement inclus dans l'ensemble des ordonnancements admissibles.

B. Aide au pilotage : la séquence de groupes admissibles générée dans l'étape précédente sert de base pour le pilotage en temps réel. Les informations fournies par le suivi de production permettent d'une part de surveiller la faisabilité au cours du temps de l'ensemble d'ordonnancements proposé, et d'autre part, de proposer au décideur, à chaque prise de décision, un choix d'actions possibles, compte tenu de l'état de l'atelier. Pour cela, les différents événements pouvant survenir dans un atelier (événements attendus et aléas) sont analysés et les différents types de décision à prendre sont étudiés.

Tout ceci constitue un véritable système interactif d'aide à la décision **[Le Gall 89]** et a été implémenté dans le logiciel ORABAID, coeur du progiciel d'ordonnancement et de pilotage temps réel d'atelier ORDO **[Roubellat 94] [Mebarki 95]**.

1.4.3. La gestion des files d'attente

Une autre approche permettant de gérer en temps réel la flexibilité des ateliers, est basée sur la gestion des files d'attente par règles de priorité. Cette approche utilise des règles de priorité qui sont des approches directes non optimales permettant de déterminer chaque fois qu'une ressource se libère, parmi toutes les opérations en attente de cette ressource, la prochaine opération à traiter. Le choix est réalisé en fonction de paramètres concernant les opérations (durée opératoire), les produits (date de fin au plus tard souhaitée), ou les ressources (taux d'utilisation) **[Mebarki 95]**.

1.4.4. Présentation de la règle DMM [Saygin 01]

Dans ce paragraphe nous présentons la règle DMM (Dissimilarity maximization method) en proposant son algorithme où nous allons montrer les différentes étapes à suivre pour intégrer la règle DMM comme outil de sélection de routages alternatifs en temps réel.

Nous montrons dans ce qui suit, la manière d'appliquer la règle DMM dans un système flexible de production afin de sélectionner un routage parmi les routages disponibles pour chaque type de pièce. Les pièces qui arrivent en premier ont la plus grande priorité suivant la règle FIFO (First in First out), les autres pièces restent en attente dans les files d'attentes d'entrée ou de sortie des différentes machines ou dans la station de chargement. L'algorithme de la règle DMM est donné comme suit:

Etape 1 : Tous les routages sont libres (disponible) donc $X(i) = 0$.

Où $X(i)$ représente le coefficient de disponibilité du routage i, il prend la valeur 0 ou 1.

Etape 2 : Calcul des coefficients de dissimilitude D_{ij} comme suit :

$$D_{ij} = \frac{\text{Nombre de types de machines non communes entre les routages i et j}}{\text{Le nombre total des machines entre les routages i et j}}$$

Etape 3 : Arrivée des pièces.

Etape 4 : Selon le type de la pièce tester s'il y'a au moins un routage libre et au moins une place libre dans la file d'attente de la station de chargement.

Etape 5 : Si la condition précédente n'est pas vérifiée, la pièce va dans une file d'attente jusqu'à ce que la condition soit vérifiée.

Etape 6 : Si la condition de l'étape 4 est vérifiée alors on calcule la somme :

$$S(j) = \sum_{i=1}^{q} X(i) * D(i, j)$$

Où $D(i, j)$ Coefficient de dissimilitude.

Etape 7 : Trouver le maximum de $S(j)$.

Etape 8 : Le routage j correspondant à la valeur de $S(j)$ trouvée dans l'étape précédente est occupé donc : $X(j) = 1$ (Le routage j contient une seule pièce à la fois).

Etape 9 : Traitement de la pièce selon le routage sélectionné j.

Etape 10 : A la fin du traitement, le routage devient de nouveau disponible $X(j) = 0$.

Etape 11 : Sortie de la pièce du système.

Remarque : Ce cycle va se répéter de l'étape 3 à l'étape 11 à chaque arrivé d'une pièce, jusqu'au un critère d'arrêt est atteint.

1.4.5. Présentation de la règle DMM modifiée [Hassam 07]

Dans cette section nous allons expliquer la règle DMM modifiée qui a été développée à partir de la règle DMM déjà citée précédemment, Cette règle est aussi utilisée dans la sélection de routages alternatifs en temps réel dans un FMS.

Dans la règle DMM, après avoir sélectionné un routage pour une pièce, ce routage ne peut pas être utilisé par une autre pièce tant que la première pièce n'est pas sortie du système donc chaque routage ne peut contenir qu'une seule pièce à la fois. La modification de cette règle vise à garder le même principe qui dépend de la maximisation des coefficients de dissimilitude pour la sélection de différents routages alternatifs mais en affectant plusieurs pièces à un seul routage. Alors si tous les routages sont sélectionnés par des pièces, la pièce suivante va être acheminée dans le routage où la file d'attente de la première machine de ce routage, contient au moins une place libre.

Dans ce paragraphe nous montrons l'intégration de la règle DMM modifiée, dans un système flexible de production pour la sélection d'un routage alternatif parmi les routages disponibles pour chaque type de pièce. Les pièces arrivant en premier ont une priorité plus élevée suivant la règle FIFO (First In First Out) les autres pièces restent en attente dans les files d'attentes d'entrée ou de sortie des différentes machines ou dans la station de chargement. La règle DMM modifiée va utiliser l'algorithme suivant pour la sélection de routages alternatifs en temps réel dans un système flexible de production.

Etape 1 : Tous les routages sont libres (disponible) donc $X(i) = 0$.

Où $X(i)$ représente le coefficient de disponibilité du routage i, il prend la valeur 0 ou 1.

Etape 2 : Calcul des coefficients de dissimilitude D_{ij} comme suit :

$$D_{ij} = \frac{\text{Nombre de types de machines non communes entre les routages i et j}}{\text{Le nombre total des machines entre les routages i et j}}$$

Etape 3 : Arrivée des pièces.

Etape 4 : Selon le type de la pièce tester :

- S'il y'a au moins un routage libre et au moins une place libre dans la file d'attente de la station de chargement. Ou,
- Si tous les routages sont occupés et la file d'attente d'entrée des premières machines de ces routages contient au moins une place libre et ces machines ne sont pas en panne.

Etape 5 : Si la condition précédente n'est pas vérifiée, la pièce va dans une file d'attente jusqu'à ce que la condition soit vérifiée.

Etape 6 : Si la condition de l'étape 4 est vérifiée alors on va calculer la somme :

$$S(j) = \sum_{i=1}^{q} X(i) * D(i, j)$$

Où $D(i, j)$ Coefficient de dissimilitude.

Etape 7 : Tester, si on a trouvé un maximum de $S(j)$ (Il y'a de routages libres).

Etape 8 : Si la condition précédente est vérifiée alors, aller à l'étape 10.

Etape 9 : Si la condition de l'étape 7 n'est pas vérifiée, alors sélectionner le routage, où la file d'attente d'entrée de sa première machine contient au moins une place libre.

Etape 10 : Le routage *j* sélectionné d'après l'étape 7 ou l'étape 9 est donc occupé, $X(j) = 1$.

Etape 11 : Traitement de la pièce selon le routage sélectionné *j*.

Etape 12 : A la fin du traitement, le routage devient de nouveau disponible $X(j) = 0$.

Etape 13 : Sortie de la pièce du système.

Remarque : Ce cycle va se répéter de l'étape 3 à l'étape 11 à chaque arrivée d'une pièce, jusqu'au un critère d'arrêt est atteint.

1.5. Etat de l'art

Beaucoup de travaux qui concernent l'ordonnancement d'un FMS ont été proposés pendant les trois dernières décennies depuis le début des années 80 où les ateliers flexibles ont commencé à gagner l'acceptation par les pays industrialisés et il continue à attirer les intérêts des secteurs académiques et industriels.

Bien que plusieurs approches analytiques, telles que la programmation mathématique, aient été utilisées pour résoudre des problèmes de commande de FMS, leur incapacité de manipuler beaucoup de dispositifs réalistes et dynamiques d' FMS a toujours été un obstacle pour les applications industrielles. En outre, ces modèles ont besoin d'une certaine durée de calcul pour trouver une solution, les rendant peu convenables pour des applications de commande en temps réel.

Dans les environnements de production dynamique (les ateliers flexibles), avec des ressources limités, des pannes des machines aléatoires ou des critères de production multiples, Il est fréquemment reconnu que la résolution d'un problème d'ordonnancement est fortement dépendante de l'atelier auquel l'ordonnancement s'applique. Cet aspect explique l'échec relatif de toutes les solutions d'amélioration automatique qui ne permet pas de s'adapter aux spécificités de chaque entreprise.

D'après Ammons et al **[Ammons 88]** la commande d'un FMS exige une interaction complexe de deux composantes :

1- Les ordinateurs pour l'exécution de la commande automatique et les activités de routages

2- Les opérateurs humains pour diriger l'automation, surveiller l'écoulement et le rendement du système, intervenir dans les opérations inattendues du système et compenser l'effet des événements imprévus.

Plusieurs chercheurs comme Dunkler et al **[Dunkler 88]** ont étudié la commande et la surveillance d'un FMS et ont prouvé que la supervision humaine peut améliorer les dates dues et les performances d'inventaire d'un FMS. D'autres comme Baek et Yoon **[Baek 99]** ont notamment constaté que globalement, les choix de l'opérateur humain donnent de meilleurs résultats que les choix de l'ordinateur. La recherche s'est donc tournée vers des solutions permettant d'intégrer la

participation de l'utilisateur du logiciel d'ordonnancement, afin que celui-ci puisse guider la résolution en fonction de ses besoins particuliers, On distingue en général deux approches de l'intervention humaine dans l'ordonnancement :

- la coopération : (la mise au point conjoint de l'ordonnancement par le logiciel et par le gestionnaire d'atelier) **[Mori 90], [Patkai 98]** et **[Esquirol 97].** Esquirol et al **[Esquirol 97]** ont proposé pour résoudre les problèmes d'ordonnancement de fabrication d'avions une plateforme homme machine coopérative ou les ordinateurs vérifient l'uniformité des milliers de contraintes et les intervenants humains sélectionnent des choix appropriés.
- Et l'interaction (alternance de décisions de l'opérateur et du logiciel) **[Breskvar 03]**, **[Smed 00]**, **[Bistline 98]** et **[Farhoodi 90].** Smed et al **[Smed 00]** ont décrit la structure et le comportement d'un système d'ordonnancement interactif qui permet l'interaction entre l'utilisateur et les algorithmes d'optimisation pour l'assemblage de circuits électroniques où l'utilisateur peut composer une heuristique et l'ordonnancement peut être manipulé directement à l'aide d'une interface d'utilisation graphique. Breskvar et Kljajic **[Breskvar 03]** ont présenté un système d'ordonnancement de production utilisant les algorithmes génétiques et un modèle visuel de simulation à événements discrets, l'avantage principal du système présenté est la décomposition du problème. Dans ce cas le planificateur a des connaissances basées sur l'expérience et l'ordinateur emploie pratiquement ces connaissances qui représentent le principal moyen de réduire l'espace de recherche des solutions. L'ordinateur emploie ses capacités pour les calculs rapides et pour la recherche de la solution globale on utilise les algorithmes génétiques. Mais d'après Saygin et Kilic **[Saygin 99]** les logiciels existants sont en général trop lents et ne peuvent pas réagir aux conditions de changements de l'atelier, ils sont basés sur des formulations simples qui ignorent des contraintes importantes de la réalité, ils sont basé sur des fonctions objectives simples, difficile à installer et à intégrer dans des systèmes commerciaux préexistants…

L'intelligence artificielle cherche à construire des systèmes de plus en plus autonomes, des algorithmes capables de résoudre des problèmes d'une certaine classe. Mais, cette fois, la machine simule l'intelligence, elle semble agir comme si elle était intelligente. D'après Chan, F.T. S. et Chan, H. K. **[Chan 04]** les approches de l'IA domineront dans les futures études sur la résolution des problèmes d'ordonnancement par rapport aux approches de résolution classiques (méthodes analytiques…). Plusieurs travaux **[O'kane 00]**, **[Bilkay 04]**, **[Akyol 07]**, **[Erkmen 97]**, **[Cakar 05]**, **[Trappey 07]**, **[Restrepo 08]**, **[Li 05]**… citent des techniques liées à l'intelligence artificielle et les méthodes issues de l'analyse de données afin d'acquérir des connaissances sur un système et de structurer ces connaissances.

O'kane **[O'kane 00]** a proposé une approche pour l'obtention d'un ordonnancement réactif d'un FMS, plusieurs concepts tel que l'apprentissage et les systèmes experts apprenants ont été mise en œuvre, cette approche est basée sur la prise de décisions intelligentes à partir des données sur l'état de l'atelier et l'augmentation de la base de connaissances permettant l'apprentissage basé sur les expériences antérieurs. Bilkay et al **[Bilkay 04]** ont proposé un algorithme basé sur la logique floue pour donner des priorités aux pièces groupées en lots dans un FMS. Dans cette approche la décision est prise en fonction de la taille du lot, les dates dues, et du temps de traitement total. L'algorithme proposé est utilisé avant l'allocation de la machine à l'opération pour assigner

les priorités aux différents types d'opérations. Cet algorithme proposé peut régénérer l'ordonnancement en cas de panne d'une machine sans incrémentation du makespan, il combine la priorité du type de pièce, la division de la taille en lots, la compensation des pannes des machines et autres changements de l'état de l'atelier et peut être utilisé en ligne et hors ligne en cas de besoin mais uniquement dans les ateliers de moyennes et petites tailles. Akyol et Araz **[Akyol 07]** ont développé un système interactif d'aide à la décision basé sur la combinaison entre les réseaux de neurones et la simulation pour la sélection des règles de priorités en temps réel afin d'obtenir des mesures de performances désirés données par un utilisateur. Erkmen et al **[Erkmen 97]** ont développé un Ordonnanceur floue accordé génétiquement (Genetically Tuned Fuzzy Scheduler (GTFS)) pour un FMS hétérogène sous incertitude, le système d'ordonnancement prend ses entrées à partir d'une table et crée un ordonnancement maître optimal. Le GTFS utilise la base des règles floues et d'inférences où les ensembles flous sont produits par un algorithme génétique pour accorder l'optimisation, l'optimisation floue sur le critère du temps dans la date limite et le besoin de machine, tenant compte la disponibilité de machine, de l'uniformité, du temps de processus, ainsi que de la capacité choisi. Cakar et Yildirim **[Cakar 05]** ont proposé un système interactif d'aide à la décision neuro-génétique couplé avec la simulation pour concevoir un système de type job shop en réalisant des valeurs prédéterminés de mesures de performances tel que le temps d'écoulement, le nombre de jobs en retards, le retard total, l'utilisation de chaque centre de travail, quand le système de fabrication a été conçus la gestion doit prendre des décisions sur la disponibilité des ressources, ou de la capacité, dans cet arrangement, le nombre de machines identiques dans chaque poste de travail et les règles de priorités (quatre règles ont été employé : SPT, EDD, CR, FCFS) ont été utilisé pour réaliser les mesures de performances désirés.

Les métaheuristiques forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile, pour lesquels on ne connaît pas de méthode classique plus efficace. Elles sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé. Les problèmes d'ordonnancement dans les systèmes de production sont généralement de type NP hard, à cause de leur complexité et l'absence de méthodes de résolution optimal, plusieurs ont pensés à utiliser ces techniques pour les résoudre et surmonter ces obstacles. Les algorithmes génétiques sont le type le plus connu et le plus utilisé des algorithmes évolutionnaires, À cause du succès rencontré par cette technique dans la résolution de beaucoup de problèmes plusieurs chercheurs l'ont utilisé pour résoudre les problèmes d'ordonnancement de type job shop. Kim et al **[Kim 08]** qui s'intéressent à l'ordonnancement de type job shop de l'industrie des cylindres hydrauliques ont fait une étude comparative sur la résolution du problème entre les règles de priorités, et les algorithmes génétique, et il ont conclu que l'ordonnancement le plus efficace pour ce type de problèmes est obtenus en combinant les deux techniques, Park et Choi **[Park 06]** ont proposé un algorithme génétique pour l'intégration du processus de planification et l'ordonnancement dans un job shop, les performances de cette approche ont été évaluées en comparant le programme d'intégration avec celui de séparation, ils ont constatés que cette approche améliore clairement les performances concernant le makespan. Guo et al **[Guo 08]** ont étudié le problème d'une ligne d'assemblage flexible, son modèle mathématique a été présenté avec les objectifs de réduire la somme pondéré des pénalités de retards et d'avances des opérations et l'équilibrage de l'écoulement de production, un algorithme génétique a été développé pour le

résoudre, les résultats obtenus ont montré que le modèle d'optimisation proposé peut résoudre le problème efficacement. Zhang et al **[Zhang 08a]** ont appliqué un algorithme génétique hybride amélioré par une recherche locale, l'approche proposée a été examinée dans des instances standards et comparée avec d'autres approches, les meilleurs résultats ont été obtenus par le modèle proposé.

D'autres chercheurs ont utilisés d'autres métaheuristiques : Ponnambalam et al **[Ponnambalam 00]** ont considéré la technique de recherche tabou pour minimiser le makespan dans un job shop. Zhang et al **[Zhang 07]** ont adapté cette métaheuristique avec une nouvelle structure de voisinage prenant en considération les chemins critiques. Vilcot et Billaut **[Vilcot 08]** ont combiné la recherche tabou avec les algorithmes génétiques pour minimiser le makespan, Scrich et al **[Scrich 04]** ont élaboré deux heuristiques (l'heuristique hiérarchique et multi débuts) à base de recherche tabou pour minimiser le retard total, la première était efficace pour les instances à petite valeur de retard, et l'autre est bonne pour les grandes valeurs de retards et avec moins de temps de calcul. Aydin et Fogarty **[Aydin 04]**, ont proposé un algorithme modulaire parallèle pour le fonctionnement d'un système multi agents basé sur le recuit simulé pour résoudre le problème d'ordonnancement dans un job shop.

Les informaticiens et les ingénieurs ont pu transformer des modèles du comportement collectif des insectes sociaux en méthodes utiles pour l'optimisation et le contrôle. Un nouveau domaine de recherche a vu le jour qui a pour objet de transformer la connaissance que les éthologistes ont des capacités collectives de résolution de problèmes des insectes sociaux en techniques artificielles de résolution de problèmes : c'est l'intelligence en essaim, Parmi les techniques de l'intelligence en essaim largement utilisés pour résoudre les problèmes de type job shop on trouve les colonies de fourmis et les essaims de particules. Boryczka **[Boryczka 04]** a développé un algorithme sur la base des colonies de fourmis, l'efficacité de cet algorithme a été testée en le combinant avec quelques règles de priorités et le comparant avec d'autres métaheuristiques, cet algorithme a donné les meilleurs résultats concernant le Makespan. Kang et al **[Kang 07]** ont proposé un système multi agents pour l'ordonnancement dynamique d'un FMS où les programmes d'ordonnancement sont générés à la base des colonies de fourmis. Kılıç et Kahraman **[Kılıç 06]** ont pensé à la combinaison des colonies de fourmis avec la logique floue et ont développé un algorithme efficace par rapport aux autres métaheuristiques pour la résolution du problème considéré. Ge et al **[Ge 06]**, ont proposé un algorithme à base d'essaims de particules pour résoudre ce type de problème, dans cet algorithme un nouveau concept de distance et de vitesse a été proposé. Xia et Wu **[Xia 06]** ont développé un nouvel algorithme en combinant les essaims de particules avec le recuit simulé pour minimiser le makespane. Lei **[Lei 08]** a adapté cette métaheuristique pour la résolution des problèmes multi objectifs (minimisation du makespan et du retard total des jobs). Sha et Hsu **[Sha 06]** ont proposé un algorithme hybride doté d'opérateurs d'échange, et liste taboue. Niu et al [**Niu 08]** ont combiné cette technique avec les opérateurs génétiques pour résoudre l'ordonnancement dans un job shop avec un temps de traitement floue. Beaucoup d'autres travaux existant dans la littérature qui utilisent les métaheuristiques pour résoudre les problèmes d'ordonnancement dans un job shop. **[Zhang 08b]**, **[Reddy 06]**, **[Ripon 07]**, **[Tsai 08]**, **[Ombuki 04]**, **[Jerald 06]**, **[Huang 08]**, **[Montgomery 06]**, **[Lian 06]**, **[Ge 07]**, **[Watson 05]**, **[Sammarra 07]**, **[Saidi-Mehrabad 07]**, **[Schuster 06]**, **[Liaw 08]**, **[Tao 07]**.

Dans l'intelligence artificielle, l'utilisation d'une architecture multi-agent permet de prendre des décisions d'une manière décentralisée, cette approche semble être bien adaptée aux problèmes complexes, la méthode multi-agent permet de résoudre des sous-problèmes localement avec agent, et de proposer une solution globale en raison des interactions entre les agents. Plusieurs chercheurs ont proposés des systèmes basés sur cette architecture : Kouiss et al **[Kouiss 97]** Aydin et Öztemel **[Aydin 00]**, Gao et al **[Gao 05]**, Zattar et al **[Zattar 07]**, Kouider et Bouzouia **[Kouider 07]**. Kouiss et al **[Kouiss 97]** ont développé un système d'ordonnancement à base d'une architecture multi agents, chaque agent est consacré à un centre de travail, il choisi localement et dynamiquement la règle de priorité la plus approprié, selon des considérations locales et globales un nouveau choix est effectué chaque fois qu'un événement prédéfinis est apparu, la méthode de choix est améliorée par l'optimisation des seuils numériques utilisés dans la détection des symptômes. Aydin et Öztemel **[Aydin 00]**, ont proposé un système d'ordonnancement dynamique qui est composé d'un environnement simulé et un agent intelligent choisi la règle de priorité la plus approprié selon les conditions de l'environnement qui applique la règle sélectionné par l'agent, les connaissances d'agent sont amélioré à l'aide d'un algorithme d'apprentissage par renforcement. Gao et al **[Gao 05]** ont proposé un mécanisme de coopération pour le contrôle et l'ordonnancement d'un atelier basé sur le comportement collectif des fourmis, dans cette approche chaque agent prend la charge d'une pièce et choisi une ressource sur la base des phéromones stockées dans l'environnement de l'information et met à jour ces phéromones pour guider le routage des agents suivants. Kouider et Bouzouia **[Kouider 07]** ont proposé une approche multi agents utilisant le recuit simulé pour la résolution du problème d'ordonnancement d'atelier de type Job Shop, l'objectif de cette approche est la minimisation du makespan.

Les travaux cités précédemment et d'autres travaux existants dans la littérature **[Tang 93]**, **[Li 05]**, **[Dominic 04]**, **[Karaouzene 07,08]** qui s'intéresse aux règles de priorités, ne considèrent pas la flexibilité de routages, cette flexibilité est l'offre au système les moyens d'un aiguillage plus souple, de façon à servir les différents segments de procédés libres ou sous engagés. De nombreux travaux se sont intéressés à l'influence de la flexibilité de routage sur les performances d'un FMS avec et sans pannes des machines, Tsubone and Horikawa **[Tsubone 99]** ont étudiée par simulation l'impact de la flexibilité des machines et de routages sur le temps d'écoulement moyen des pièces sous différents conditions d'atelier, Mahmoodi et Mosier **[Mahmoodi 99]** ont étudiée l'effet des règles d'ordonnancement et de flexibilité sur les performances d'un FMS aléatoire (temps moyen d'écoulement, pourcentage moyen de taches en retard et retard moyen). ElMekkawy et ElMaraghy **[ElMekkawy 03]** ont développé un algorithme de réordonnancement en temps réel en évitant les impasses, dans leur modèle, les routages alternatifs, la disponibilité du système de manutention et la limitation des capacités de files sont prises en compte.

Caprihan et al **[Caprihan 97]** ont démontré l'effet de variations 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 défini dans **[Caprihan 04]** les différents modes d'occurrences des retards d'informations (retard des machines ou des pièces), et dans **[Caprihan 05]** ils ont étudié l'effet du retard de l'information (avec ces différents modes possibles) sur les performances d'un FMS, ils ont proposé dans **[Caprihan 06]** une stratégie d'ordonnancement prenant en considération la flexibilité de routages basé sur la logique floue dans un FMS où les retards d'informations sont manifeste, cette approche est basé sur deux variables d'entrées RWINQ (relative work-in-next-

queue) et RNINQ (relative number-in-next-queue) des files d'attentes et a donné de meilleurs performances concernant le retard moyen, le pourcentage de jobs en retard, le moyenne du temps d'écoulement, mais ils ont ignoré certains critères importants comme le taux d'utilisation des machines et du système de manutention. Buyurgan et Saygin **[Buyurgan 08]** ont présenté une plateforme qui utilise le processus d'hiérarchie analytique (analytical hierarchy process : AHP) pour l'ordonnancement temps réel et le routage des pièces. Cette approche proposée pour la prise de décisions multicritères, inclus les machines à états finis et un modèle d'ordonnancement qui emploie AHP pour évaluer à partir de l'état actuel les futurs états possibles dans un horizon limité de vision en comparant les mesures de performances de chaque état (performances en relation avec les machines ou les pièces). Dans cette approche le modèle de commande est développé en considérant que les poids relatifs des mesures de performances sont constants et indépendants des alternatives futures du système, AHP permet une comparaison des mesures de performances par pairs en incluant l'évaluation des critères et des états futures, et tous ces états sont considérés jusqu'à un établissement d'ordonnancement et la prise des décisions du routage, mais le système étudié n'était pas vraiment saturé (les pièces viennent avec une moyenne de 15 min) et ils ont ignoré certaines objectives comme le taux de production et le taux d'utilisation du système de manutention. Ce travail est une perspective des travaux de Buyurgan et Mendoza **[Buyurgan 06a]** où les états sont rangés sur la base des mesures des performances et l'état qui a le rang le plus élevé est choisi, cette méthode a l'inconvénient de capturer les valeurs relatives des mesures de performances et d'éliminer trop d'états à l'avance. Pour surmonter ces difficultés Buyurgan et Saygin **[Buyurgan 06b]** ont employé les concepts de la logique floue, la corrélation incertaine et complexe parmi les différents critères est incorporée au processus décisionnel (ex : quel quantité d'équilibre moyen d'utilisation peut être sacrifiée pour réduire le temps d'écoulement moyen par unité) mais en raison de fuzzyfication et défuzzyfication pour chaque état future possible, la complexité des calculs augmente considérablement.

Saygin et Kilic **[Saygin 99]** ont proposé une plateforme qui intègre le processus de planification flexible et l'ordonnancement prédictif (en temps différé), il ont présenté un concept nommé dissimilarity maximisation method (DMM) pour minimiser la congestion dans un FMS, l'idée de cette règle c'est de maximiser les dissimilitudes entre les routages occupés. L'efficacité de cette règle dans la résolution des problèmes de sélection de routage en temps réel a été démontré dans **[Saygin 01]** où elle a dépassé d'autres règles tel que FIFO/FA (first-in first-out/ first available) et equal probability loading (EPL) si chaque machine utilise la règle FIFO, et dans **[Saygin 04]** où les trois règles ont été combinée avec d'autres règles de priorité. Dans l'étude de Hassam et Sari **[Hassam 07]** sur la règle DMM, Il ont remarqué que pour un taux d'arrivée de pièces important et pour une capacité des tailles de files d'attentes faible le système de production est saturé et le taux d'utilisation des machines et du transporteur est assez faible, ce qui va influencer sur les performances du système de production. Pour cela ils ont proposé la règle DMM modifiée qui est une modification de la règle DMM qui vise à garder le même principe mais si tous les routages sont sélectionnés par des pièces, la pièce suivante va être acheminée dans le routage où la file d'attente de la première machine de ce routage, contient au moins une place libre. Il a été démontré dans **[Ghomri 07]** que les règles DMM et DMM modifiée sont les meilleurs si on les compares avec d'autres règles de sélection de routage qui sont EPL (Equal Probability loading), FA (First Available), SQ (Shortest Queue), LWQ (Least Work in Queue).

D'autres chercheurs sont allés a utiliser les métaheuristiques, Zhao et Wu **[Zhao 01]** ont adapté les algorithmes génétiques pour résoudre le problème de séquencement des jobs dans un FMS avec flexibilité de routages et des machines mais cette algorithme a été appliqué en temps différé et un nombre limité de pièces. Rossi et Dini **[Rossi 07]** ont proposé un système logiciel basé sur l'optimisation par les colonies de fourmis pour résoudre le problème d'ordonnancement dans un job shop avec flexibilité de routages et installation séparable. Rossi et Boschi **[Rossi 09]** ont proposé un autre système basé sur la combinaison des colonies de fourmis et les algorithmes génétiques mais les systèmes proposés ont été testés que pour un nombre limité d'opérations. Dans **[Souier 08]** nous avons adapté plusieurs métaheuristiques (les colonies de fourmis, les algorithmes génétiques, les essaims de particules, le recuit simulé, la recherche tabou et l'électromagnétisme) pour résoudre le problème de sélection de routage dans un FMS avec un nombre important de pièces mais en temps différé.

Certains chercheurs ont pensés à prendre en considération la flexibilité du routage avec d'autres problèmes, Buyurgan et al **[Buyurgan 04]** ont proposée une approche heuristique pour la sélection des outils dans un FMS, en utilisant le rapport (durée de vie virtuelle d'un outil / nombre d'outils dans le magasin) cette approche sélectionne le type d'outil qui a le rapport maximum en considérant les alternatifs des outils pour les opérations assignés aux machines tout en gardant la flexibilité de routages alternatifs. Liu et al **[Liu 01]** ont développé un algorithme d'ordonnancement pour obtenir le temps optimal de remplacement des outils, l'algorithme a été présenté pour trouver le coût minimal, la limite de commande et la décision optimale de remplacement d'outil, la flexibilité de routage a été prise en compte dans leur modèle. Sabuncuoglu et Lahmar **[Sabuncuoglu 03]** ont étudié le problème de chargements de machines dans un FMS par simulation, ils ont comparé les politiques d'agrégation (assigner toutes les opérations d'un type de pièces à une seule machine) et désagrégation (assigner les opérations d'une pièce à plusieurs machines) et ont analysé leurs interaction avec d'autres facteurs tel que la flexibilité de routage, de séquencement, la charge d'une machine…

1.6. Conclusion

Dans ce chapitre, nous avons abordé les spécifications d'ordonnancement dans les systèmes flexibles de production.

La plupart des méthodes et techniques de résolution de ces problèmes se placent dans une perspective d'ordonnancement prévisionnel, où l'ensemble des tâches à réaliser est ordonnancé à priori. Mais cette approche d'ordonnancement prévisionnel n'est pas toujours applicable, compte tenu du fonctionnement aléatoire et souvent perturbé de certains ateliers. C'est pourquoi, une approche d'ordonnancement temps réel, permettant de s'adapter aux événements (attendus ou aléas) apparaissant dans l'état du système, est nécessaire pour gérer au mieux les systèmes flexibles de production.

La gestion des files d'attente selon des règles de priorité est l'une des méthodes les plus simples, et parmi les plus utilisées pour gérer en temps réel les ateliers, mais ces règles gèrent le travail au niveau du poste de charge et ne permettent pas donc de prendre en considération la flexibilité de routages.

L'ordonnancement de type job shop est généralement de type complexe. Il s'agit d'un problème combinatoire démontré comme étant NP-difficile. Ce qui signifie que les algorithmes optimaux existant pour ce problème sont impossibles à utiliser pratiquement lorsque la taille du problème augmente.

L'avènement des métaheuristiques a permis de résoudre plusieurs problèmes pour lesquels il n'existe pas de méthodes spécifiques, dans notre état de l'art nous avons regroupé les travaux les plus récents qui traitent l'adaptation des métaheuristiques les plus connues pour résoudre les problèmes d'ordonnancement dans les FMS sans apporter de détailles sur leurs caractéristiques, leurs classifications, leurs algorithmes de base… Nous avons donc choisi d'utiliser ces métaheuristiques pour résoudre le problème d'ordonnancement dans un job shop. Afin de pouvoir réaliser cela, nous consacrons le chapitre suivant à la présentation détaillée de ces techniques.

Chapitre 2

Métaheuristiques pour l'optimisation difficile

2.1. Introduction

Les ingénieurs et les décideurs sont confrontés quotidiennement à des problèmes de complexité grandissante, qui surgissent dans des secteurs techniques très divers, comme dans la recherche opérationnelle, la conception de systèmes mécaniques, le traitement d'images, en électroniques… Le problème à résoudre peut souvent s'exprimer comme un problème d'optimisation : on sélectionne une ou plusieurs variables, on définit une fonction objectif ou fonction de coût que l'on cherche à minimiser ou à maximiser par rapport à tous les paramètres et les contraintes concernés.

Il existe des méthodes déterministes permettant de résoudre certains problèmes d'optimisation en un temps fini. Ces méthodes nécessitent généralement un certain nombre de caractéristiques de la fonction objectif, comme la stricte convexité, la continuité ou encore la dérivabilité. On peut citer comme exemple de méthodes : la programmation linéaire, quadratique ou dynamique, la méthode du gradient, la méthode de Newton…

Certains problèmes d'optimisation demeurent cependant hors de portée des méthodes exactes. Un certain nombre de caractéristiques peuvent en effet être problématiques, comme l'absence de convexité stricte, l'existence de discontinuités, une fonction non dérivable, présence de bruit, etc.

Dans de tels cas, le problème d'optimisation est dit difficile, car aucune méthode exacte n'est capable de le résoudre exactement en un temps raisonnable, on devra alors faire appel aux techniques permettant une optimisation approchée.

Nous nous intéressant dans ce chapitre à un groupe de méthodes dénommées métaheuristiques qui forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile, pour lesquels on ne connaît pas de méthode classique plus efficace. Elles sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé.

Dans la section 2.2, nous présentons la définition, les domaines d'utilisation, les caractéristiques et la classification des métaheuristiques. La section 2.3 sera consacrée aux principes des métaheuristiques les plus répondus (le recuit simulé SA, la recherche tabou TS, les algorithmes génétiques GA, les colonies de fourmis ACO, les essaims particulaires PSO, électromagnétisme métaheuristique EM) en citant leurs définitions, leurs origines et leurs algorithmes de bases.

2.2. Généralités sur les métaheuristiques

2.2.1 Définition

Avant l'adoption du mot métaheuristique, on parlait plutôt d'heuristique qui est une méthode approchée se voulant simple, rapide et adaptée à un problème donnée. Sa capacité à optimiser un problème avec un minimum d'informations est contrebalancée par le fait qu'elle n'offre aucune garantie quant à l'optimalité de la meilleure solution trouvée.

Du point de vue de la recherche opérationnelle, ce défaut n'est pas toujours un problème, tout spécialement quand seule une approximation de la solution optimale est recherchée. Mais la plupart des heuristiques ont été conçues spécifiquement pour un problème donné.

Contrairement aux heuristiques, les métaheuristiques visent à résoudre une large gamme de problèmes différents sans changements majeurs dans l'algorithme **[Dréo 04]**.

Les premières métaheuristiques datent des années 80, et bien que d'origines discrètes, on peut les adapter à des problèmes continus. Elles interviennent généralement quand les méthodes classiques ont échoué, et sont d'une efficacité relativement imprévisible. Le terme métaheuristique est utilisé car, par opposition aux heuristiques particulières pour un problème donné, les métaheuristiques peuvent être utilisées pour plusieurs types de problèmes, et l'heuristique n'est pas réellement explicite, mais déduite d'un algorithme donné. Les métaheuristiques ont également comme caractéristiques communes leur caractère plus ou moins stochastique, ainsi que leur inspiration par une analogie avec d'autres sciences (physique, biologie, etc.).

2.2.2 Domaines d'utilisation des métaheuristiques

Les métaheuristiques sont utilisées pour résoudre les problèmes d'optimisation difficile qui sont des problèmes pour lesquelles aucune méthode exacte n'est capable de résoudre exactement en un temps raisonnable. Ces problèmes peuvent se découper en deux types de problèmes : les problèmes discrets et les problèmes continus.

- certains problèmes d'optimisation discrète, pour lesquels on ne connaît pas d'algorithme exact polynomial (c'est à dire dont le temps de calcul est proportionnel à N^n, où N désigne le nombre de paramètres inconnus du problème, et n est une constante entière). C'est le cas, en particulier, des problèmes dits "NP difficiles", pour lesquels on conjecture qu'il n'existe pas un constant n tel que le temps de résolution soit borné par un polynôme de degré n.
- certains problèmes d'optimisation à variables continues, pour lesquels on ne connaît pas d'algorithme permettant de repérer un optimum global (c'est-à-dire la meilleure solution possible) à coup sûr et en un nombre fini de calculs. **[Dréo 03], [Dréo 04]**

Des efforts ont été longtemps menés, séparément, pour résoudre ces deux types de problèmes. Dans l'optimisation continue, il existe ainsi un arsenal important de méthodes classiques dites d'optimisation global **[Horst 95]**, mais ces techniques sont souvent inefficaces si la fonction objectif ne pas possède une propriété structurelle particulière, tel que la convexité. Dans le domaine de l'optimisation discrète, un grand nombre d'heuristiques qui produisent une solution proche de l'optimum ont été développées mais la plupart d'entre elles ont été conçues spécifiquement pour un problème donné.

L'arrivée des métaheuristiques marque une réconciliation des deux domaines (celle ci s'appliquent à toutes sortes de problèmes discrets et peuvent s'adapter aussi aux problèmes continues). En pratique, certains problèmes sont mixtes et présentent à la fois des variables discrètes et des variables continues. On peut donc souligner une autre richesse des métaheuristiques : elle se prêtent à toutes sortes d'extensions.

2.2.3 Caractéristiques et classification des métaheuristiques

Selon Clerc et Siarry **[Clerc 04]** les métaheuristiques ont en commun les caractéristiques suivantes :

- La plupart des métaheuristiques utilisent des processus aléatoires comme moyens de récolter de l'information et de faire face à des problèmes comme l'explosion combinatoire.
- En plus de cette base stochastique, les métaheuristiques sont généralement itératives, c'est à dire qu'un même schéma de recherche est appliqué plusieurs fois au cours de l'optimisation, et directes, c'est à dire qu'elles n'utilisent pas l'information du gradient de la fonction objectif. Elles tirent en particulier leur intérêt de leur capacité à éviter les optimums locaux, soit en acceptant une dégradation de la fonction objectif au cours de leur progression, soit en utilisant une population de points comme méthode de recherche.
- Les métaheuristiques du fait de leur capacité à être utilisées sur un grand nombre de problèmes différents, se prêtent facilement à des extensions.
- Souvent d'origine discrète à l'exception des essaims de particules et l'électromagnétisme.
- Elles sont inspirées par analogie avec la réalité: avec la physique (le recuit simulé), avec la biologie (les algorithme génétiques) ou avec l'éthologie (les colonie de fourmis)...
- Les concepts de base des métaheuristiques peuvent être décrit de manière abstraite, sans faire appel à un problème spécifique.
- Les métaheuristiques peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche.
- Elles partagent aussi les mêmes inconvénients : les difficultés de réglage des paramètres, et le temps de calcul élevé.

On peut faire la différence entre les métaheuristiques qui s'inspirent de phénomènes naturels et celles qui ne s'en inspirent pas. Par exemple, les algorithmes génétiques et les algorithmes des colonies de fourmis s'inspirent respectivement de la théorie de l'évolution et du comportement de fourmis à la recherche de nourriture. Par contre, la méthode tabou n'a semble-t-il pas été inspirée par un phénomène naturel. Une telle classification ne semble cependant pas très utile et est parfois difficile à réaliser. En effet, il existe de nombreuses métaheuristiques récentes qu'il est difficile de classer dans l'une des deux catégories. Certains se demanderont par exemple si l'utilisation d'une mémoire dans la méthode tabou n'est pas directement inspirée de la nature.

Une autre façon de classifier les métaheuristiques est de distinguer celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois. Les méthodes qui tentent itérativement d'améliorer une solution sont appelées méthodes de recherche locale ou méthodes de trajectoire. La méthode tabou, le recuit simulé sont des exemples typiques de

méthodes de trajectoire. Ces méthodes construisent une trajectoire dans l'espace des solutions en tentant de se diriger vers des solutions optimales. L'exemple le plus connu de méthode qui travaille avec une population de solutions est l'algorithme génétique.

Les métaheuristiques peuvent également être classées selon leur manière d'utiliser la fonction objective. Étant donné un problème d'optimisation consistant à minimiser une fonction f sur un espace S de solutions, certaines métaheuristiques dites statiques travaillent directement sur f alors que d'autres, dites dynamiques, font usage d'une fonction g obtenue à partir de f en ajoutant quelques composantes qui permettent de modifier la topologie de l'espace des solutions, ces composantes additionnelles pouvant varier durant le processus de recherche.

Des chercheurs préfèrent classifier les métaheuristiques en fonction du nombre de structures de voisinages utilisées. Étant donné qu'un minimum local relativement à un type de voisinage ne l'est pas forcément pour un autre type de voisinage, il peut être intéressant d'utiliser des métaheuristiques basées sur plusieurs types de voisinages.

Certaines métaheuristiques font usage de l'historique de la recherche au cours de l'optimisation, alors que d'autres n'ont aucune mémoire du passé. Les algorithmes sans mémoire sont en fait des processus markoviens puisque l'action à réaliser est totalement déterminée par la situation courante. Les métaheuristiques qui font usage de l'historique de la recherche peuvent le faire de diverses manières. On différentie généralement les méthodes ayant une mémoire à court terme de celles qui ont une mémoire à long terme.

2.3. Principes des métaheuristiques les plus répondues

2.3.1. Le recuit simulé (Simulated Annealing : *SA*)

2.3.1.1. Le recuit réel et le recuit simulé

La méthode de recuit simulée a été conçue par Kirkpatrick et al en 1983 **[Kirkpatrick 83]** qui étaient des spécialistes de physique. La détermination numérique des configurations des systèmes physiques posait de redoutables problèmes d'optimisation. L'analogie entre un problème d'optimisation et un système physique est présentée dans le tableau suivant :

Problème d'optimisation	Système physique
Fonction objective	Energie libre
Paramètres du problème	Coordonnées des particules
Trouver une bonne configuration	Trouver les états à basse énergie

Tableau 2.1 : Analogie entre un problème d'optimisation et un système physique.

Kirkpatrick et al **[Kirkpatrick 83]** ont proposé de traiter ces problèmes en s'inspirant de la technique expérimentale du recuit utilisé par les métallurgistes pour obtenir un état solide bien ordonnée avec énergie minimale. Cette technique consiste à porter le matériau à haute température T puis baisser lentement celle-ci.

A titre d'illustration, la figure 2.1 représente l'effet de la technique du recuit, et celui de la technique opposée de la trempe sur un système formé d'un ensemble de particules.

La technique du recuit consiste à chauffer préalablement le matériau pour lui conférer une énergie élevée. Puis on refroidit lentement le matériau en marquant des paliers de température de durée suffisante, si la descente de température est trop rapide, il apparaît des défauts qui peuvent être éliminé par réchauffement local. Cette stratégie de baisse contrôlée de la température conduit à un état solide stable à un minimum absolu de l'énergie. La technique opposée est celle de la trempe qui consiste à baisser très rapidement la température, on obtient dans ce cas une structure métastable qui correspond à un minimum local de l'énergie.

L'idée d'utiliser la technique de recuit en vue de traiter un problème d'optimisation a donnée naissance au recuit simulé. Pour terminer la présentation de la méthode, il nous reste à explorer l'algorithme qui permet de programmer le recuit sur un ordinateur.

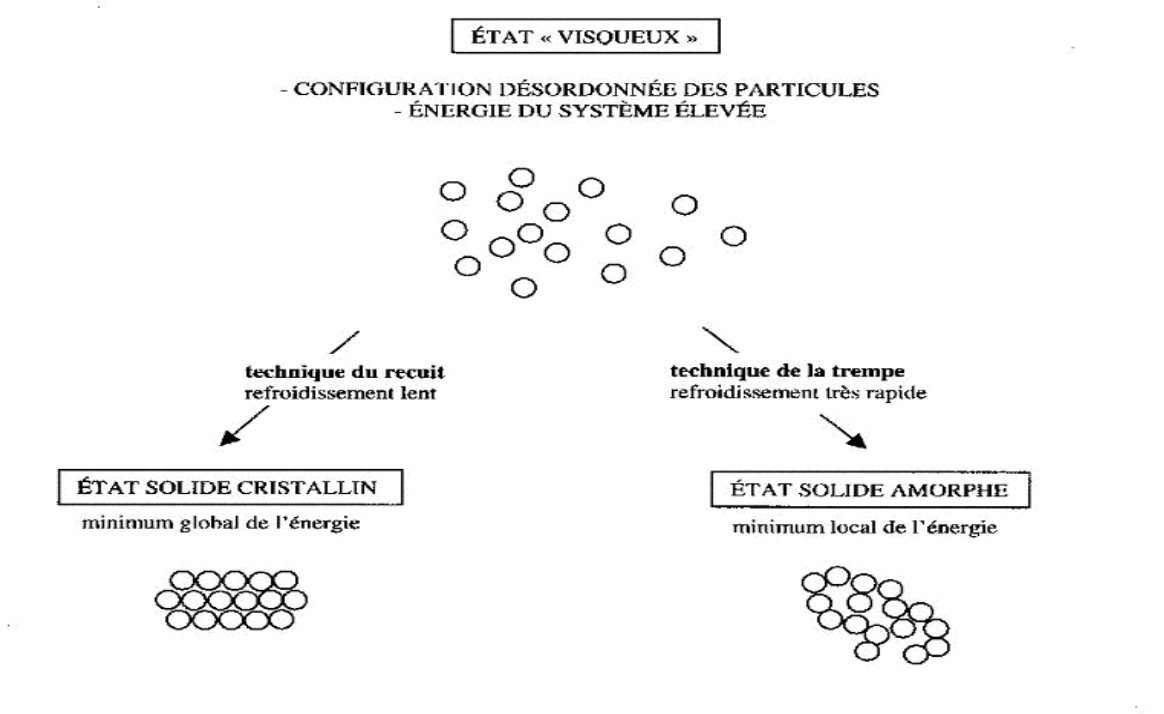

Figure 2.1 : Comparaison des techniques du recuit et de la trempe [Dréo 04].

2.3.1.2. L'algorithme de base

Cette méthode d'optimisation est basée sur les travaux de Metropolis **[Metropolis 53]** qui permet de décrire le comportement d'un système dans l'équilibre thermodynamique à une certaine température.

Cette technique transporte le procédé du recuit à la résolution d'un problème d'optimisation : la fonction objective à minimiser étant l'énergie E du matériel. La température T est également introduite, qui est dans ce cas un paramètre de contrôle de l'algorithme.

Le principe de cet algorithme est simple. A partir d'un état initial à une température T, on génère une autre solution par une modification élémentaire de manière aléatoire. Si cette solution améliore la fonction objectif, cette dernière est automatiquement acceptée. Si elle dégrade la

fonction objectif elle peut être acceptée avec une probabilité *exp(-ΔE)* où *ΔE* est la variation de la fonction objectif, une fois que l'équilibre thermodynamique est atteint, on baisse la température légèrement avant d'effectuer une nouvelle itération. L'organigramme du recuit simulé est présenté dans la figure 2.2.

Depuis son apparition, la méthode du recuit simulé a prouvé son efficacité dans des domaines aussi divers que la conception électronique, le traitement d'images, l'organisation des réseaux informatiques des loto…

Les inconvénients de cette technique résident d'une part dans les réglages des paramètres, comme la gestion de décroissance de la température, de bons réglages relèvent souvent du savoir faire de l'utilisateur. D'autre part, les temps de calcul peuvent devenir très importants, ce qui a conduit à des parallélisations de la méthode. En revanche, la méthode du recuit simulé a l'avantage d'être souple vis-à-vis des évolutions du problème et facile à implémenter. Elle a donnée d'excellents résultats pour nombres de problèmes, le plus souvent de grande taille.

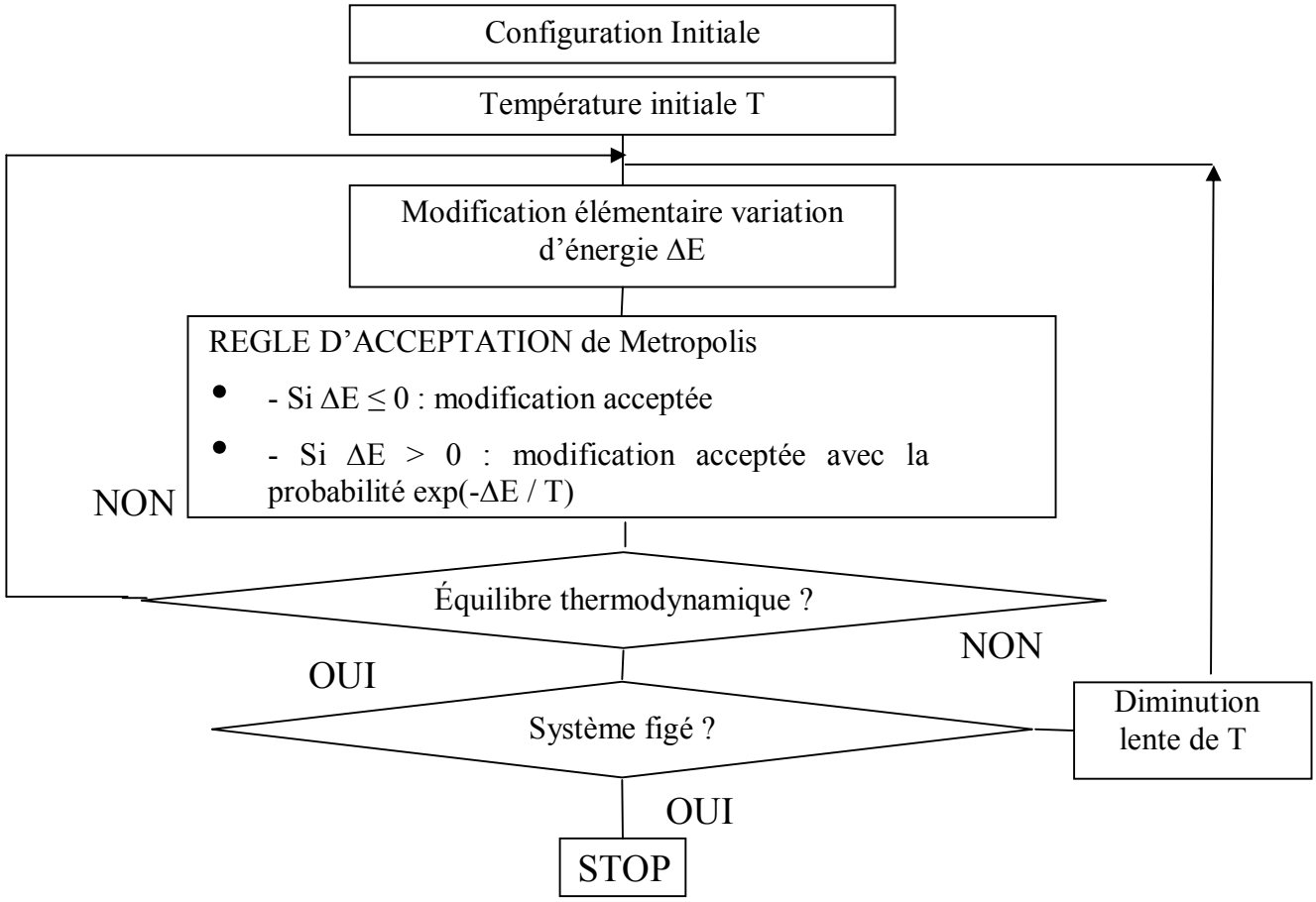

Figure 2.2 : l'organigramme du recuit simulé [Dréo 04].

2.3.2. Recherche tabous (Taboo Search : *TS*)

La méthode de recherche avec tabous, ou simplement recherche tabou ou méthode tabou, a été formalisée par Glover **[Glover 97]**. Sa principale particularité tient dans la mise en œuvre de mécanismes inspirés de la mémoire humaine. La méthode tabou prend sur ce plan, le contre-pied du recuit simulé, totalement dépourvu de mémoire, et donc incapable de tirer les leçons du passé.

Le principe de base de la méthode tabou est simple : comme le recuit simulé, la méthode tabou fonctionne avec une seul configuration courante à la fois (au départ, une solution quelconque), qui est actualisé au cours des itérations successives.

Le principe est de choisir à chaque itération la meilleure solution $T \in V(s)$ (l'ensemble des voisins de *s*).

A chaque itération, le mécanisme de passage d'une configuration, soit *s* à la suivante *t* comporte deux étapes:

- On construit l'ensemble des voisins de *s*, c'est-à-dire l'ensemble des configurations accessibles en un seul mouvement élémentaire à partir de *s*, soit *V(s)* l'ensemble de ces voisins.
- On évalue la fonction objectif *f* du problème en chacune des configurations de *V(s)*, la configuration *t* est la meilleure de ces configurations. Cette configuration est adoptée même si elle est moins bonne que *s*. c'est grâce à cette particularité que la méthode tabou permet d'éviter des minimums locaux de *f*.

Le danger est alors de revenir à une configuration déjà retenu. Pour éviter ce phénomène on crée une liste *Liste_tab* qui mémorise les dernières solutions visitées et interdit tout déplacement vers une solution de cette liste. Cette liste *Liste_tab* est appelée liste taboue, Les solutions ne demeurent dans *Liste_tab* que pour un nombre limité d'itérations. La liste *Liste_tab* est donc une mémoire à court terme. Si une solution est dans *Liste_tab* on dit que c'est une solution taboue. De même, tout mouvement qui nous mène de la solution courante à une solution de *Liste_tab* est appelé mouvement tabou. L'organigramme de cet algorithme est représenté sur la figure suivante :

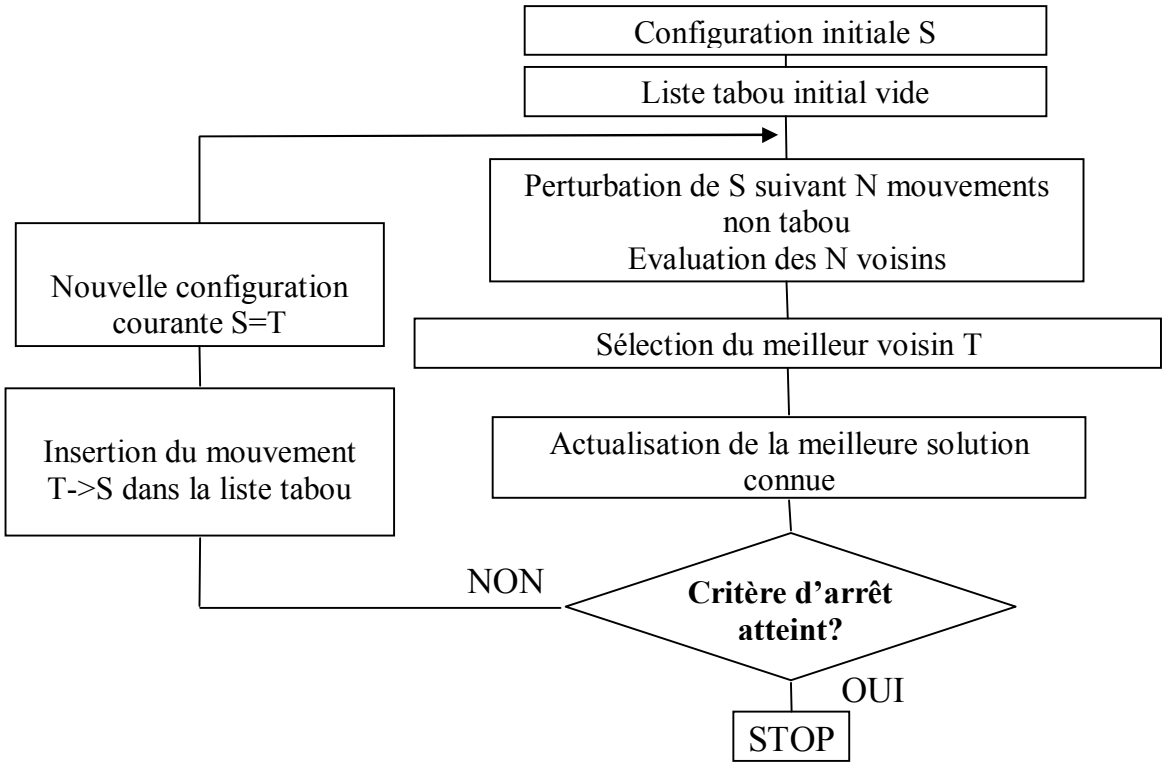

Figure 2.3 : l'organigramme de la recherche tabou [Dréo 04].

L'algorithme modélise ainsi une forme de mémoire, la mémoire à court terme des solutions visitées récemment. Pour certains problèmes d'optimisation comme d'affectation quadratique, la méthode tabou a donnée d'excellents résultats, en outre sous sa forme de base, la méthode comporte moins de paramètres de réglage que le recuit simulé, ce qui la rend plus simple d'emploi.

2.3.3. Les algorithmes génétiques (Genetic Algorithms : *GA*)

Les algorithmes évolutionnaires sont des techniques de recherche inspirées par l'évolution biologique des espèces apparues à la fin des années 50 **[Fraser 57]**. Ces méthodes ont connu d'abord un intérêt limité du fait de leur important coût d'exécution, mais elles connaissent depuis plus de dix ans un développement considérable grâce notamment à l'augmentation de la puissance des calculateurs.

Dans les années 1960 à 1970, des que les premiers calculateurs de puissance plus crédible ont commencé à être accessibles, de nombreuses tentatives de modélisation de l'évolution ont été entreprise. Parmi ces tendances on trouve les algorithmes génétiques.

2.3.3.1. Définition des algorithmes génétiques

Les algorithmes génétiques sont le type d'algorithme le plus connu et le plus utilisé des algorithmes évolutionnaires, développée dans les années 70 par Holland **[Holland 75]** puis approfondis par Goldberg en 1989 **[Goldberg 89]**, les algorithmes génétiques possèdent la particularité qu'ils font évoluer des populations d'individus codés généralement en binaire.

Les algorithmes génétiques sont devenus après la modification de Goldberg extraordinairement populaires.

Dans un GA, chaque solution est stockée dans un chromosome artificiel représenté par un code. Chacun de ces chromosomes est défini par deux caractéristiques :

- Le phénotype est l'apparence de l'individu dans une représentation naturelle nous permettant de savoir si la solution est bonne ou mauvaise.
- Le génotype est la représentation des gènes de l'individu est fonction de ses chromosomes. Le programmeur doit dans un premier temps choisir la représentation chromosomique (génotype) de chacun des individus de la population.

2.3.3.2. Principe de fonctionnement d'un algorithme génétique

Selon Darwin, **[Darwin 59]** les mécanismes à l'origine de l'évolution des êtres vivants reposent sur la compétition qui sélectionne les individus les plus adaptés à leur milieu en leur assurant une descendance, ainsi que sur la transmission aux enfants des caractéristiques utiles qui ont permis la survie des parents.

Les algorithmes génétiques sont des méthodes de recherche probabilistes, basés sur ce modèle de l'évolution, les individus soumis à l'évolution sont des solutions appartenant à l'espace de recherche du problème à optimiser. L'ensemble des individus traités simultanément par l'algorithme génétique constitue une population, l'algorithme itère et passe d'une génération à l'autre jusqu'à la satisfaction d'un critère d'arrêt. Durant chaque génération, un ensemble d'opérateurs est appliquée aux individus d'une population pour engendrer la nouvelle population à la génération suivante.

Le principe d'un algorithme génétique se décrit simplement. Un ensemble de *N* points dans un espace de recherche, choisi à priori au hasard, constitue la population initiale; chaque individu *x* de la population possède une certaine performance, qui mesure son degré d'adaptation à l'objectif visé: dans le cas de la minimisation d'une fonction objectif *f*, *x* est d'autant plus performant que *f(x)* est plus petit. Un AG consiste à faire évoluer progressivement par générations successives, la composition de la population en maintenant sa taille constante. Au cours des générations, l'objectif est d'améliorer globalement la performance des individus; on essaie d'obtenir un tel résultat en mimant les deux principaux mécanismes qui régissent l'évolution des êtres vivants, selon la théorie de Darwin :

- La sélection, qui favorise la reproduction et la survie des individus les plus performants.
- La reproduction, la recombinaison et les variations des caractères héréditaires des parents, pour former des descendants aux potentialités nouvelles.

Selon Goldberg **[Goldberg 89]**, les opérateurs de reproduction sont classés en deux catégories :

- Les opérateurs de mutation, qui modifient un individu pour former un autre, cet opérateur consiste à modifier la valeur d'un génotype aléatoirement avec un faible taux de mutation, il existe plusieurs variantes dont les plus connues sont la mutation déterministe et la mutation bit-flip, la mutation bit-flip consiste à choisir un bit dans la chaîne binaire et l'inverser tandis que la mutation déterministe consiste à choisir au hasard un nombre fixé de bits et à les inverser.
- Les opérateurs de croisement, qui engendre un ou plusieurs enfants à partir de combinaison de deux parents, Il existe trois variantes de croisement : le croisement un point, le croisement deux points et le croisement uniforme. Le premier se déroule en deux étapes : le choix aléatoire d'un point de coupure identique sur les deux chaînes binaires et la coupure des deux chaînes et échange des deux fragments situés à droite. Dans le deuxième on choisi aléatoirement deux points de coupure sur les deux chaînes binaires et on coupe les deux chaînes et échange des deux fragments situées entre les deux points. Dans le croisement uniforme, on utilise un "masque de croisement" qui est un mot binaire de même taille que les individus, si la n^{ieme} position du masque est "0" on laisse inchangés les symboles à la n^{ieme} position des deux chaînes, un "1" déclenche un échange des symboles correspondants, le masque est engendré aléatoirement pour chaque couple d'individus.

Le schéma d'un algorithme génétique simple est présenté dans la figure 2.4. On remarque qu'il met en oeuvre un opérateur de sélection proportionnel (la chance de sélection d'un individus est proportionnel à sa fitness), et un remplacement générationnel c'est-à-dire que la population des enfants remplace complètement celle des parents. Les opérateurs de variations travaillent sur les génotypes, le croisement est considéré comme l'opérateur de recherche essentiel, la mutation est appliquée avec un faible taux afin de garder une certaine diversité dans la population.

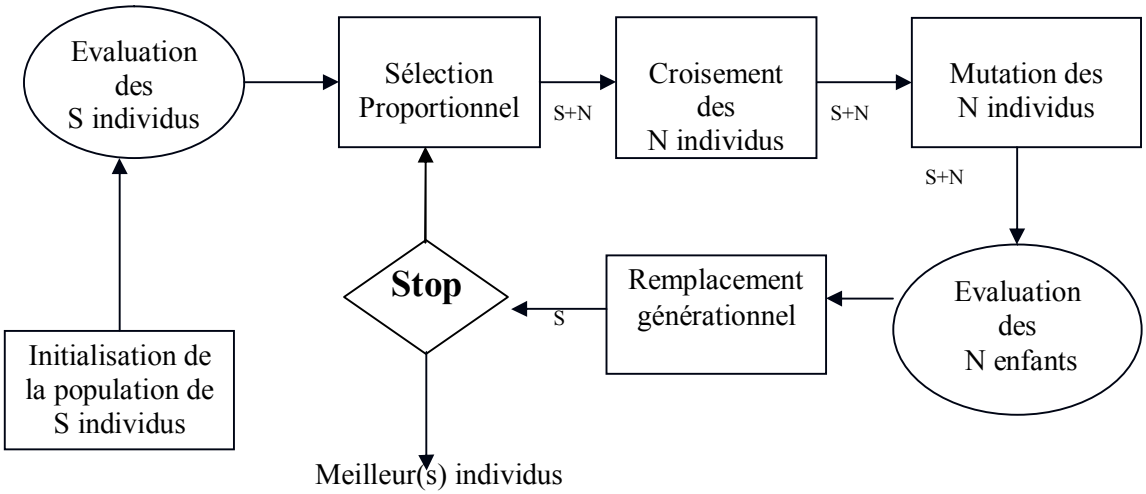

Figure 2.4 : le schéma d'un algorithme génétique [Dréo 04].

2.3.4. Les colonies de fourmis (Ant Colony Optimization : *ACO*)

Les algorithmes de colonies de fourmis forment une classe de metaheuristiques récemment proposée pour les problèmes d'optimisation difficiles. Ces algorithmes s'inspirent des comportements collectifs de dépôt et de suivi de piste observés dans les colonies de fourmis. Une colonie d'agents simples (les fourmis) communiquent indirectement via des modifications dynamiques de leur environnement (les pistes de phéromone) et construisent ainsi une solution à un problème, en s'appuyant sur leur expérience collective.

Le premier algorithme de ce type (Ant system) a été conçu pour le problème du voyageur de commerce, mais n'a pas permis de produire des résultats compétitifs. Cependant, l'intérêt pour la métaphore était lancée et de nombreux algorithmes s'en inspirant ont depuis été proposés dans divers domaines, certains atteignant des résultats très convaincants.

2.3.4.1. Optimisation naturelle: pistes de phéromone

Les algorithmes de colonies de fourmis sont nés à la suite d'une constatation : les insectes sociaux en général, et les fourmis en particulier, résolvent naturellement des problèmes relativement complexes. Les biologistes ont étudié comment les fourmis arrivent à résoudre collectivement des problèmes trop complexes pour un seul individu, notamment les problèmes de choix lors de l'exploitation de sources de nourriture.

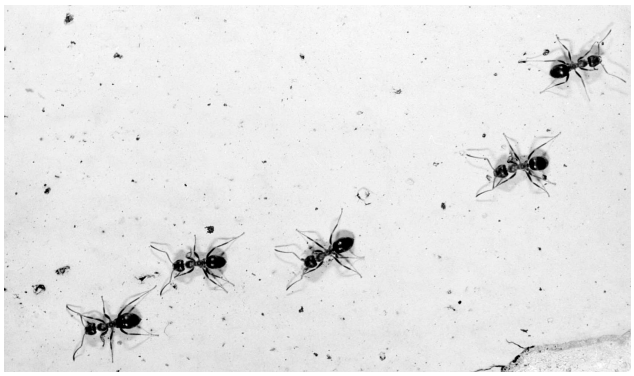

Figure 2.5 Des fourmis suivant une piste de phéromone [Dréo 03]

Les fourmis ont la particularité d'employer pour communiquer des substances volatiles appelées phéromones. Elles sont attirées par ces substances, qu'elles perçoivent grâce à des récepteurs situés dans leurs antennes. Ces substances sont nombreuses et varient selon les espèces. Les fourmis peuvent déposer des phéromones au sol, grâce à une glande située dans leur abdomen, et former ainsi des pistes odorantes, qui pourront être suivies par leurs congénères (figure 2.5).

Les fourmis utilisent les pistes de phéromone pour marquer leur trajet, par exemple entre le nid et une source de nourriture. Une colonie est ainsi capable de choisir (sous certaines conditions) le plus court chemin vers une source à exploiter, **[Goss 89]**, **[Beckers 92]** sans que les individus aient une vision globale du trajet.

2.3.4.2. Optimisation par colonies de fourmis et problème du voyageur de commerce

Le problème du voyageur de commerce consiste à trouver le trajet le plus court reliant *n* villes données, chaque ville ne devant être visitée qu'une seule fois. Ce problème a fait l'objet de la première implémentation d'un algorithme de colonies de fourmis : le Ant System (AS) **[Colorni 92]**. Le passage de la métaphore à l'algorithme est ici relativement facile et le problème du voyageur de commerce est bien connu et étudié.

2.3.4.2.1 Algorithme de base

Dans l'algorithme AS, à chaque itération t ($1 \leq t \leq t_{max}$), chaque fourmi k ($k = 1$ à m) parcourt le graphe et construit un trajet complet de $n = card(N)$ étapes (on note $card(N)$ le cardinal de l'ensemble N). Pour chaque fourmi, le trajet entre une ville i et une ville j dépend de :

1. la liste des villes déjà visitées, qui définit les mouvements possibles à chaque pas, quand la fourmi k est sur la ville i : j_i^k

2. l'inverse de la distance entre les villes : $\eta_{ij} = 1/d_{ij}$ appelée visibilité. Cette information statique est utilisée pour diriger le choix des fourmis vers des villes proches, et éviter les villes trop lointaines.

3. la quantité de phéromone déposée sur l'arête reliant les deux villes, appelée l'intensité de la piste.

La règle de déplacement est la suivante **[Bonabeau 99]** :

$$p_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^\alpha . (\eta_{ij}(t))^\beta}{\sum_{i \in j_i^k} (\tau_{ij}(t))^\alpha . (\eta_{ij}(t))^\beta} & Si \quad j \in j_i^k \\ 0 \quad Si \quad j \notin j_i^k \end{cases} \qquad (2.1)$$

Où α et β sont deux paramètres contrôlant l'importance relative de l'intensité de la piste, $\tau_{ij}^k(t)$ et de la visibilité η_{ij}.

Après un tour complet, chaque fourmi laisse une certaine quantité de phéromone $\Delta \tau_{ij}^k(t)$ sur l'ensemble de son parcours, quantité qui dépend de la qualité de la solution trouvée.

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & si (i,j) \in T^k(t) \\ 0 & si (i,j) \notin T^k(t) \end{cases} \qquad (2.2)$$

Où T^k *(t)* est le trajet effectué par la fourmi *k* à l'itération *t*, L^k *(t)* la longueur de la tournée et *Q* un paramètre fixé.

L'algorithme ne serait pas complet sans le processus d'évaporation des pistes de phéromone. En effet, pour éviter d'être piégée dans des solutions sous-optimales, il est nécessaire de permettre au système d'oublier les mauvaises solutions. On contrebalance donc l'additivité des pistes par une décroissance constante des valeurs des arêtes à chaque itération. La règle de mise à jour des pistes est donc :

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \qquad (2.3)$$

Où *m* est le nombre de fourmis et ρ le taux d'évaporation. La quantité initiale de phéromone sur les arêtes est une distribution uniforme d'une petite quantité $\Gamma_0 \geq 0$

Le pseudo- code de l'algorithme est :

Pour *t = 1* à t_{max}

Pour chaque fourmi *k = 1* à *m*

Choisir une ville au hasard

Pour chaque ville non visitée *i*

Choisir une ville *j*, dans la liste j_i^k des villes restantes, selon la formule (2.1)

Fin Pour

Déposer une piste $\Delta\tau_{ij}^{k}(t)$ sur le trajet $T^k(t)$ conformément à l'équation (2.2)

Fin Pour

Évaporer les pistes selon la formule (2.3)

Fin Pour

2.3.4.2.2 Variantes [Dréo 03]

A. Système des fourmis & élitisme

Une première variante du Système de Fourmis (Ant System) a été proposée dans **[Dorigo 96]**. Elle est caractérisée par l'introduction de fourmis élitistes. Dans cette version, la meilleure fourmi (celle qui a effectué le trajet le plus court) dépose une quantité de phéromone plus grande, dans l'optique d'accroître la probabilité des autres fourmis d'explorer la solution la plus prometteuse.

B. Système des colonies de fourmis (Ant colony system)

L'algorithme Ant Colony System (ACS) a été introduit pour améliorer les performances du premier algorithme sur des problèmes de grandes tailles **[Dorigo 97a]**, **[Dorigo 97b]**. ACS est fondé sur des modifications du AS :

1. ACS introduit une règle de transition dépendant d'un paramètre q_0 ($0 \leq q_0 \leq 1$), qui définit une balance diversification /intensification. Une fourmi *k* sur une ville *i* choisira une ville *j* par la règle :

$$j = \begin{cases} \arg\max_{u \in J_j^k} \left[(\tau_{iu}(t)).(\eta_{ij})^{\beta} \right] & si \quad q \le q_0 \\ j & si \quad q \succ q_0 \end{cases} \qquad (2.4)$$

Où q est une variable aléatoire uniformément distribuée sur [0, 1] et $j \in j_i^k$ une ville sélectionnée aléatoirement selon la probabilité :

$$P_{iJ}^k(t) = \frac{(\tau_{iJ}(t)).(\eta_{iJ})^{\beta}}{\sum_{l \in j_i^k} (\tau_{il}(t)).(\eta_{il}(t))^{\beta}} \qquad (2.5)$$

En fonction du paramètre q_0, il y a donc deux comportements possibles : si $q > q_0$ le choix se fait de la même façon que pour l'algorithme AS, et le système tend à effectuer une diversification, si $q \le q_0$, le système tend au contraire vers une intensification. En effet, pour $q \le q_0$, on ne peut pas choisir un trajet non exploré.

2. La gestion des pistes est séparée en deux niveaux : une mise à jour locale et une mise à jour globale. Chaque fourmi dépose une piste lors de la mise à jour locale :

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \rho.\tau_0 \qquad (2.6)$$

Où τ_0 est la valeur initiale de la piste. A chaque passage, les arêtes visitées voient leur quantité de phéromone diminuer, ce qui favorise la diversification par la prise en compte des trajets non explorés.

A chaque itération, la mise à jour globale s'effectue comme ceci :

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \rho.\Delta\tau_{ij}^k(t) \qquad (2.7)$$

Où les arêtes (i, j) appartiennent au meilleur tour T^+ de longueur L^+ et où $\Delta\tau_{ij}^k(t)$ est égale à $1/L^+$ Ici, seule la meilleure piste est donc mise à jour, ce qui participe à une intensification par sélection de la meilleure solution.

3. Le système utilise une liste de candidats : Cette liste stocke pour chaque ville les v plus proches voisines, classées par distances croissantes. Une fourmi ne prendra en compte une arête vers une ville en dehors de la liste que si celle-ci a déjà été explorée.

Concrètement, si toutes les arêtes ont déjà été visitées dans la liste de candidats, le choix se fera en fonction de la règle (2.5), sinon c'est la plus proche des villes non visitées qui sera choisie.

C. ACO & 3-opt

Cette variante est une hybridation entre le ACS et une recherche locale de type 3-opt **[Dorigo 97a]**. Ici, la recherche locale est lancée pour améliorer les solutions trouvées par les fourmis (et donc les ramener à l'optimum local le plus proche).

D. Max- Min Ant System

Cette variante (notée MMAS) est fondée sur l'algorithme AS et présente quelques différences notables **[Stutzle 97]**, **[Stutzle 00]**:

1. Seule la meilleure fourmi met à jour une piste de phéromone.

2. Les valeurs des pistes sont bornées par τ_{min} et τ_{max}

3. Les pistes sont initialisées à la valeur maximum τ_{max} .

4. La mise à jour des pistes se fait de façon proportionnelle, les pistes les plus fortes étant moins renforcées que les plus faibles.

5. Une ré-initialisation des pistes peut être effectuée.

Les meilleurs résultats sont obtenus en mettant à jour la meilleure solution avec une fréquence de plus en plus forte au cours de l'exécution de l'algorithme.

2.3.5. Les essaims particulaires (Particle Swarm Optimization : *PSO*)

2.3.5.1. Origines

L'optimisation par essaim particulaire (OEP) est une méthode proposée en 1995 aux Etats Unis sous le nom de Particle Swarm Optimization (PSO) par Eberhart et Kennedy **[Eberhart 95]**.

Initialement, ses deux concepteurs, Russel Eberhart et James Kennedy, cherchaient à modéliser des interactions sociales entre des « agents » devant atteindre un objectif donné dans un espace de recherche commun, chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information. La règle de base était qu'il ne devait y avoir aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations, seulement des connaissances locales. Un modèle simple fut alors élaboré. Dès les premières simulations, le comportement collectif de ces agents évoquait celui d'un essaim d'êtres vivants convergeant parfois en plusieurs sous essaims vers des sites intéressants. Ce comportement se retrouve dans bien d'autres modèles, explicitement inspirés des systèmes naturels. Ici, la métaphore la plus pertinente est probablement celle de l'essaim d'abeilles, particulièrement du fait qu'une abeille ayant trouvé un site prometteur sait en informer certaines de ses consoeurs et que celles-ci vont tenir compte de cette information pour leur prochain déplacement. Finalement, le modèle s'est révélé être trop simple pour vraiment simuler un comportement social, mais par contre très efficace en tant qu'outil d'optimisation.

2.3.5.2. Description informelle

La version historique peut facilement être décrite en se plaçant du point de vue d'une particule. Au départ de l'algorithme, un essaim est réparti au hasard dans l'espace de recherche, chaque particule ayant également une vitesse aléatoire. Ensuite, à chaque pas de temps :

- chaque particule est capable d'évaluer la qualité de sa position et de garder en mémoire sa meilleure performance, c'est-à-dire la meilleure position qu'elle a atteinte jusqu'ici (qui peut en fait être parfois la position courante) et sa qualité (la valeur en cette position de la fonction à optimiser).

- chaque particule est capable d'interroger un certain nombre de ses voisins et d'obtenir de chacune d'entre elles sa propre meilleure performance (et la qualité afférente).
- A chaque pas de temps, chaque particule choisit la meilleure des meilleures performances dont elle a connaissance, modifie sa vitesse en fonction de cette information et de ses propres données et se déplace en conséquence **[Clerc 04]**.

2.3.5.3. Principales caractéristiques et principe de la version de base

Ce modèle présente quelques propriétés intéressantes, qui en font un bon outil pour de nombreux problèmes d'optimisation, particulièrement les problèmes fortement non linéaires, continus ou mixtes (certaines variables étant réelles et d'autres entières) :

- Il est facile à programmer, quelques lignes de code suffisent dans n'importe quel langage évolué,
- Il est robuste (de mauvais choix de paramètres dégradent les performances, mais n'empêchent pas d'obtenir une solution).

Comme c'était affirmé avant, l'OEP simule le comportement des groupes d'abeilles (ou d'oiseaux). Supposons le scénario suivant :

Un groupe d'oiseaux cherche aléatoirement la nourriture dans un champ. Il n'y a qu'une seule source de nourriture dans ce champ. Aucun des oiseaux ne connaît où se trouve exactement la nourriture, mais chacun sait à chaque fois quelle est la distance qui le sépare de la nourriture. A chaque essai, (i.e à chaque itération), la question qui se pose est la suivante : Quelle est la meilleure stratégie pour retrouver la nourriture ? La solution efficace est de suivre l'oiseau le plus proche de la nourriture.

L'OEP s'inspire de ce scénario pour résoudre les problèmes d'optimisation. Chaque oiseau est une solution possible dans l'espace de recherche. On l'appelle Particule. Chaque particule possède : une valeur de fitness (distance par rapport à la solution).

- une vélocité pour guider le mouvement de la particule et une position dans l'espace de recherche
- La vélocité et la position sont mise à jour suivant deux forces best local et global. La première l'attire au best local qui est la position qui a donné la meilleure fitness pour la particule (i.e c'est la position la plus proche de l'objectif que cette particule a pu atteindre). L'autre best est le "Global Best", c'est la meilleure position trouvée par la particule et ses voisins, comme suit :

$$V_i(t) = V_i(t-1) + c_1 r_1 (p_i + x_i(t-1)) + c_2 r_2 (p_g + x_i(t-1)) \quad (2.8)$$

$$X_i(t) = X_i(t-1) + V_i(t) \quad (2.9)$$

Où $v_i(t)$ est la vélocité de la particule i à l'itération t. $X_i(t)$ représente la position de la particule i à l'itération t. Le vecteur p_i est la mémoire de la particule i à la génération courante, et le vecteur p_g est la meilleure solution trouvée par l'essaim. Le coefficient cognitif c_1 et le coefficient social c_2 sont des constants positifs et les coefficients d'accélération r_1 et r_2 sont deux nombres aléatoires compris entre 0 et 1 suivent une distribution uniforme. Le pseudo algorithme de OEP est le suivant :

n : nombre d'individus

Initialiser la population

Initialiser les paramètres

Tant que (critère d'arrêt est non atteint)

Pour *i = 1* à *n* (pour chaque particule)

Evaluer fonction objective

Si $F(x_i) > F(P_i)$ alors : (mise à jour du best local)

$P_i=x_i$

Finsi

Si $F(x_i) > F(Pg)$ alors : (mise à jour du best global)

$Pg= x_i$

Finsi

Fin pour

Pour *i = 1* à *n*

On met a jour la particule à l'aide des formules (2.8) et (2.9)

Fin pour

Fin tant que **[Eberhart 95].**

À l'origine, PSO a été conçu pour résoudre les problèmes d'optimisation continus, mais elle peut être adapté aux problèmes discrets comme notre problème de sélection de routage, où (2.8) et (2.9) ont été remplacés par l'équation suivante proposée par Pan et al **[Pan 05]** :

$$X_i(t)=c_2\oplus F_3(c_1\oplus F_2(w\oplus F_1(X_i(t-1)),p_i(t-1)),G(t-1)) \qquad (2.10)$$

L'équation (2.10) se compose de trois composants: Le premier composant est $\lambda_i(t)=w\oplus F_1(X_i(t-1))$ qui représente la vélocité de la particule. F_1 représente un opérateur qui modifie la particule avec une probabilité *w*, un nombre aléatoire uniforme *r* est généré entre 0 et 1. Si *r* est inférieur à *W* alors F_1 est appliqué pour produire une permutation perturbée de la particule par $\lambda_i(t)=F_1(X_i(t-1))$, sinon la permutation courante est gardée $\lambda_i(t)=X_i(t-1)$. De la même manière, le deuxième composant qui est la partie cognitive de la particule $\delta_i(t)=c_1\oplus F_2(\lambda_i(t),p_i(t-1))$ et le troisième composant qui est la partie sociale de la particule $X_i(t)=c_2\oplus F_3(\delta_i(t),G_i(t-1))$ ont été modifie avec F_2 et F_3 représentent les croisements avec les probabilités C_1 et C_2.

2.3.6. L'électromagnétisme (electromagnetism like method : *EM*)

Electromagnétisme métaheuristique a été proposé par Birbil et Fang en 2003 **[Birbil 03]** pour résoudre les problèmes d'optimisation difficile continus efficacement. Elle a été inspirée d'une analogie du mécanisme attraction-répulsion de la théorie d'électromagnétisme, elle est une métaheuristique basée sur une population.

Dans cette approche la charge de chaque point est relative à la valeur de la fonction objective que nous essayons d'optimiser. Cette charge détermine également l'attraction ou la répulsion du point.

D'ailleurs, la force électrostatique entre deux points est directement proportionnelle aux charges de ces points et inversement proportionnelle au carré de la distance entre ces points. La charge fixe de chaque particule est donnée comme suit :

$$q_i = \exp\left(-n * \left(f(x_i) - f(x_{best}) / \left(\sum_{k=1}^{m} \left(f(x_k) - f(x_{best})\right)\right)\right)\right), \forall i \qquad (2.11)$$

Où q_i est la charge de particule i, $f(x_i)$, $f(x_{best})$, et $f(x_k)$ sont les valeurs des fonctions objectives des particules i, la meilleure particule, et la particule k. m est la taille de la population et n la dimension du problème.

La qualité de solution ou la charge de chaque particule détermine l'effet d'attraction et de répulsion dans la population. Une meilleure solution encourage d'autres particules à converger aux vallées attrayantes tandis qu'une mauvaise solution décourage les autres particules pour se déplacer vers cette région.

Ces particules se déplacent avec toute la force et ainsi des solutions diversifiées sont produites. La formulation suivante donne la force de la particule i.

$$F_i = \sum_{j \neq i}^{m} \begin{Bmatrix} (x_j - x_i) * \dfrac{(q_i * q_j)}{\|x_j - x_i\|^2} : f(x_j) < f(x_i) \\ (x_i - x_j) * \dfrac{(q_i * q_j)}{\|x_j - x_i\|^2} : f(x_j) \geq f(x_i) \end{Bmatrix}, \forall i \qquad (2.12)$$

Le générique pseudo-code de cette métaheuristique est le suivant:

Initialiser les points

Tant que (critère d'arrêt non atteint) faire

Effectuer une recherche local pour chaque particule.

Calculer la force total *F ()* de chaque particule.

Déplacer la particule par *F ()*.

Evaluer les particules.

Fin Tant que

Dans cet algorithme, il y' a quatre procédures : l'initialisation, la recherche locale, le calcul de la force totale et le déplacement par rapport à la force totale.

L'initialisation est utilisée pour distribuer les m points uniformément dans une région faisable. La procédure suivante est d'effectuer une recherche locale autour d'un point pour avoir de bonnes solutions. Les procédures du EM algorithme sont : calcul de la force totale et le déplacement par rapport à la force total, la première procédure est utilisée pour calculer la force totale de chaque

point (formule 2.12), et l'autre consiste à déplacer le particule (point) selon la direction de la force total de ce point. Le pseudo code de la procédure de déplacement est le suivant **[Birbil 03]**:

Pour *i = 1* à *m* **faire**

Si $i \neq$ best **alors**

$\lambda = U(0, 1)$

$F_i = F_i / \|F_i\|$

Pour *k = 1* à *n* **faire**

Si $F_k^i \succ 0$ **alors**

$x_k^i = x_k^i + \lambda . F_k^i . \left(u_k - x_k^i\right)$

Sinon

$x_k^i = x_k^i + \lambda . F_k^i . \left(x_k^i - l_k\right)$

Fin si

Fin pour

Où λ est un nombre compris entre 0 et 1 généré par une distribution uniforme, l_k et u_k sont les limites minimale et maximale de déplacement, n est la dimension du problème, x et F sont les vecteurs position et force.

Cette métaheuristique peut être adapté aux cas non continus comme notre problème de sélection de routage où nous avons utilisé un framework qui inclut les opérations des algorithmes génétiques (croisement) et un EM algorithme modifié proposé par Chen et al en 2007 **[Chen 07].**

Cet algorithme hybride commence à déterminer quelle particule sera déplacée ou modifiée par les opérations de croisement. L'algorithme suivant est le pseudo code de la procédure principale du framework hybride :

Génération d'une population aléatoire.

Tant que (critère d'arrêt est non atteint)

Recherche_local ();

Evaluation de la fonction objective de chaque individu

avg =calcAvgObjectiveValues() (calcul de moyen des fonctions objectifs).

Pour *i = 1 à m*

Si $i \neq best$ et $f(x_i) < avg$ alors

j = particule sélectionné

croisement_uniforme (x_i, x_j)

Finsi

Sinon Si $f(x_i) > avg$ Alors

Calcul force et déplace (x_i)

Finsi

Fin pour

Evaluation de la fonction objective de chaque individu

Fin tant que

Selon l'algorithme précédent, nous initialisons les particules dans une population. Puis le procédé de recherche locale est mis en application avant les procédures de EM et les opérations génétiques. Pour déterminer quelle solution est bonne ou mauvaise, nous calculons une valeur objective moyenne *avg*, si la solution est meilleure que *avg* alors elle va subir un croisement avec une autre solution, sinon elle sera déplacée dans la direction de sa force.

La force et la charge électrostatique sont calculées par les formules suivantes proposées Chen et al en 2007 **[Chen 07].**

$$q_{ij} = \frac{f(x_i) - f(x_j)}{f(x_{worst}) - f(x_{best})} \qquad (2.13)$$

Où q_{ij} est la charge entre les particules i et j, $f(x_i)$, $f(x_j)$, $f(x_{best})$, $f(x_{worst})$ sont les valeurs des fonctions objectives des particules i et j, la meilleure solution, et la plus mauvaise solution.

Si la fonction $f(x_i)$ est supérieure à $f(x_j)$, la particule j attire la particule i, sinon si $f(x_i) < f(x_j)$, un effet de répulsion apparaît, il n y'a aucune action si $f(x_i) = f(x_j)$ car $q_{ij} = 0$, la force F_{ij} est calculée comme suit :

$$F_{ij} = (x_i - x_j) * q_{ij} \qquad (2.14)$$

2.4. Exemples d'applications

La méthode du recuit simulé s'est avérée intéressante dans la résolution de nombreux problèmes d'optimisation, NP difficile tel que les problèmes modèles d'optimisation combinatoires comme le problème du voyageur de commerce, les problèmes de partitionnement des graphes, le problème du couplage minimal de points, l'affectation quadratique et d'autre problèmes pratiques qui ont un très grand intérêt industriel surtout en électronique comme le problème de placement des circuits électroniques.

La recherche tabou a prouvé son efficacité quand elle a été appliqué sur de nombreux problèmes combinatoires comme l'affectation quadratique.

Les colonies de fourmis ont pu retenir des performances particulièrement intéressantes dans le cas d'affectation quadratique, des problèmes de planification, de l'ordonnancement séquentiel, du routage de véhicules, ou du routage sur le réseau… **[Dréo 03]**

Les algorithmes évolutionnaires ont été appliqués à différents problèmes issus de la recherche opérationnelle : la coloration de graphes, le problème du voyageur de commerce, la gestion d'emploi du temps ...

On trouve aussi des applications en robotique pour la recherche de trajectoire optimale ou l'évitement d'obstacles, en vision pour la détection ou la reconnaissance d'images, en recalage d'images médicales, en reconnaissance de formes, mais aussi en mécanique pour l'optimisation de formes, en chimie, économie et gestion de production, gestion de trafic aérien ... **[Spalanzani 99]**

Les essaims particulaires ont été appliqués sur de nombreux problèmes comme la restauration supervisée d'image, le problème de l'apprentissage des Modèles de Markov Cachés, problème des n- reines….

Enfin, l'électromagnétisme a été appliqué sur de nombreux types de problèmes comme l'ordonnancement de projets, les problèmes d'ordonnancement de type flow shop et la résolution des équations de relations floues. **[Chen 07]**

2.5. Conclusion

Ce chapitre a décrit des généralités sur les métaheuristiques et a montré leurs multiples facettes proposées depuis 20 ans pour la résolution approché des problèmes d'optimisation difficile. Le succès de la démarche ne doit pas masquer la principale difficulté à laquelle est confronté l'utilisateur, en présence d'un problème d'optimisation concret : celui du choix d'une méthode "efficace", capable de produire une solution "optimale" ou de qualité acceptable au prix d'un temps de calcul "raisonnable" et une méthode efficace pour un problème peut donner de mauvais résultats pour un autre.

Face à ce souci, la théorie n'est pas encore d'un grand secours, car les théorèmes de convergence sont souvent inexistants ou applicables seulement sous des hypothèses très restrictives. En outre, le réglage "optimal" des divers paramètres d'une métaheuristique, qui peut être préconisé par la théorie, est souvent inapplicable en pratique, car il induit un coût de calcul prohibitif. En conséquence, le choix d'une bonne méthode, et le réglage des paramètres de celle-ci, font généralement appel au savoir faire et à l'expérience de l'utilisateur, plutôt qu'à l'application fidèle de règles bien établies.

Pour notre problème de sélection de routage, nous somme intéressé à ces techniques mais on ne peut pas savoir à l'avance quel est la technique la plus efficace pour ce problème.

C'est justement le but des chapitres suivants où nous avons appliqué ces techniques à notre problème, et comme base de comparaisons nous avons utilisé les règles DMM et DMM modifiée dont l'efficacité a été démontré dans des travaux précédents **[Saygin 01,04]**, **[Hassam 07]** et **[Ghomri 07]** où ils ont dépassé les autres règles de sélection de routage.

Chapitre 3

Métaheuristiques pour la sélection de routages alternatifs en temps réel -sans ré ordonnancement de la station de chargement-

3.1. Introduction

Avec l'ordonnancement en temps différé on trouve beaucoup de problèmes à cause des changements inévitables, la définition d'un ordonnancement prévisionnel limite les capacités d'un FMS et ne permet pas de répondre aux attentes des responsables d'atelier flexible car on ne peut pas :

- s'adapter aux changements d'états de l'atelier, c'est-à-dire prendre en compte tous les changements d'état qui affectent le plan de production prévu (aléas ou changements d'état attendus),
- surveiller le déroulement de l'exécution des opérations, afin de pouvoir détecter les situations anormales, c'est-à-dire les situations où il apparaît une dérive entre les réalisations de l'atelier et les objectifs de production.

Les facteurs cités ci-dessus et beaucoup d'autres, rendent le ré-ordonnancement obligatoire afin d'éviter l'augmentation des temps d'attente, l'augmentation des en-cours, la faible utilisation des machines et des équipements et éventuellement la dégradation des performances du système de production **[Wu 89]**, **[Ishii 96]**.

Le contrôle et l'ordonnancement temps réel des systèmes flexibles de production 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 **[Saygin 95]**, **[Peng 98]**. 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 **[Byrne 97]**, **[Kazerooni 97]** 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 **[Das 97]**, **[Cho 95]**.

Mais, cette approche n'est pas applicable pour les systèmes flexibles de production aléatoires **[Rachamadugu 94]**, 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 **[Rachamadugu 94]**. 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 **[Mamalis 95]**, **[Rachamadugu 94]**.

Le manque de méthodes d'ordonnancement des FMS en temps réel, qui utilisent la flexibilité d'opérations et de routage sont la force motrice de la règle de sélection de routages alternatifs en temps réel DMM et DMM modifiée **[Hassam 07]**.

L'objective de ce chapitre est de comparer les performances des métaheuristiques étudiées (les colonies de fourmis, les algorithmes génétiques, le recuit simulé, la recherche tabou, les essaims particulaires, électromagnétisme métaheuristique) adaptées pour résoudre le problème en temps réel (sans ré-ordonnancement de la station de chargement) avec les règles DMM et DMM modifiée.

Dans ce chapitre nous allons présenter le modèle FMS que nous allons étudier, les algorithmes des métaheuristiques adaptées à la résolution de ce problème en temps réel ainsi que les résultats obtenus et leurs interprétations après plusieurs simulations du modèle sur un temps de simulation de 20000 heures et un régime transitoire de 3000 heures, avec 10 réplications (essais) pour chaque simulation.

Les résultats des règles DMM et DMM modifiée présentés dans ce travail sont ceux trouvés par Hassam **[Hassam 06]**

3.2. La présentation du modèle FMS étudiée

Pour faire la comparaison entre les deux règles DMM, DMM modifiée et les métaheuristiques, nous avons étudié et simulé un système flexible de production.

Ce système contient sept machines et deux stations : une station de chargement et une de déchargement. Six types de pièces différentes sont traités dans le système.

Les machines et les stations qui composent le système étudié sont 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).

Chaque machine comporte une file d'attente d'entrée et une file d'attente de sortie, la station de chargement contient aussi une file d'attente d'entrée.

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

- Les routages alternatifs de chaque type de pièce sont connus avant le début de la production.
- Le temps de traitement 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 peut traiter une seule pièce à la fois.

Les routages alternatifs et les temps de traitement de chaque type de pièce sont donnés dans le tableau 3.1

Type de parts	Taux d'arrivée	Routage 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_2 (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_1 (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_1 (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_1 (25) – TP (1) – T_2 (35) – FH_2 (50) – SD
F	8 %	SC – FH_1 (40) – SD
		SC – FH_2 (40) – SD

Tableau 3.1 : Routages alternatifs et temps de traitement des pièces [Saygin 01].

La configuration du système flexible de production est donnée dans la figure suivante :

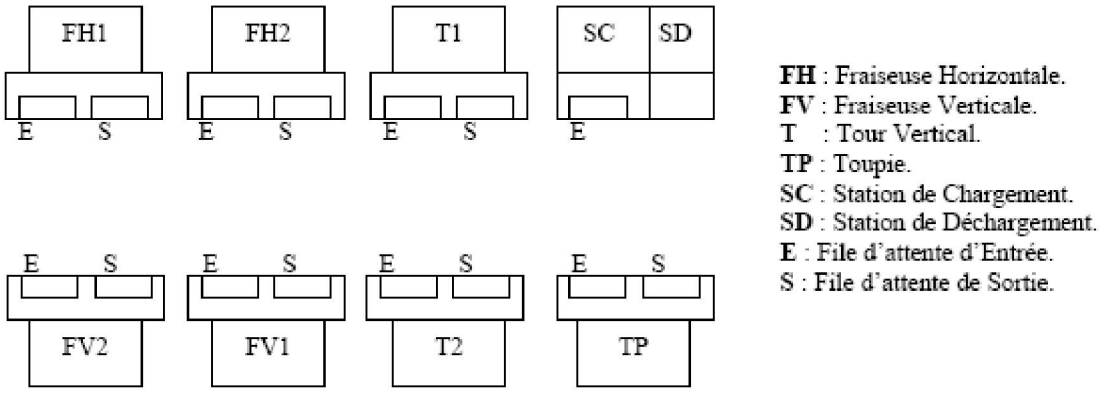

Figure 3.1 : Configuration du modèle FMS étudié [Saygin 01].

3.3. Les algorithmes des métaheuristiques

Toutes les métaheuristiques présentent des algorithmes itératifs, donc c'est très difficile de trouver la solution optimale à partir de la première itération ce qui justifie l'échec d'adaptation de ces techniques pour la résolution des problèmes complexes en ligne.

Pour 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. Pour DMM, et DMM modifiée la dissimilitude entre les routages occupés a été maximisée, à l'aide des métaheuristiques nous avons essayé d'équilibrer les charges de routages non pas en terme de nombre de pièces mais en terme de temps opératoire (maximiser le produit des charges de routages). Pour ce faire nous avons pris en considération les types des premières pièces existantes (le nombre de ces pièces égale à la taille de la station de chargement) dans la file infinie (avant qu'ils passent à la station de chargement) en essayant d'affecter chaque pièce à un routage selon son type. Les routages de ces pièces sont modifiés à chaque itération selon le principe de chaque métaheuristique et pour chaque routage nous avons calculé sa charge en vue de maximiser leur produit.

3.3.1. Le recuit simulé (Simulated Annealing : *SA*)

n est la capacité des files d'attente.

S'il y`a une place libre dans la station de chargement alors à partir d'un état initial (les *n* premières pièces sont affecté à certains routages selon leurs types d'une façon aléatoire) à une température *T*, on génère une autre solution par une modification élémentaire d'une manière aléatoire (on modifie les routages de certaines pièces), puis on calcule le produit des charges de routages. Si cette solution améliore la fonction objectif (le produit des charges de routages), cette dernière est automatiquement acceptée. Si elle dégrade la fonction objectif elle peut être acceptée avec une probabilité $exp(-\Delta E)$ où ΔE est la variation de la fonction objectif, puis on abaisse la température légèrement avant d'effectuer une nouvelle itération.

Le pseudo- code de l'algorithme est :

S'il y`a une place libre dans la station de chargement alors

Construire l'état initial (Affecter les *n* premières pièces à des routages d'une façon aléatoire).

Calculer le produit des charges de routages.

Pour $t = 1$ à t_{max}

Modifier les routages de certaines pièces parmi les *n* premières pièces contenues dans la file infinie.

Calculer le produit des charges de routages.

Si la fonction objectif a été améliorée alors cette solution est acceptée

Finsi

Sinon générer un nombre aléatoire

Si ce nombre est inférieur ou égale à $exp(-\Delta E)$: ΔE est la variation de la fonction objectif alors cette solution est acceptée

Finsi

Finsi

Fin Pour.

Finsi

3.3.2. Recherche avec tabous (Taboo Search : *TS*)

n est la capacité des files d'attente.

Le pseudo- code de l'algorithme est :

S'il y`a une place libre dans la station de chargement alors

Construire l'état initial *S*

Initialiser les paramètres.

Calculer les produits des charges de routages.

Tant que (critère d'arrêt est non atteint)

Pour $t = 1$ à nbre_voisins

Modifier les routages de certaines pièces choisies aléatoirement parmi les *n* premières pièces de la file infinie avec des mouvements non tabous (modification de *S*).

Calculer le produit des charges de routages.

Si la fonction objectif (produit des charges de routages) est supérieure à la meilleure solution, alors mettre à jour la meilleure solution (L'élitisme).

Finsi

Fin pour

Soit *T* le meilleur de ces voisins.

Insertion de mouvements *T->S* dans la liste tabou.

S = T.

Fin tant que

Finsi.

3.3.3. Les algorithmes génétiques (Genetic Algorithms : *GA*)

n est la capacité des files d'attente.

Dans cet algorithme, chaque chromosome artificiel représente les routages choisis des premières pièces par un code. Après chaque évaluation des individus on modifie la meilleure solution s'il y' a une amélioration.

La mutation consiste à modifier les routages de certaines pièces parmi les *n* premières pièces contenues dans la file infinie.

Le croisement est exécuté en prenant deux génotypes, puis on choisi un endroit le long de la chaîne, on coupe chacun d'eux à cet endroit et on relie la partie droite d'une chaîne à la partie gauche de l'autre et vice versa.

Le pseudo- code de l'algorithme est :

S'il y`a une place libre dans la station de chargement alors

Génération d'une population aléatoire.

Tant que (critère d'arrêt est non atteint)

Pour chaque individu

Evaluation de la fitness de cet individu (le produit des charges de routages).

Si la fonction objectif est supérieure à la meilleure solution alors mettre à jour la meilleure solution.

Fin pour

Sélection des individus pour la reproduction (opérateur de sélection).

Application de l'opérateur de croisement (on obtient un ensemble de nouveaux individus).

Application de l'opérateur de mutation sur les nouveaux individus.

Constitution de la nouvelle génération.

Fin tant que

Finsi

3.3.4. Les colonies de fourmis (Ant Colony Optimization : *ACO*)

L'algorithme original a été adapté à notre problème en remplaçant la ville *i* par la pièce *i* et la ville *j* par le routage *j*.

Pour chaque pièce *i*, le choix du routage *j* est basé sur un compromis entre l'intensité de la trace $\tau_{ij}^{k}(t)$ et la visibilité η_{ij} (dépend du nombre de pièces dans la file d'entrée de la première machine de ce routage et sa charge).

L'importance relative des deux éléments est toujours contrôlée par deux coefficients α et β.

Si le nombre total de fourmis est *m* et la taille de la station de chargement est *n*, un cycle est réalisé lorsque chacune des *m* fourmis affecte les *n* premières pièces de la file infini à des routages *j*.

Après un tour complet (l'affectation de toutes les *n* premières pièces de la file infini aux routages par les fourmis), chaque fourmi laisse une certaine quantité de phéromone $\Delta\tau_{ij}^{k}(t)$ qui dépend de la qualité de la solution trouvée (le produit des charge de routages) sur l'ensemble des routages sélectionnés pour les pièces.

Le pseudo- code de l'algorithme est :

S'il y`a une place libre dans la station de chargement alors

Pour $t = 1$ à t_{max}

Pour chaque fourmi $k = 1$ à m

Choisir pour la première pièce de la file infini un routage au hasard suivant son type.

Pour chaque pièce *i* contenu dans la deuxième place jusqu' au la n^{eme} place de la file infini.

Choisir un routage *j*, parmi les routages possibles selon une probabilité dépend de l'intensité de la trace et du nombre de pièces dans la file d'entrée de la première machine de ce routage et sa charge.

Fin Pour

Evaluation de la fonction objectif. (Produit des charges de routages)

Déposer une piste $\Delta\tau_{ij}^{k}(t)$ sur le trajet $T^k(t)$ (pour chaque routage *j* choisi pour la pièce *i* par la fourmis *k*).

Fin Pour.

Évaporer les pistes et modifier les intensités.

Fin Pour.

Finsi.

3.3.5. Les essaims particulaires (Particle Swarm Optimization : *PSO*)

Bl_i , et B_g sont les bests locaux et globaux.

n_1 : nombre d'individus

n est la taille des files d'attente.

Le pseudo code de l'algorithme est le suivant :

S'il y`a une place libre dans la station de chargement alors

Initialiser la population.

Initialiser les paramètres.

Tant que (critère d'arrêt est non atteint)

Pour $i = 1$ à n_1 (pour chaque particule X_i)

Calculer le produit des charges de routages de cette particule.

Si $F(x_i) > F(Bl_i)$ alors : (mise à jour du best local)

$Bl_i=x_i$

Finsi

Si $F(x_i) > F(B_g)$ alors : (mise à jour du best global)

Bg= x_i

Finsi

Fin pour

Pour $i = 1$ à n_1 (on met a jour le particule)

$$X_i(t) = c_2 \oplus F_3\left(c_1 \oplus F_2\left(w \oplus F_1\left(X_i(t-1)\right), Bl_i(t-1)\right), B_g(t-1)\right)$$

Fin pour

Fin tant que

Finsi.

F_1 représente un opérateur qui modifie les routages de quelques pièces parmi les premières pièces de la file infinie avec une probabilité w, un nombre aléatoire uniforme r est généré entre 0 et 1. Si r est inférieur à W alors F_1 est appliqué pour produire une permutation perturbée. De la même manière, on applique F_2 et F_3 qui représentent les croisements avec les bests locaux et globaux selon des probabilités C_1 et C_2.

3.3.6. L'électromagnétisme (electromagnetism like method : *EM*)

Le pseudo- code de l'algorithme est :

m : la taille de la population.

S'il y`a une place libre dans la station de chargement alors

Génération d'une population aléatoire.

Tant que (critère d'arrêt est non atteint)

Pour $i = 1$ à m (pour chaque particule X_i)

Recherche_locale en modifiant les routages de certains pièces parmi les n premières de la file infini.

Evaluation de la fonction objectif (Produit des charges de routages).

Si la fonction objectif est supérieure à la meilleure solution alors mettre à jour la meilleure solution.

Fin pour

avg = calcAvgObjectiveValues() (calcul de moyen des fonctions objectifs).

Pour $i = 1$ à m

Si $i \neq best$ et $f(x_i) > avg$ alors

j = particule sélectionné.

croisement_uniforme (x_i, x_j).

Finsi

Sinon Si $f(x_i) < avg$ Alors

Calcul force et déplace (x_i).

Finsi

Fin pour

Fin tant que

Finsi.

3.4. Résultats et interprétations (sans ré-ordonnancement)

Afin de montrer les améliorations apportées par les métaheuristiques étudiées et adaptées pour la sélection de routages alternatifs en temps réel, nous avons fait plusieurs études en simulation avec des variations sur les critères du système étudié. Le but de notre travail est d'évaluer les performances des métaheuristiques lorsque le système est en état de saturation.

C'est pour ça que les résultats que nous allons présenter reposent sur deux critères principaux :

- Le taux de création (arrivée) des pièces.
- La taille des files d'attente d'entrée et de sortie des stations.

Dans cette section nous avons montré les résultats de simulation des deux méthodes (DMM et DMM modifiée), l'exécution des programmes des métaheuristiques ainsi que les interprétations des résultats obtenus.

3.4.1. Etude comparative sans introduction de pannes

3.4.1.1. Taux de production

Le taux de production ou de sortie des pièces est calculé en divisant le nombre de pièces sorties du système sur le nombre de pièces arrivées à la file infinie, afin de faire une normalisation.

Les graphes des figures 3.2 à 3.6 présentent le taux de sortie des pièces du système en fonction du taux de création de pièces et pour différentes capacités de files d'attente.

Les figures 3.6 et 3.7 présentent le taux de sortie des pièces du système en fonction des capacités de files d'attente et en maintenant le temps de création fixe.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.99	90.52	64.54	43.2	21.51
Le recuit simulé	99.99	99.99	99.99	99.99	94.73	61.9	41.21	20.61
Les essaims particulaires	99.99	99.99	99.99	99.99	99.22	69.33	46.1	23.12
Les algorithmes génétiques	99.99	99.99	99.99	99.99	98.44	71.08	47.25	23.7
Recherche avec tabous	99.99	99.99	99.99	99.99	95.28	64.93	43.29	21.78
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	98.52	66.94	44.75	22.42
DMM modifiée	99.99	99.99	99.98	99.71	84.47	60.73	41.67	21.15
DMM	99.99	99.99	99.97	81.4	24.65	32.05	15.43	8.88

Tableau 3.2 : Taux de sortie des pièces pour une capacité de file d'attente = 2.

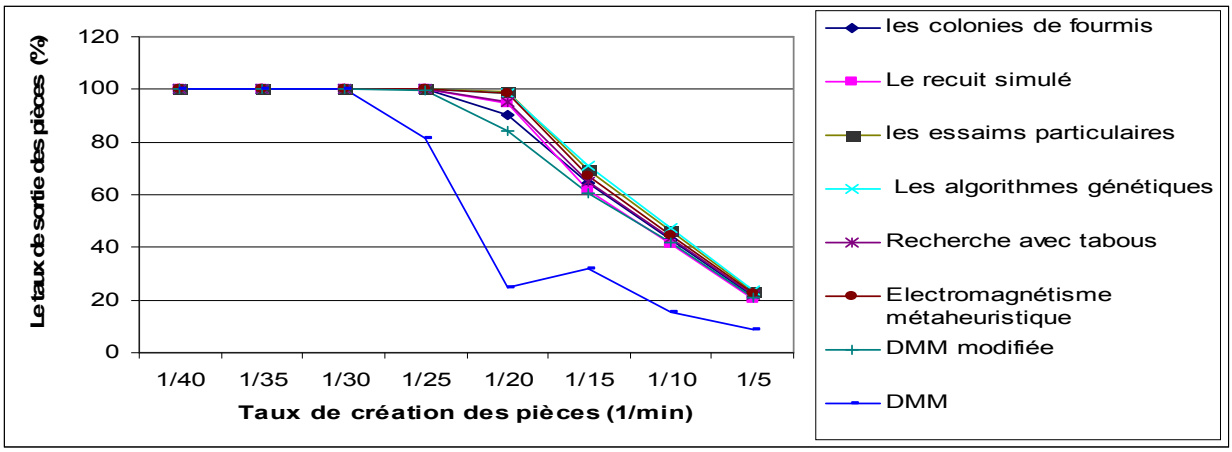

Figure 3.2 : Taux de sortie des pièces pour une capacité de file d'attente = 2.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.99	99.99	68.78	45.73	22.85
Le recuit simulé	99.99	99.99	99.99	99.99	99.99	64.69	43.44	21.71
Les essaims particulaires	99.99	99.99	99.99	99.99	99.99	70.29	46.92	23.4
Les algorithmes génétiques	99.99	99.99	99.99	99.99	99.99	72.06	47.88	23.93
Recherche avec tabous	99.99	99.99	99.99	99.99	99.99	69.65	46.38	23.19
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	99.99	69.08	46.03	23.02
DMM modifiée	99.99	99.99	99.99	99.98	90.61	67.79	45.42	22.58
DMM	99.99	99.99	99.99	99.98	57.84	34.11	33.03	15.67

Tableau 3.3 : Taux de sortie des pièces pour une capacité de file d'attente = 4.

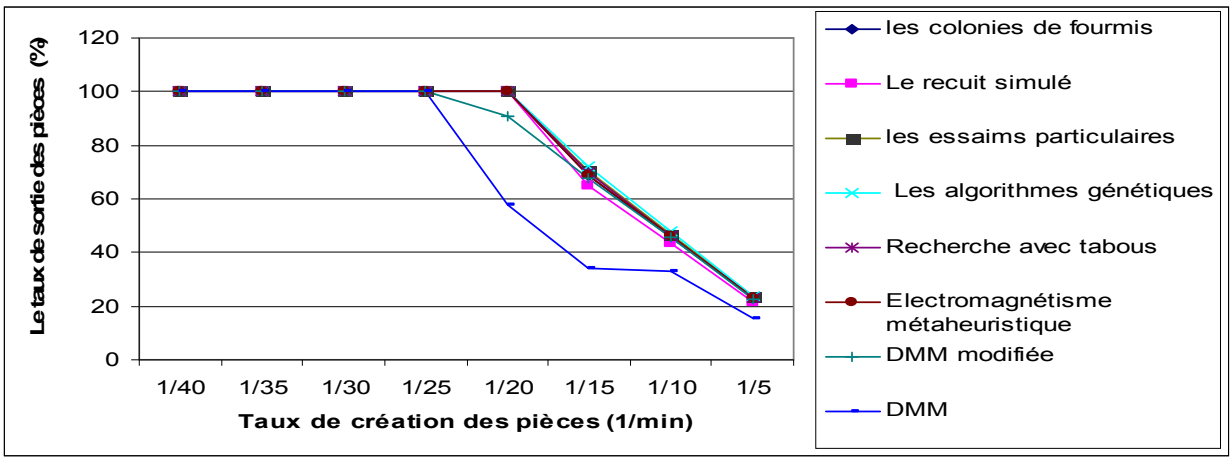

Figure 3.3 : Taux de sortie des pièces pour une capacité de file d'attente = 4.

Les courbes des figures et les tableaux 3.2 et 3.3 montrent que pour de petites files d'attente et un taux de création de pièces supérieur ou égale à 1/30, les résultats obtenus par les métaheuristiques étudiées sont meilleurs que celles des méthodes DMM et DMM modifiée (sauf le recuit simulé) et qu'en dessous du taux de création 1/30 le taux de production est pratiquement le même pour les deux méthodes et les métaheuristiques.

Ces courbes montrent que pour de petites capacités de files d'attente et un taux de création de pièces supérieur à 1/20 que les algorithmes génétiques sont plus efficaces que les autres métaheuristiques et les méthodes DMM modifiée et DMM, et pour un taux de création égale à 1/20

les essaims particulaires sont les meilleurs. Et pour un taux de création inférieur à 1/20, les résultats obtenus par les métaheuristiques sont les mêmes.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.99	99.99	68.97	45.88	22.98
Le recuit simulé	99.99	99.99	99.99	99.99	99.99	67.48	45.17	22.7
Les essaims particulaires	99.99	99.99	99.99	99.99	99.99	71.15	47.47	23.78
Les algorithmes génétiques	99.99	99.99	99.99	99.99	99.99	72.41	48.26	24.17
Recherche avec tabous	99.99	99.99	99.99	99.99	99.99	71.36	47.6	23.8
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	99.99	69.13	46.23	23.16
DMM modifiée	99.99	99.99	99.99	99.99	93.11	69.79	46.77	23.44
DMM	99.99	99.99	99.99	99.99	91.94	53.33	41.3	18.42

Tableau 3.4 : Taux de sortie des pièces pour une capacité de file d'attente = 6.

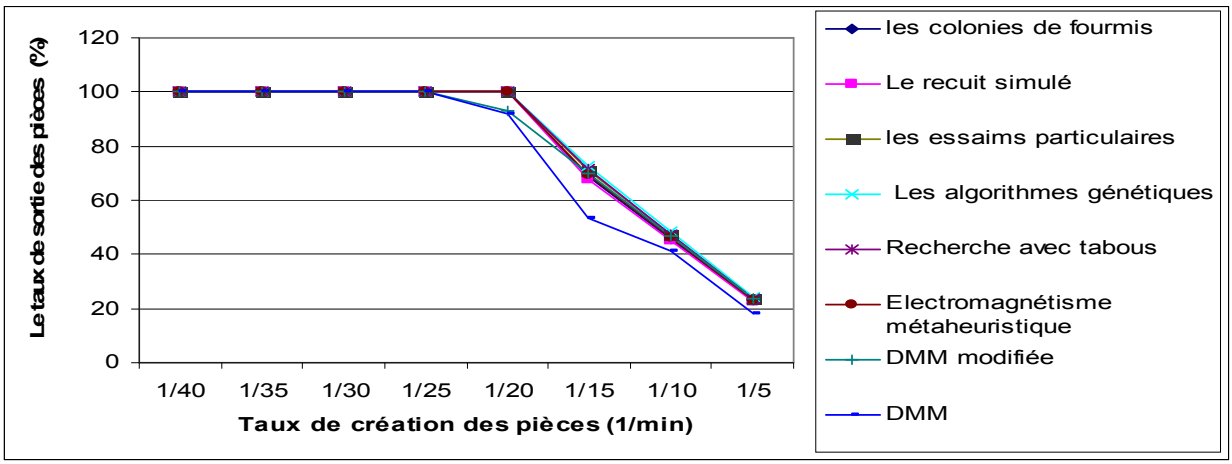

Figure 3.4 : Taux de sortie des pièces pour une capacité de file d'attente = 6.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.99	99.99	69.07	46.29	23.13
Le recuit simulé	99.99	99.99	99.99	99.99	99.99	69.82	46.67	23.39
Les essaims particulaires	99.99	99.99	99.99	99.99	99.99	71.91	47.99	24.07
Les algorithmes génétiques	99.99	99.99	99.99	99.99	99.99	72.76	48.59	24.24
Recherche avec tabous	99.99	99.99	99.99	99.99	99.99	72.24	48.13	24.07
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	99.99	70.65	47.21	23.69
DMM modifiée	99.99	99.99	99.99	99.99	94.25	70.83	47.32	23.51
DMM	99.99	99.99	99.99	99.99	95.71	71.35	47.3	24.01

Tableau 3.5 : Taux de sortie des pièces pour une capacité de file d'attente = 8.

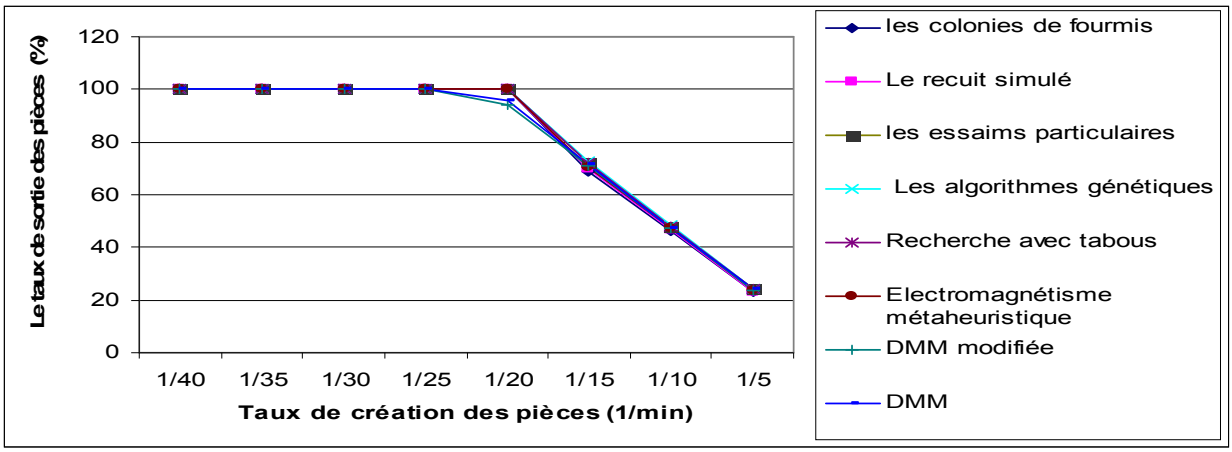

Figure 3.5 : Taux de sortie des pièces pour une capacité de file d'attente = 8.

Les courbes des figures et les tableaux 3.4 et 3.5 montrent que pour de grandes capacités de files d'attente et un taux de création de pièces supérieur à 1/20, les algorithmes génétiques sont toujours les meilleurs. Et hors de cet intervalle, les taux de production obtenus par les métaheuristiques sont identiques.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	21.51	22.85	22.98	23.13
Le recuit simulé	20.61	21.71	22.7	23.39
Les essaims particulaires	23.12	23.4	23.78	24.07
Les algorithmes génétiques	23.7	23.93	24.17	24.24
Recherche avec tabous	21.78	23.19	23.8	24.07
Electromagnétisme métaheuristique	22.42	23.02	23.16	23.69
DMM modifiée	21.15	22.58	23.44	23.51
DMM	8.88	15.67	18.42	24.01

Tableau 3.6 : Taux de sortie des pièces pour un taux de création de pièces = 1/5 (1/min).

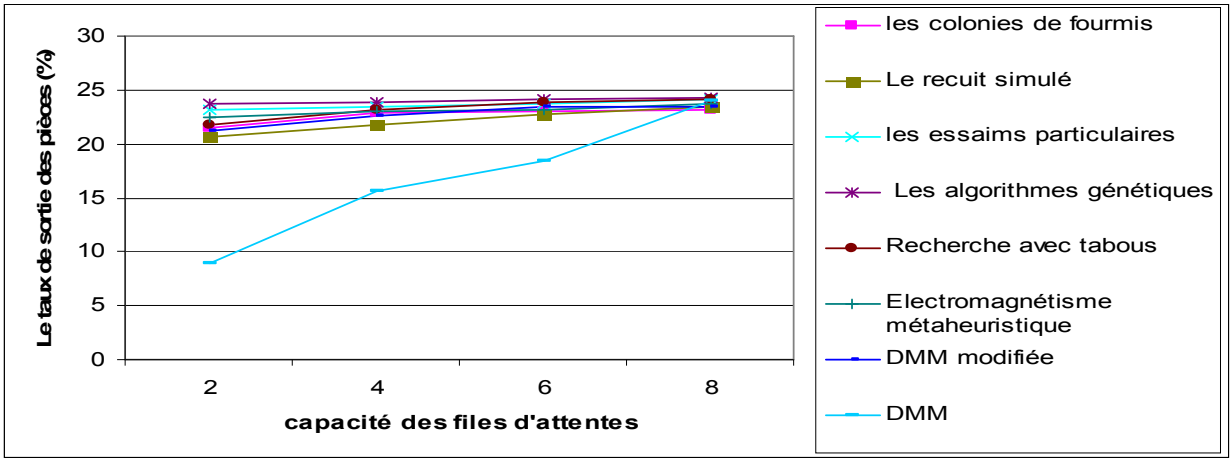

Figure 3.6 : Taux de sortie des pièces pour un taux de création de pièces = 1/5 (1/min).

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	90.52	99.99	99.99	99.99
Le recuit simulé	94.73	99.99	99.99	99.99
Les essaims particulaires	99.22	99.99	99.99	99.99
Les algorithmes génétiques	98.44	99.99	99.99	99.99
Recherche avec tabous	95.28	99.99	99.99	99.99
Electromagnétisme métaheuristique	98.52	99.99	99.99	99.99
DMM modifiée	84.47	90.61	93.11	94.25
DMM	24.65	57.84	91.94	95.71

Tableau 3.7: Taux de sortie des pièces pour un taux de création de pièces = 1/20 (1/min).

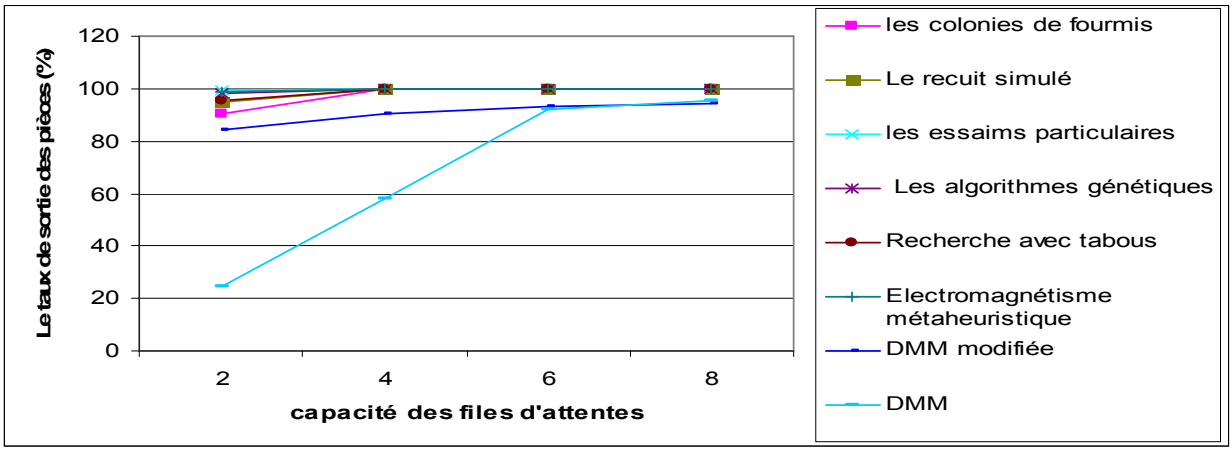

Figure 3.7: Taux de sortie des pièces pour un taux de création de pièces = 1/20 (1/min).

Les figures 3.6 et 3.7 nous montrent qu'en variant les capacités de files d'attente et en maintenant le temps de création fixe, les algorithmes génétiques sont les meilleurs pour un système très saturé. Pour un système moins saturé (Un taux de création de pièces égale à 1/20) les essaims particulaires sont les plus efficaces pour une taille de files d'attente égale à 2 et si la taille des files d'attente augmente les résultats obtenus par les métaheuristiques sont les mêmes.

A partir des résultats montrés sur la figure et tableau 3.7, nous pouvons remarquer que le système n'est pas saturé pour les métaheuristiques et saturé pour les règles DMM et DMM modifiée pour un taux de création de pièces égale à 1/20 ce qui veut dire que le système sature moins rapidement si on utilise les métaheuristiques.

3.4.1.2. Le temps de cycle

Le temps de cycle d'une pièce est le temps de présence d'une pièce dans le système depuis qu'elle entre dans la station de chargement jusqu'elle ce qu'elle quitte la station de déchargement. On s'intéresse donc au temps de cycle moyen (par min) de toutes les pièces sortantes du système.

Dans cette sous section nous allons montrer l'impact du taux de création de pièces, et la capacité des files d'attente sur le temps de cycle.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	90.9	83.7	89	98.8	155.2	166.6	165.4	163.5
Le recuit simulé	95.3	83.8	90.3	99.5	135.5	170.4	168.9	168.1
Les essaims particulaires	89.6	81.2	91.7	96.1	125.9	173.5	173.2	169.4
Les algorithmes génétiques	89.6	81.8	88.6	98.7	132.7	169.8	168.9	168.1
Recherche avec tabous	91.4	80.3	86.2	98	135.9	171.7	174.8	173.9
Electromagnétisme métaheuristique	92.2	81.6	90.1	101.1	124.8	170.5	170.8	168.9
DMM modifiée	81.9	87.8	101.6	155.8	203.3	204.7	207.2	204.2
DMM	81.3	88.1	102.1	153.7	187.8	176.5	186.3	187.2

Tableau 3.8: Temps de cycle pour une capacité de file d'attente = 2.

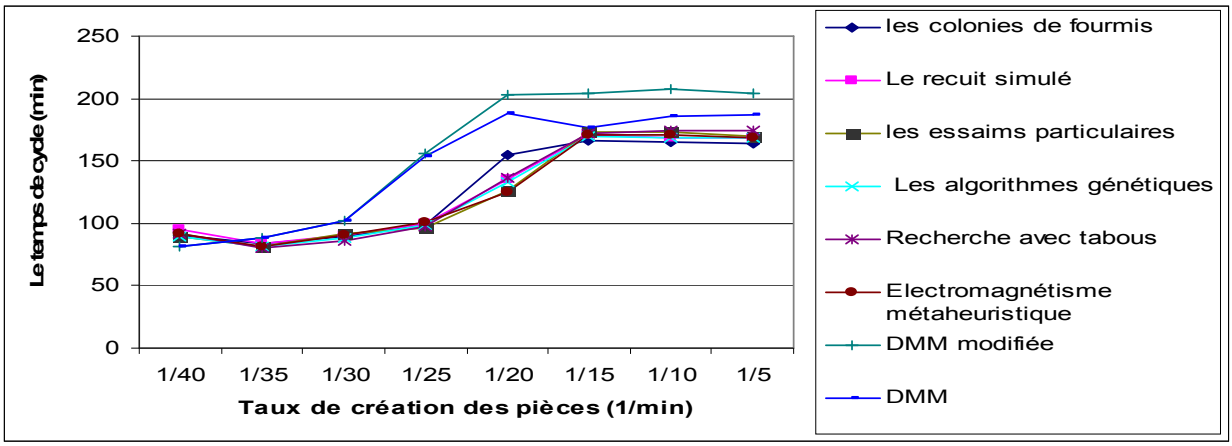

Figure 3.8: Temps de cycle pour une capacité de file d'attente = 2

La courbe de figure 3.8 et le tableau 3.8 montrent que pour une taille de file d'attente égale à 2 les temps de cycle de la méthode DMM et DMM modifiée sont supérieurs à ceux obtenus par toutes les métaheuristiques étudiées sauf pour un taux de création de pièces égale à 1/40 où la méthode DMM donne le meilleur temps de cycle.

Pour une petite capacité de file d'attente et un taux de création de pièces élevé (supérieur à 1/20), les colonies de fourmis sont meilleures que les autres métaheuristiques.

Pour un taux de création de pièces inférieur ou égale à 1/20, nous ne pouvons pas dire qu'une métaheuristique est la meilleure pour tous les taux.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	89.3	82.3	89.5	95.6	130	233.5	242.8	247.9
Le recuit simulé	92.6	81.9	87.6	94.6	120.2	249.5	249.3	249.1
Les essaims particulaires	92.3	82.4	88.3	97.4	119.9	248.7	246.7	249.3
Les algorithmes génétiques	91.1	82.4	91.4	98.6	118	250.9	254.6	249.9
Recherche avec tabous	95.6	83.6	89.2	96.7	125.3	255.7	252.4	252.1
Electromagnétisme métaheuristique	93.5	81.4	89.5	98.8	120.7	243.8	252.5	249.6
DMM modifiée	82.02	86.63	98.07	130.0	309.8	310.4	309.9	311.8
DMM	82.03	86.69	98.07	129.9	245.2	247.2	231.0	246.3

Tableau 3.9: Temps de cycle pour une capacité de file d'attente = 4.

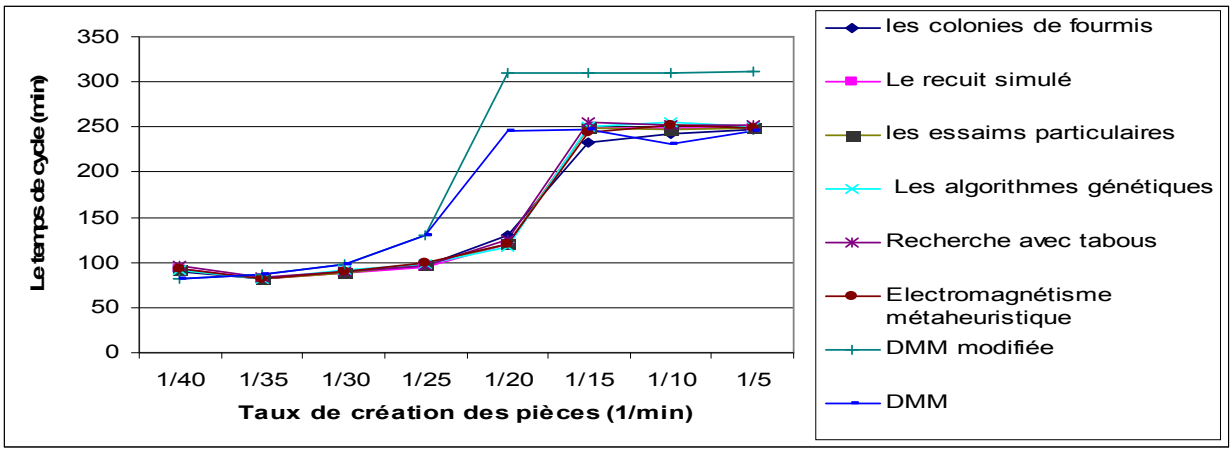

Figure 3.9: Temps de cycle pour une capacité de file d'attente = 4.

La courbe de figure 3.9 et le tableau 3.9 montrent que pour une taille de file d'attente égale à 4, nous ne pouvons pas dire qu'une règle ou métaheuristique est la meilleure pour tous les taux de création de pièces.

Pour un taux de création supérieur à 1/15, les temps de cycles obtenus par DMM sont meilleurs que ceux de DMM modifiée et les métaheuristiques, les colonies de fourmis sont les meilleures pour un taux de création égale à 1/15, les temps de cycle sont les minimaux pour les algorithmes génétiques pour un taux de création égale à 1/20, le recuit simulé pour un taux compris entre 1/25 et 1/30, l'électromagnétisme pour un taux égale à 1/35 et DMM modifiée pour un taux égale 1/40.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	91.1	82.6	89.3	97	121.4	326	324	320.6
Le recuit simulé	89	80.6	88.8	100.1	126	325.3	327.5	324.9
Les essaims particulaires	91	82.5	91.1	102.2	133.6	332.3	326	331.2
Les algorithmes génétiques	93.1	83.4	91.6	98.9	121.3	331.7	332.8	338.1
Recherche avec tabous	97.1	78.8	89.6	98.6	125.6	329.2	328.8	330.3
Electromagnétisme métaheuristique	90.3	82.1	88.7	99.81	122.2	320.2	326.4	328.1
DMM modifiée	81.26	86.6	97.57	126.8	416.7	416.7	414.7	414.3
DMM	81.26	86.67	97.65	127.2	276.9	279.2	269.9	283.9

Tableau 3.10: Temps de cycle pour une capacité de file d'attente = 6.

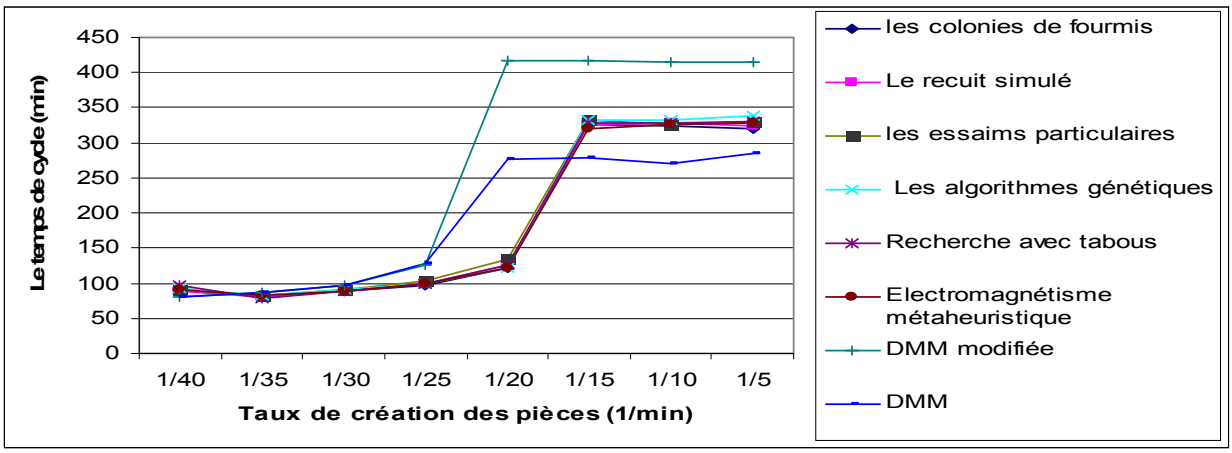

Figure 3.10: Temps de cycle pour une capacité de file d'attente = 6.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	92.8	83.7	88.7	96.5	119.8	398.8	399.1	382.4
Le recuit simulé	90.6	81.4	88.2	95.7	124.5	408.8	401.4	405.2
Les essaims particulaires	92	82.3	90.2	98.9	117.5	407.4	409.2	406.9
Les algorithmes génétiques	91.6	81.2	89.1	96.5	119.7	417.4	417.2	417
Recherche avec tabous	94.4	81.9	89.5	97.8	124	425.2	429	428.5
Electromagnétisme métaheuristique	92.1	82.7	89	97.4	122.7	410	403.6	407.1
DMM modifiée	81.26	86.6	97.53	126.1	519.3	521.2	521.2	532.8
DMM	81.26	86.67	97.7	125.8	319.2	310.9	311.4	321.3

Tableau 3.11: Temps de cycle pour une capacité de file d'attente = 8.

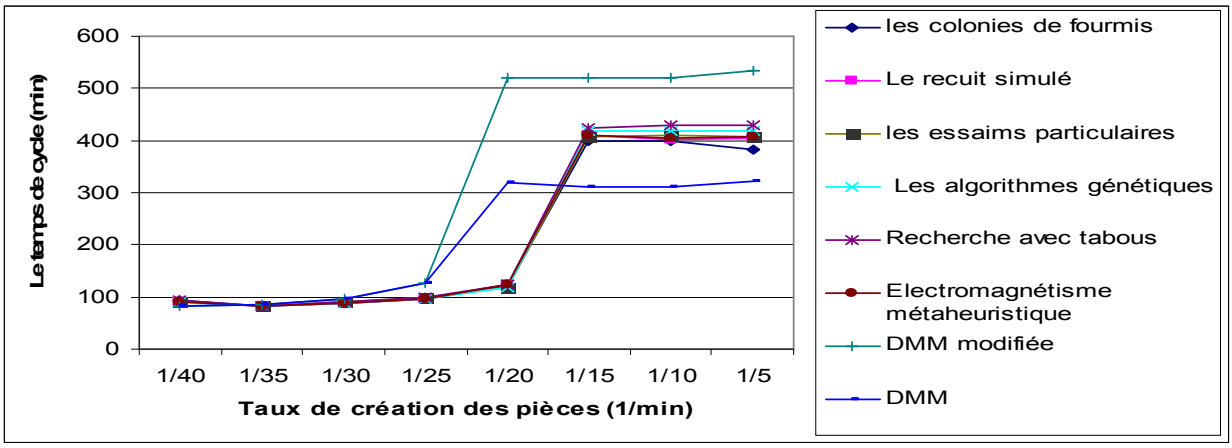

Figure 3.11: Temps de cycle pour une capacité de file d'attente = 8.

Les courbes des figures et tableaux (3.10 et 3.11) montrent que pour une taille de file d'attente supérieure à 4. Les temps de cycle de la méthode DMM sont inférieurs à ceux des métaheuristiques étudiées et DMM modifiée pour un taux de création de pièces supérieur à 1/20, dans cet intervalle les colonies de fourmis sont les plus efficaces si on les compare avec les autres métaheuristiques.

Dans l'intervalle ([1/35, 1/20]), les temps de cycles obtenus par toutes les métaheuristiques sont meilleurs que ceux de DMM et DMM modifiée, mais nous ne pouvons pas dire qu'une métaheuristique est la meilleure pour tous les taux de création. DMM et DMM modifiée sont les meilleures pour un taux de création de pièces égale à 1/40.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	163.5	247.9	320.6	382.4
Le recuit simulé	168.1	249.1	324.9	405.2
Les essaims particulaires	169.4	249.3	331.2	406.9
Les algorithmes génétiques	168.1	249.9	338.1	417
Recherche avec tabous	173.9	252.1	330.3	428.5
Electromagnétisme métaheuristique	168.9	249.6	328.1	407.1
DMM modifiée	204.2	311.8	414.3	551.1
DMM	187.2	246.3	283.9	321.3

Tableau 3.12: Temps de cycle pour un taux de création de pièces =1/5 (1/min)

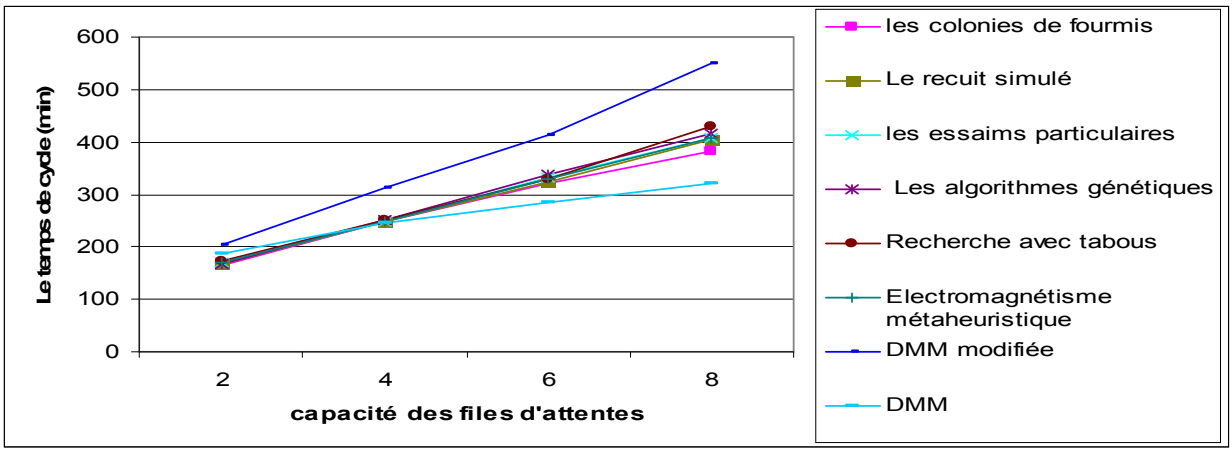

Figure 3.12: Temps de cycle pour un taux de création de pièces =1/5 (1/min)

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	155.2	130	121.4	119.8
Le recuit simulé	135.5	120.2	126	124.5
Les essaims particulaires	125.9	119.9	133.6	117.5
Les algorithmes génétiques	132.7	118	121.3	119.7
Recherche avec tabous	135.9	125.3	125.6	124
Electromagnétisme métaheuristique	124.8	120.7	122.2	122.7
DMM modifiée	203.3	309.8	416.7	519.3
DMM	187.8	245.2	276.9	319.2

Tableau 3.13: Temps de cycle pour un taux de création de pièces =1/20 (1/min)

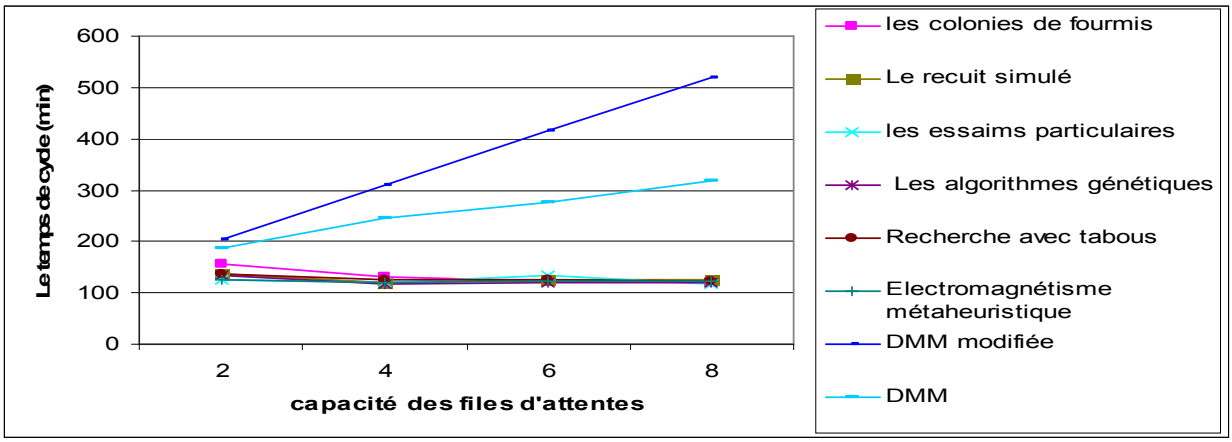

Figure 3.13: Temps de cycle pour un taux de création de pièces =1/20 (1/min)

Les figures et les tableaux (3.12 et 3.13) montrent que si on varie les capacités de files d'attente en gardant le taux de création fixe, nous pouvons dire que les métaheuristiques ont pu améliorer le temps de cycle pour un système très saturé et de petites files d'attente où les colonies des fourmis donnent les meilleurs temps de cycle.

Nous pouvons aussi remarquer l'augmentation des temps de cycle pour les règles DMM et DMM modifiée par rapport aux métaheuristiques étudiées pour un taux de création de pièces égale

à 1/20 car pour ce taux de création le système est saturé si on utilise les règles DMM et DMM modifiée.

3.4.1.3. Les en-cours

Les en-cours sont les pièces présentées dans le système. Cette sous section est consacrée aux en-cours.

L'analyse présentée dans cette sous section montre la variation des en-cours en fonction du taux de création de pièces et en fonction des capacités de files d'attente.

La figure 3.14 nous montre que les en-cours obtenus par les métaheuristiques sont inférieurs à ceux obtenus par les règles DMM et DMM modifiée, pour un système saturé et de petites files d'attente.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	3.9	4.04	4.45	5.21	7.29	8.4	8.44	8.46
Le recuit simulé	3.91	4.03	4.46	5.22	7.11	7.95	7.95	7.95
Les essaims particulaires	3.88	4.03	4.45	5.25	7.35	8.67	8.67	8.64
Les algorithmes génétiques	3.9	4.04	4.46	5.23	7.45	8.9	8.85	8.85
Recherche avec tabous	3.89	4.03	4.47	5.26	7.98	8.4	8.4	8.41
Electromagnétisme métaheuristique	3.89	4.04	4.44	5.28	7.99	8.64	8.66	8.62
DMM modifiée	6.79	12.95	15.79	19.01	17.84	18.19	16.86	18.12
DMM	8.37	7.79	14.33	15.96	15.82	15.4	12.82	13.67

Tableau 3.14: Les en-cours pour une capacité de file d'attente = 2.

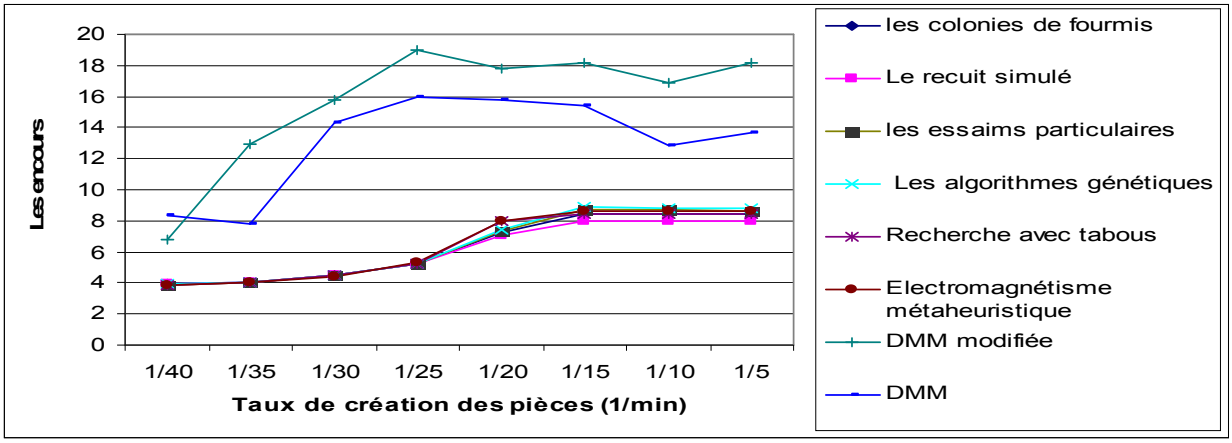

Figure 3.14: Les en-cours pour une capacité de file d'attente = 2.

La figure et le tableau 3.14 montrent que pour de capacités de files d'attente égale à 2, le nombre de pièces qui sont restées dans le système de la méthode DMM est inférieur à celui obtenu par DMM modifiée sauf pour un taux de création égale à 1/40.

Pour une taille de files d'attente égale à 2, nous pouvons remarquer que le recuit simulé est le plus efficace pour un taux de création supérieur à 1/25, et hors de cet intervalle il n'existe pas une métaheuristique qui donne les meilleurs résultats pour tous les taux.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	7.89	8.02	8.45	9.2	11.31	18.46	18.56	18.56
Le recuit simulé	7.89	8.03	8.44	9.2	11.15	18.46	18.52	18.6
Les essaims particulaires	7.88	8.04	8.47	9.26	11.16	19.27	19.28	19.24
Les algorithmes génétiques	7.89	8.04	8.46	9.26	11.26	19.68	19.72	19.73
Recherche avec tabous	7.89	8.03	8.48	9.28	11.19	19.46	19.45	19.53
Electromagnétisme métaheuristique	7.89	8.03	8.45	9.25	11.14	18.83	19.2	18.89
DMM modifiée	3.79	4.8	6.89	46.96	50.99	49.87	43.81	42.46
DMM	3.78	4.8	6.83	14	15.4	14.1	15.3	11.4

Tableau 3.15: Les en-cours pour une capacité de file d'attente = 6.

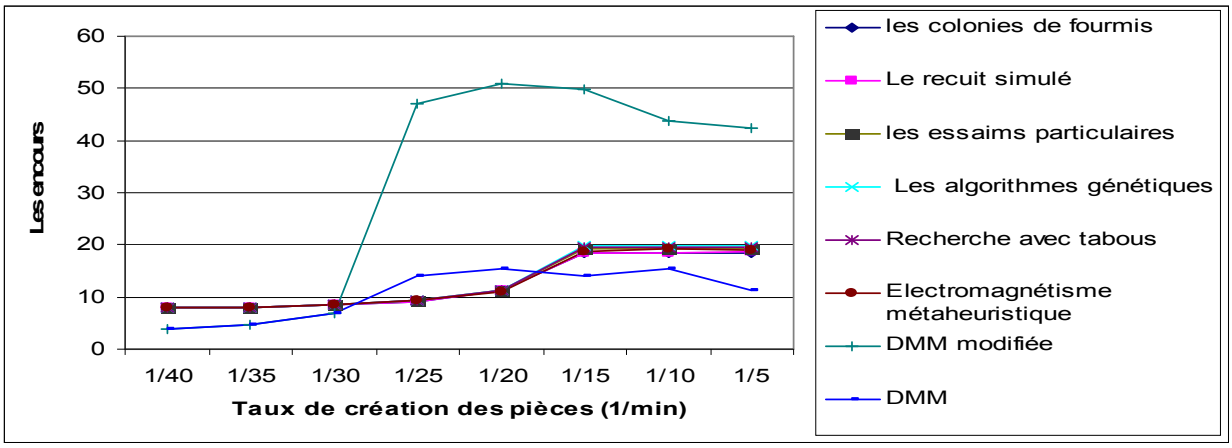

Figure 3.15: Les en-cours pour une capacité de file d'attente = 6.

A partir des figures et tableaux 3.15, nous pouvons remarquer que si la taille de files d'attente est égale à 6, le nombre de pièces restantes obtenus par la règle DMM est inférieur à ceux obtenus par la règle DMM modifiée et les métaheuristiques, pour un taux de création compris entre [1/15, 1/5] et [1/40, 1/30], si on s'intéresse aux métaheuristiques dans cette intervalle, on remarque que les colonies de fourmis et le recuit simulé sont meilleurs que les autres métaheuristiques, sauf pour un taux égale à 1/40. Et hors de cet intervalle (dans l'intervalle [1/25, 1/20]), le recuit simulé est le plus efficace.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	8.44	13.56	18.56	23.45
Le recuit simulé	7.95	13.17	18.52	23.99
Les essaims particulaires	8.67	13.92	19.28	24.76
Les algorithmes génétiques	8.85	14.25	19.72	25.27
Recherche avec tabous	8.4	13.94	19.45	24.97
Electromagnétisme métaheuristique	8.66	13.45	19.20	24.32
DMM modifiée	16.86	32.42	43,81	69.1
DMM	12.82	13.1	15.3	17.9

Tableau 3.16: Les en-cours pour un taux de création de pièces =1/10 min

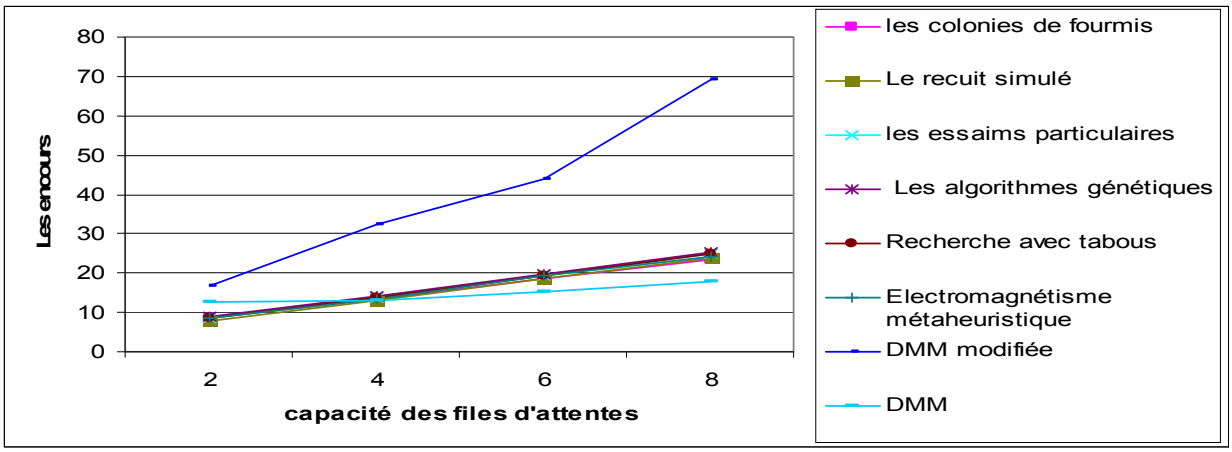

Figure 3.16: Les en-cours pour un taux de création de pièces =1/10 min

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	7.29	9.31	11.31	13.3
Le recuit simulé	7.11	9.5	11.15	13.38
Les essaims particulaires	7.35	9.21	11.16	13.28
Les algorithmes génétiques	7.45	9.35	11.26	13.15
Recherche avec tabous	7.98	9.3	11.19	13.29
Electromagnétisme métaheuristique	7.99	9.2	11.14	13.36
DMM modifiée	17.84	29.36	50.99	59.45
DMM	15.82	13.1	15.4	16.8

Tableau 3.17: Les en-cours pour un taux de création de pièces =1/20 min

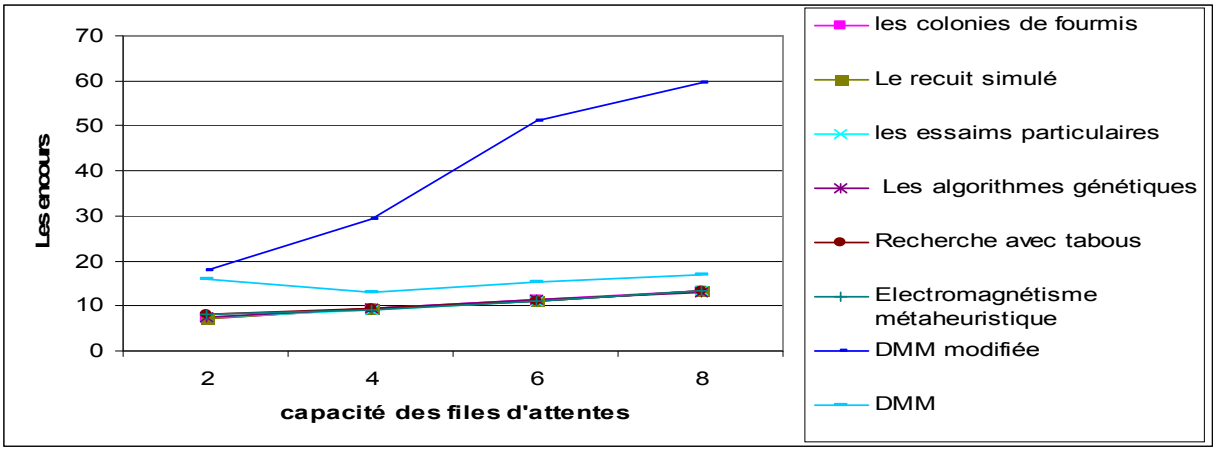

Figure 3.17: Les en-cours pour un taux de création de pièces =1/20 min

La figure 3.16 montre que pour un taux d'arrivée élevé et de petites files d'attente, le recuit simulé est le meilleur si on s'intéresse aux en-cours et si la taille de file d'attente augmente la règle DMM est la plus efficace.

La figure 3.17 montre que l'augmentation des en-cours pour les règles DMM et DMM modifiée est importante que pour les métaheuristiques pour un taux de création égale à 1/20, car pour ce taux le système est saturé si on utilise les règles DMM et DMM modifiée et n'est pas saturé si on utilise les métaheuristiques.

3.4.1.4. Taux d'utilisation des machines

Le taux d'utilisation des machines est un critère très important dans la mesure des performances d'un système de production.

Les courbes des figures 3.18 et 3.19 nous donnent le taux d'utilisation moyen des machines FV_1 et FV_2. Nous pouvons bien remarquer que le taux d'utilisation des fraiseuses verticales est plus élevé pour les métaheuristiques étudiées que pour DMM et DMM modifiée.

Les courbes des figures 3.20 et 3.21 montrent aussi que le taux d'utilisation pour les machines T_1 et T_2 est plus important pour certaines métaheuristiques (les essaims particulaires, les algorithmes génétiques, la recherche avec tabous et électromagnétisme métaheuristique) que pour DMM et DMM modifiée pour un taux de création important (supérieur à 1/25) et de petites files d'attente.

Le taux d'utilisation de la toupie (voir les figures 3.22 et 3.23) est plus faible dans la méthode DMM et les métaheuristiques que celui dans la méthode DMM modifiée surtout pour des taux de création supérieurs à 1/25.

Les machines FH_1 et FH_2 sont moins utilisées dans la méthode DMM et DMM modifiée que dans les métaheuristiques (voir les figures 3.24 et 3.25).

La règle DMM modifiée donne de meilleurs résultats par rapport à la règle DMM concernant le taux d'utilisation des machines pour un système saturé.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	14.53	17.56	20.66	24.74	28.78	26.77	26.93	26.67
Le recuit simulé	14.33	17.39	19.91	24.89	30.49	23.34	23.27	23.26
Les essaims particulaires	14.71	17.42	20.16	24.44	32.51	27.61	27.44	27.72
Les algorithmes génétiques	14.62	17.67	19.7	25.54	31.94	28.6	28.36	28.72
Recherche avec tabous	14.78	17.78	20.45	24.59	30.13	24.1	24.08	24.24
Electromagnétisme métaheuristique	14.77	17.71	20.16	24.9	32.47	26.12	26.22	26.27
DMM modifiée	12.7	14.61	16.99	20.32	21.65	17.52	21.13	21.58
DMM	12.69	14.58	17.04	16.65	6.26	10.98	7.88	9.08

Tableau 3.18: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 2.

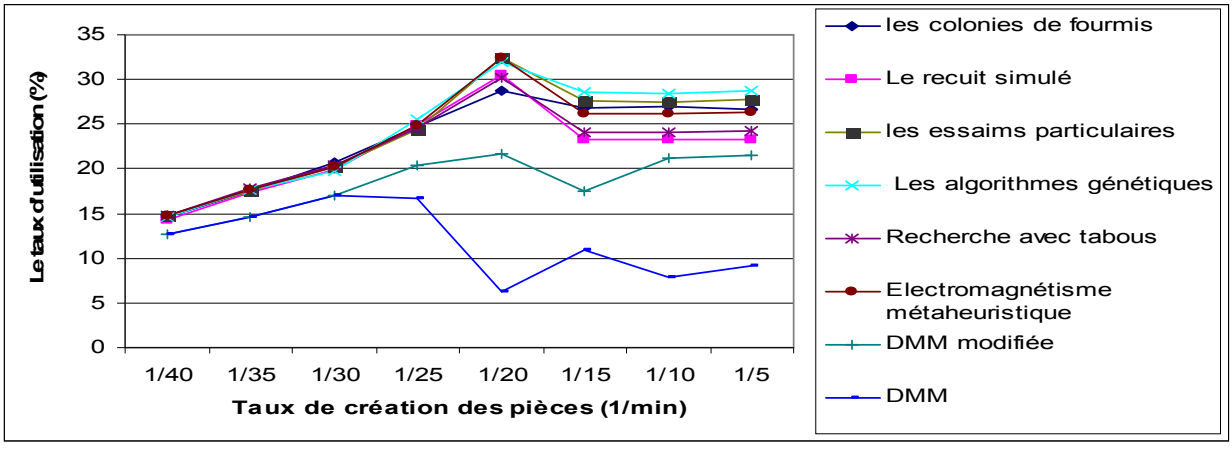

Figure 3.18: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 2.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	14.4	17.57	20	24.64	33.27	26.14	26.12	26.22
Le recuit simulé	14.93	17.6	19.99	24.45	33.33	25.91	26.07	26.19
Les essaims particulaires	14.56	17.64	19.91	24.59	33.24	28.04	28.09	28.23
Les algorithmes génétiques	14.62	17.28	19.83	24.44	33.51	28.37	28.38	28.7
Recherche avec tabous	14.49	17.63	20.28	24.7	33.18	27.55	27.51	27.53
Electromagnétisme métaheuristique	14.74	17.76	20.49	24.5	33.37	27.79	26.94	26.96
DMM modifiée	12.67	14.57	17.03	20.33	23.73	23.72	23.84	23.82
DMM	12.68	14.57	17.04	20.39	23.46	18.03	20.92	18.74

Tableau 3.19: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 6.

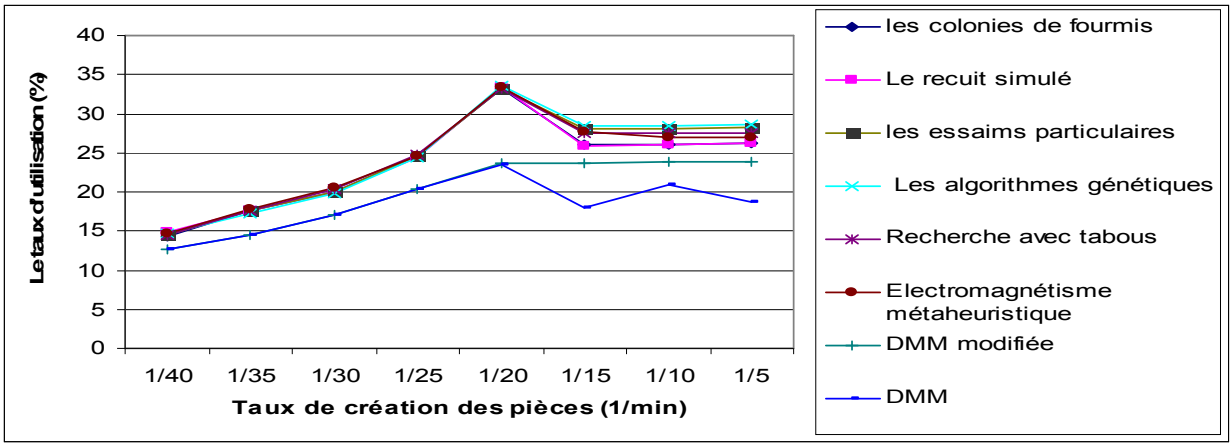

Figure 3.19: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 6.

On remarque que si on s'intéresse au taux d'utilisation des machines FV_1 et FV_2 pour un système saturé, les algorithmes génétiques sont plus efficaces que les règles DMM, DMM modifiée et les autres métaheuristiques (sauf pour une capacité de file d'attente égale à 2 et un taux de création de pièces égale à 1/20 où les essaims particulaires sont les meilleurs).

Pour un système non saturé, nous ne pouvons pas dire qu'une métaheuristique est meilleure par rapport aux autres et qu'une règle est meilleure par rapport à l'autre pour tous les taux de création des pièces :

- Pour de petites capacités de files d'attente, les algorithmes génétiques sont les meilleures pour un taux d'arrivée de pièces égale à 1/25, les colonies de fourmis sont les plus efficaces pour un taux égale à 1/30, et la recherche tabou est la meilleure pour un taux inférieur ou égale à 1/35.
- Pour une capacité de file d'attente égale à 6, la recherche tabou est la meilleure pour un taux d'arrivée de pièces égale à 1/25, l'électromagnétisme donne la plus grande utilisation pour un taux compris entre [1/35,1/30], et le recuit simulé est le plus efficace pour un taux égale à 1/40.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	48.44	54.59	63.99	76.31	85.73	81.97	82.26	82.06
Le recuit simulé	48.59	54.74	64.16	76.2	89.42	80.49	80.41	80.47
Les essaims particulaires	48.29	54.71	63.96	76.55	93.2	89.2	88.84	88.95
Les algorithmes génétiques	48.36	54.51	64.32	75.67	92.72	90.98	90.85	90.96
Recherche avec tabous	48.24	54.42	63.72	76.43	90.36	84.73	84.75	85.27
Electromagnétisme métaheuristique	48.24	54.48	63.96	76.19	92.39	86.16	86.56	86.75
DMM modifiée	49.97	56.94	66.59	79.85	84.52	82.72	83.58	84.7
DMM	49.94	57	66.56	65.05	24.62	42.59	30.9	35.5

Tableau 3.20: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 2.

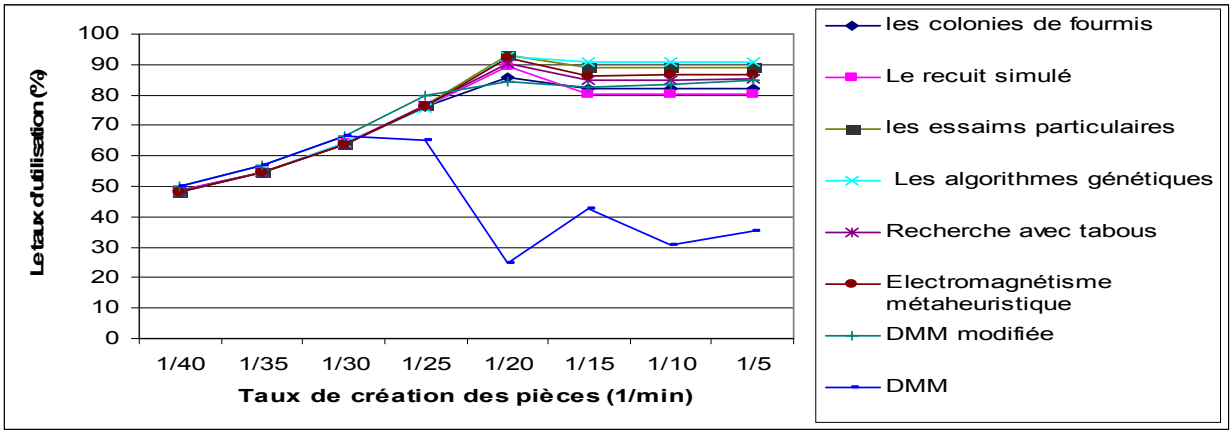

Figure 3.20: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 2.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	48.54	54.76	64.09	76.39	93.52	89.57	89.35	89.48
Le recuit simulé	48.11	54.57	64.09	76.55	93.37	87.37	87.68	88.13
Les essaims particulaires	48.41	54.53	64.16	76.43	93.54	91.53	91.59	91.69
Les algorithmes génétiques	48.37	54.82	64.23	76.55	93.33	93.3	93.25	93.24
Recherche avec tabous	48.47	54.53	63.86	76.35	93.59	92.28	92.38	92.34
Electromagnétisme métaheuristique	48.27	54.42	63.69	76.51	93.44	89.30	89.53	89.74
DMM modifiée	49.97	56.97	66.61	79.93	92.8	92.88	93.14	93.37
DMM	49.96	56.98	66.62	79.99	91.72	71	82.46	73.54

Tableau 3.21: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 6.

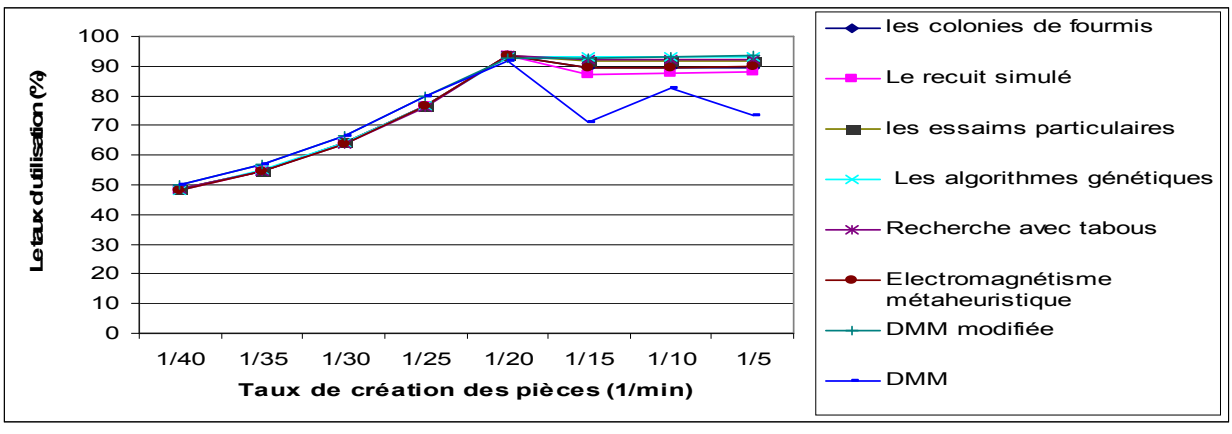

Figure 3.21: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 6.

Les courbes des figures et les tableaux 3.20 et 3.21 montrent aussi que le taux d'utilisation pour les machines T_1 et T_2 est plus important pour les algorithmes génétiques que pour DMM, DMM modifiée et les autres métaheuristiques étudiées, pour un taux de création important (supérieur à 1/25), sauf pour certains cas :

- Pour un taux de création égale à 1/20 et une taille de files d'attente égale à 2, les essaimes particulaires sont les meilleurs.
- Pour un taux de création égale à 1/5 et une taille de files d'attente égale à 6, DMM modifiée est la plus efficace.
- Pour un taux de création égale à 1/20 et une taille de files d'attente égale à 6, la recherche tabou est la meilleure.

Pour un taux de création inférieur à 1/25, DMM et DMM modifiée sont plus efficaces que les métaheuristiques étudiées, dans cet intervalle, certaines métaheuristiques sont meilleures que d'autres mais il n'existe pas une métaheuristique qui donne les meilleurs taux d'utilisation des machines pour tous les taux d'arrivée des pièces.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	1.307	1.417	1.663	1.97	2.169	2.108	2.113	2.117
Le recuit simulé	1.322	1.431	1.699	1.959	2.24	2.209	2.209	2.212
Les essaims particulaires	1.292	1.428	1.679	1.995	2.3	2.357	2.359	2.351
Les algorithmes génétiques	1.299	1.408	1.711	1.907	2.307	2.393	2.399	2.387
Recherche avec tabous	1.287	1.4	1.656	1.983	2.295	2.348	2.35	2.364
Electromagnétisme métaheuristique	1.287	1.405	1.679	1.959	2.269	2.319	2.323	2.329
DMM modifiée	1.46	1.69	2.09	3.82	5.35	8.6	6.06	5.57
DMM	1.45	1.68	2.09	2.79	1.25	2.03	1.6	1.91

Tableau 3.22: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 2

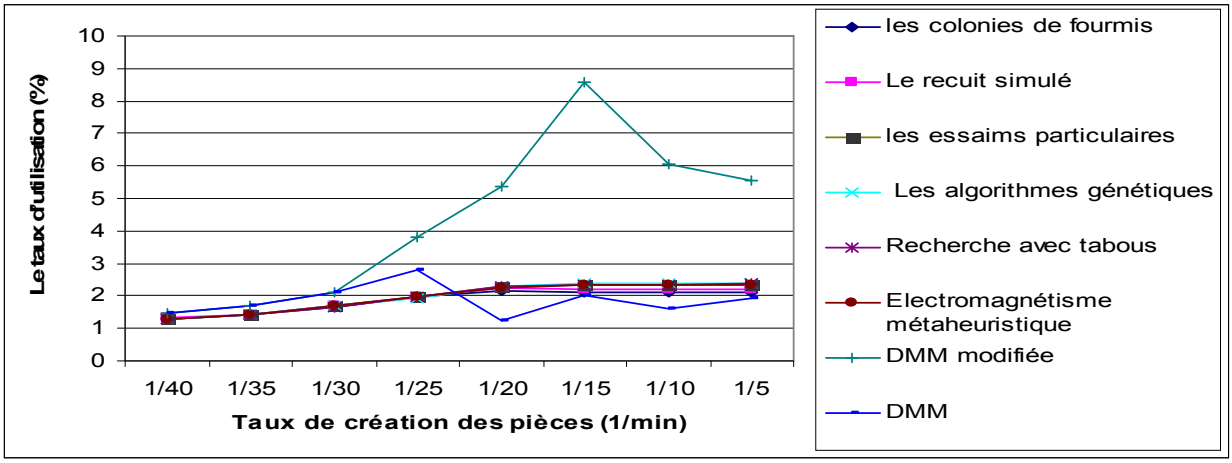

Figure 3.22: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 2.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	1.317	1.434	1.693	1.979	2.277	2.451	2.443	2.443
Le recuit simulé	1.274	1.414	1.693	1.995	2.272	2.371	2.376	2.389
Les essaims particulaires	1.304	1.41	1.699	1.983	2.279	2.442	2.442	2.44
Les algorithmes génétiques	1.299	1.44	1.706	1.995	2.258	2.499	2.497	2.481
Recherche avec tabous	1.309	1.411	1.669	1.975	2.284	2.496	2.502	2.499
Electromagnétisme métaheuristique	1.289	1.399	1.653	1.991	2.269	2.409	2.411	2.419
DMM modifiée	1.45	1.66	1.93	2.32	2.73	2.73	2.74	2.75
DMM	1.45	1.66	1.94	2.32	2.67	2.07	2.41	2.14

Tableau 3.23: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 6.

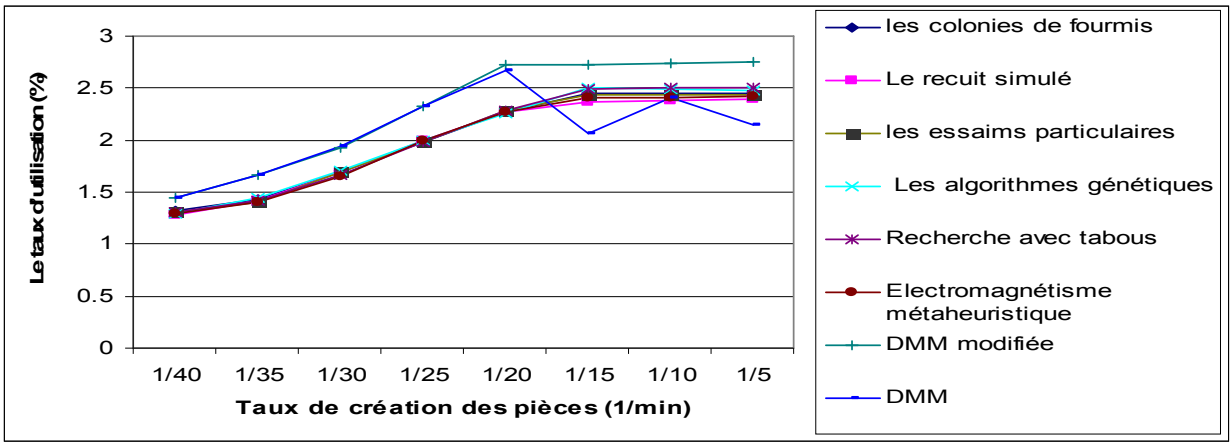

Figure 3.23: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 6.

Le taux d'utilisation de la toupie (voir les figures 3.22 et 3.23) est plus faible pour la méthode DMM et les métaheuristiques que celui obtenu par la méthode DMM modifiée pour des taux de création supérieurs à 1/30. Pour les autres taux de création, les taux d'utilisation pour DMM et DMM modifiée sont pratiquement égaux.

Si on s'intéresse aux métaheuristiques, nous pouvons remarquer qu'il n'existe pas une grande différence entre les métaheuristiques mais nous pouvons voir que pour un système saturé, les algorithmes génétiques sont les meilleurs pour une taille de files d'attente égale à 2, et la recherche tabou est la plus efficace pour de grandes capacités de files d'attente.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	23.19	25.15	29.52	35	38.52	37.44	37.52	37.59
Le recuit simulé	23.45	25.4	30.16	34.79	39.79	39.18	39.18	39.23
Les essaims particulaires	22.92	25.35	29.81	35.42	40.87	41.82	41.87	41.71
Les algorithmes génétiques	23.05	25	30.45	33.88	40.99	42.47	42.57	42.36
Recherche avec tabous	22.84	24.85	29.4	36.22	40.76	41.63	41.66	41.92
Electromagnétisme métaheuristique	22.83	24.95	29.81	34.8	40.32	41.14	41.23	41.31
DMM modifiée	18.05	20.61	23.96	28.63	30.34	29.15	30.16	30.4
DMM	18.07	20.61	23.94	23.46	8.87	15.33	11.11	12.78

Tableau 3.24: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente = 2.

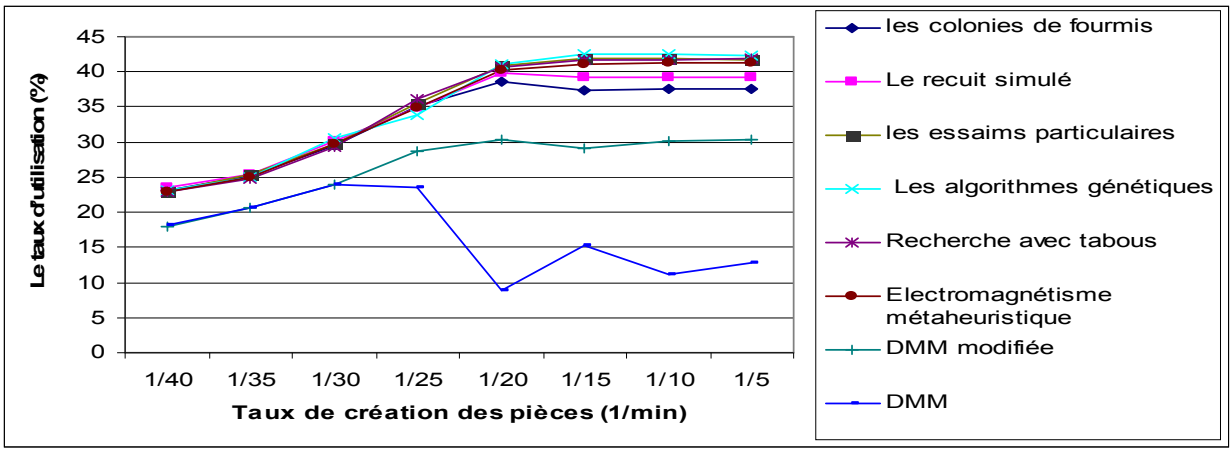

Figure 3.24: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente = 2.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	23.36	25.45	30.04	35.15	40.48	43.47	43.32	43.33
Le recuit simulé	22.61	25.1	30.05	35.42	40.39	42.06	42.15	42.37
Les essaims particulaires	23.14	25.05	30.16	35.22	40.52	43.33	43.33	43.28
Les algorithmes génétiques	23.06	25.55	30.27	35.43	40.14	44.34	44.29	44.02
Recherche avec tabous	23.23	25.05	29.64	35.07	40.61	44.27	44.37	44.33
Electromagnétisme métaheuristique	22.88	24.85	29.34	35.35	40.35	42.74	42.78	42.91
DMM modifiée	18.06	20.63	24.04	28.94	33.51	33.46	33.66	33.72
DMM	18.06	20.63	24.08	28.81	33.17	25.62	29.85	26.64

Tableau 3.25: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente = 6.

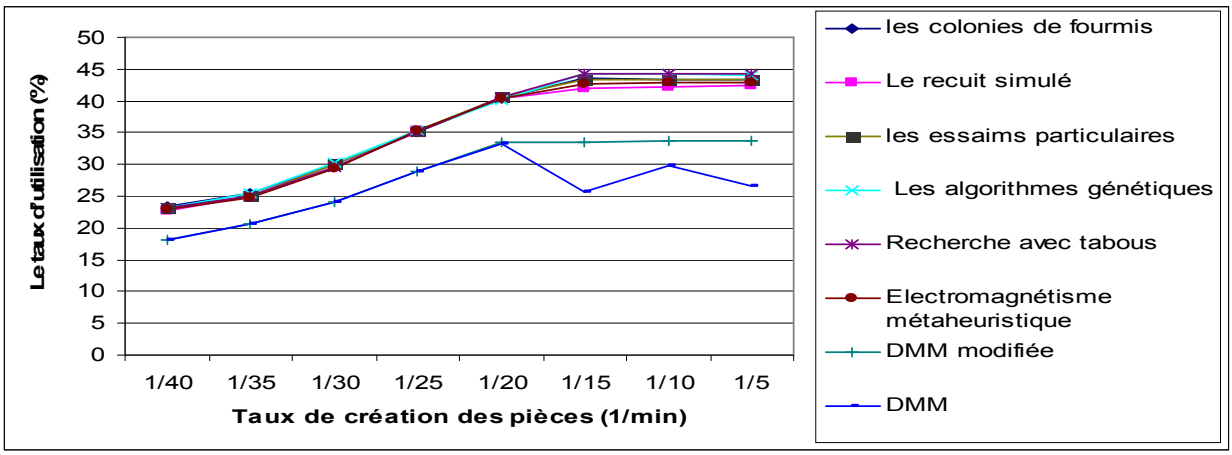

Figure 3.25: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente = 6.

Les machines FH_1 et FH_2 sont moins utilisées dans la méthode DMM et DMM modifiée que dans les métaheuristiques (voir les figures 3.24 et 3.25).

Pour un taux de création supérieur à 1/25 et une taille de files d'attente égale à 2, les algorithmes génétiques sont plus efficaces que DMM, DMM modifiée et les autres métaheuristiques, pour une taille de files d'attente égale à 6, l'utilisation des machines FH_1 et FH_2

est supérieure pour la recherche tabou que pour les autres métaheuristiques et règles DMM et DMM modifiée.

Pour un système non saturé, nous ne pouvons pas dire qu'une métaheuristique est la meilleure pour tous les taux d'arrivée des pièces mais dans chaque cas certaines métaheuristiques sont meilleures que d'autres.

3.4.1.5. Taux d'utilisation de l'AGV

Les figures 3.26 et 3.27 montrent que l'utilisation de l'AGV est plus importante pour les métaheuristiques (sauf le recuit simulé et les colonies de fourmis) que la méthode DMM et DMM modifiée pour un système saturé et de petites files d'attente.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	17.47	19.58	22.89	27.36	30.64	29.37	29.46	29.41
Le recuit simulé	17.55	19.65	23.08	27.3	31.92	29.1	29.07	29.1
Les essaims particulaires	17.4	19.64	22.98	27.48	33.2	32	31.97	31.98
Les algorithmes génétiques	17.43	19.54	23.16	27.04	33.07	32.69	32.67	32.68
Recherche avec tabous	17.37	19.5	22.86	27.42	32.32	30.68	30.69	30.88
Electromagnétisme métaheuristique	17.37	19.52	22.98	27.29	32.89	31.12	31.2	31.26
DMM modifiée	14.98	17.46	21	27.31	30.26	29.16	30.19	30.44
DMM	15	17.4	20.93	21.67	8.43	14.35	10.55	12.08

Tableau 3.26: Le taux d'utilisation de l'AGV pour une capacité de file d'attente = 2.

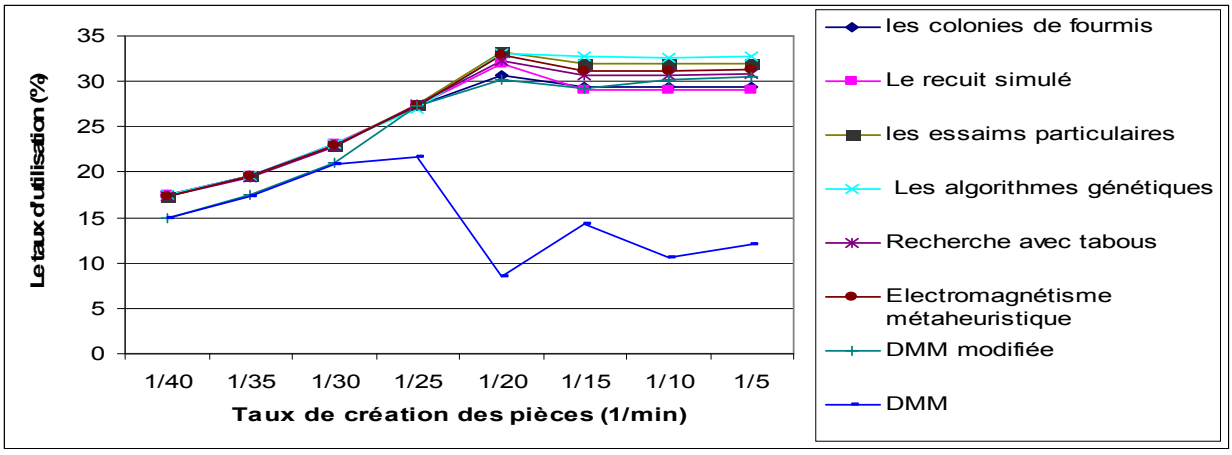

Figure 3.26: Le taux d'utilisation de l'AGV pour une capacité de file d'attente = 2.

La figure 3.26 montre que pour une taille de file d'attente égale à 2 et un taux de création de pièces supérieur à 1/20, le taux d'utilisation d'AGV pour les algorithmes génétiques est plus important que pour les autres métaheuristiques et règles DMM et DMM modifiée.

Dans l'intervalle ([1/25,1/20]), les essaims particulaires sont les plus efficaces, les algorithmes génétiques sont les meilleurs pour un taux égale à 1/30, le recuit simulé est le meilleur pour un taux de création de pièces inférieur à 1/30.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	17.52	19.67	23.05	27.39	33.26	32.37	32.29	32.33
Le recuit simulé	17.31	19.57	23.04	27.47	33.23	31.54	31.64	31.81
Les essaims particulaires	17.46	19.55	23.08	27.41	33.27	32.96	32.98	33
Les algorithmes génétiques	17.43	19.7	23.11	27.47	33.16	33.62	33.59	33.56
Recherche avec tabous	17.48	19.55	22.93	27.38	33.3	33.29	33.33	33.32
Electromagnétisme métaheuristique	17.38	19.49	22.85	27.46	33.22	32.21	32.28	32.36
DMM modifiée	15	17.49	20.84	25.62	33.53	33.51	33.56	33.64
DMM	15	17.48	20.76	25.62	30.67	23.75	27.63	24.68

Tableau 3.27: Le taux d'utilisation de l'AGV pour une capacité de file d'attente = 6.

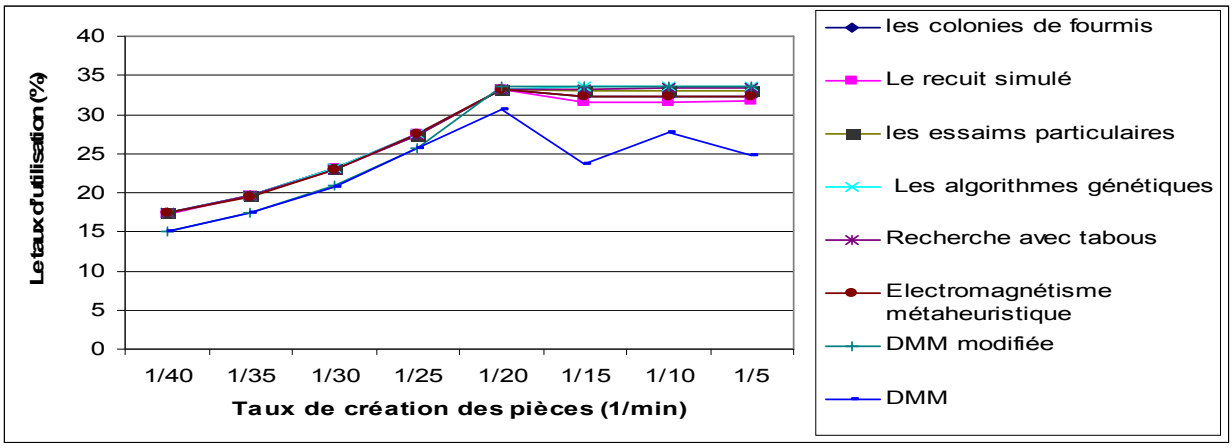

Figure 3.27: Le taux d'utilisation de l'AGV pour une capacité de file d'attente = 6.

La figure 3.27 montre que pour une taille de files d'attente égale à 6 et un taux de création de pièces égale à 1/5, la règle DMM modifiée donne le meilleur taux d'utilisation d'AGV. Et dans l'intervalle [1/15,1/10], les taux d'utilisation d'AGV pour les algorithmes génétiques sont meilleurs que ceux obtenus par les autres métaheuristiques et règles DMM et DMM modifiée. Pour un taux de création égale à 1/20, DMM modifiée est la plus efficace.

Dans l'intervalle ([1/35,1/25]), les algorithmes génétiques sont les meilleurs, et les colonies de fourmis sont les plus efficaces pour un taux de création de pièces égale à 1/40.

3.4.2. Etude comparative avec introduction de pannes

Dans cette section, nous nous proposons d'étudier et de comparer les métaheuristiques étudiées et les règles DMM et DMM modifiée avec l'introduction de pannes. Les pannes sont introduites sur les ressources toutes les 100 heures. La panne va durer 2 heures suivant une loi exponentielle.

Dans ce qui suit, nous allons donner les résultats et les interprétations de l'étude comparative entre les métaheuristiques étudiées et les règles DMM et DMM modifiée présentés déjà dans le paragraphe précédent, avec l'introduction de pannes.

3.4.2.1. Taux de production

La simulation du système en introduisant des pannes, nous montre que les taux de production pour les métaheuristiques étudiées sont nettement supérieurs à ceux des règles DMM et DMM modifiée avec une capacité de file d'attente égale à 2 (voir figure 3.28).

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.98	84.47	63.13	42.11	21.13
Le recuit simulé	99.99	99.99	99.99	99.99	80.72	60.22	40.24	20.23
Les essaims particulaires	99.99	99.99	99.99	99.99	90.04	67.5	44.99	22.61
Les algorithmes génétiques	99.99	99.99	99.99	99.99	92	69.05	46.22	23.12
Recherche avec tabous	99.99	99.99	99.99	99.99	84.53	63.45	42.28	21.12
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	89.79	65.61	43.76	21.96
DMM modifiée	99.99	99.97	99.62	85.02	66.7	50.91	34.29	17.04
DMM	99.99	99.97	99.95	53.07	32.46	29.29	15.43	8.55

Tableau 3.28: Taux de sortie des pièces pour une capacité de file d'attente = 2.

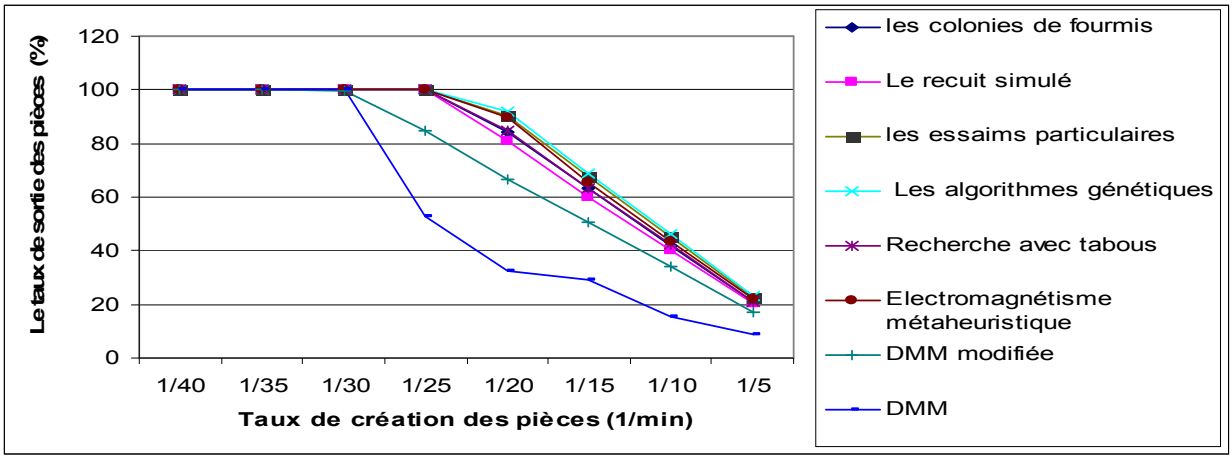

Figure 3.28: Taux de sorite des pièces pour une capacité de file d'attente = 2.

La figure 3.28 nous montre aussi que le taux de production pour une file d'attente égale à 2 est plus important pour les algorithmes génétiques que pour les autres métaheuristiques pour un taux de création supérieur à 1/25 et hors de cet intervalle les taux de production obtenus par les métaheuristiques sont identiques.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	84.47	90.27	99.99	99.99
Le recuit simulé	80.72	88.52	99.01	99.99
Les essaims particulaires	90.04	99.99	99.99	99.99
Les algorithmes génétiques	92	98.95	99.99	99.99
Recherche avec tabous	84.53	95.07	99.63	99.99
Electromagnétisme métaheuristique	89.78	93.46	99.99	99.99
DMM modifiée	66.7	90.6	93.11	94.24
DMM	32.46	76.07	91.94	95.69

Tableau 3.29:Taux de sortie des pièces pour un taux de création de pièces = 1/20 (1/min).

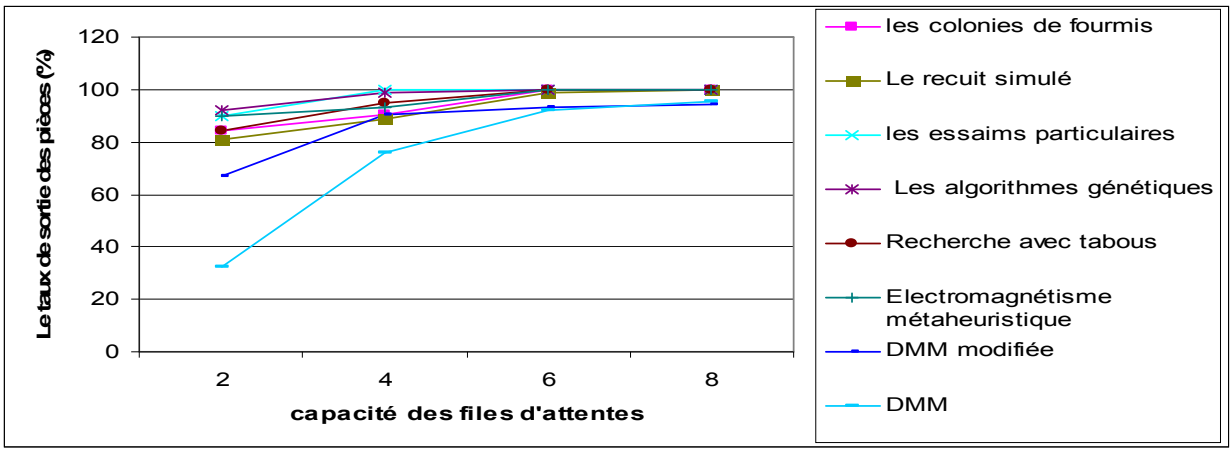

Figure 3.29:Taux de sortie des pièces pour un taux de création de pièces = 1/20 (1/min).

La figure 3.29 nous montre que pour un taux de création de pièces égale à 1/20, le taux de production est meilleur pour les algorithmes génétiques par rapport aux autres métaheuristiques et les règles DMM et DMM modifiée pour une taille de files d'attente égale à 2, et les essaims particulaires sont les plus efficaces pour une taille de files d'attente égale à 4. Pour de grandes capacités de files d'attente les résultats obtenus par les métaheuristiques sont presque les mêmes.

De ces résultats avec et sans présence de pannes, nous pouvons constater que le taux de production en utilisant les algorithmes génétiques est plus important que celui en utilisant les autres métaheuristiques et les règles DMM et DMM modifiée pour un taux de création de pièces très élevé (supérieur à 1/20). Nous pouvons remarquer aussi que les essaims particulaires sont les meilleurs pour un taux de création de pièces égale à 1/20, pour ce taux de création le système n'est pas saturé pour les métaheuristiques et saturé pour les règles DMM et DMM modifiée ce qui preuve que le système sature moins rapidement si on utilise les métaheuristiques.

3.4.2.2. Le temps de cycle

D'après les résultats illustrés dans les figures 3.30 et 3.31, nous remarquons que pour une taille de files d'attente égale à 2, les temps de cycle des métaheuristiques étudiées sont inférieurs à ceux obtenus par les règles DMM et DMM modifiée sauf pour un taux de création de pièces supérieur à 1/40.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	97.5	82.6	90.1	104	169.8	169.8	167.3	175.4
Le recuit simulé	93	86.2	93.5	99.4	172.3	175.7	175	173.3
Les essaims particulaires	95.5	84.9	93.2	103.2	170.8	175.1	174.6	173.9
Les algorithmes génétiques	94.2	85.7	93.3	104.6	170.8	174.6	175.2	176.2
Recherche avec tabous	94.5	85.9	91.9	103.4	177.1	178.4	173.7	176
Electromagnétisme métaheuristique	92.7	83.3	91.2	102.1	174	172.9	175.8	169.9
DMM modifiée	91.1	99.1	119.9	325.1	353.3	346.8	339.8	341.1
DMM	91.3	101.6	120.9	185.3	203.7	193.8	199.6	203

Tableau 3.30: Temps de cycle pour une capacité de file d'attente = 2.

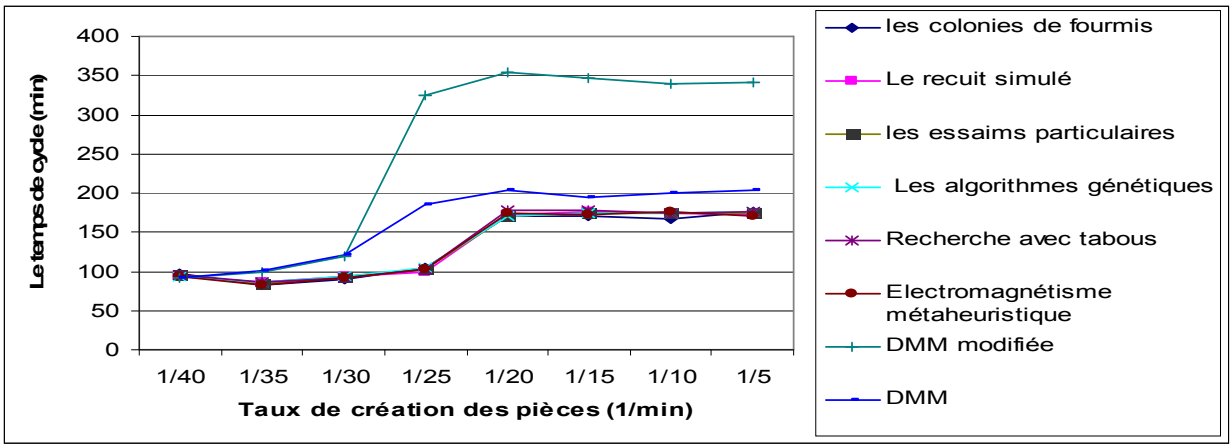

Figure 3.30: Temps de cycle pour une capacité de file d'attente = 2.

La figure et tableau 3.30 montrent que pour un taux de création de pièces égale à 1/5 et une taille de file d'attente égale à 2, l'électromagnétisme est la plus efficace si on s'intéresse aux temps de cycle. Dans l'intervalle [1/20,1/10], les temps de cycles obtenus par les colonies de fourmis sont meilleurs par rapport aux autres métaheuristiques et règles DMM et DMM modifiée.

Hors de cet intervalle, nous ne pouvons pas dire qu'une métaheuristique est la meilleure par rapport aux autres.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	169.8	239.2	141.7	153.7
Le recuit simulé	172.3	233.2	154.8	146.5
Les essaims particulaires	170.8	157.8	144.7	149.6
Les algorithmes génétiques	170.8	161	150.9	145.4
Recherche avec tabous	177.1	205.1	162.6	146.2
Electromagnétisme métaheuristique	174	197.9	146.9	139.4
DMM modifiée	353.34	309.84	414.32	519.36
DMM	203.79	239.49	276.95	319.23

Tableau 3.31: Temps de cycle pour un taux de création de pièces = 1/20 (1/min).

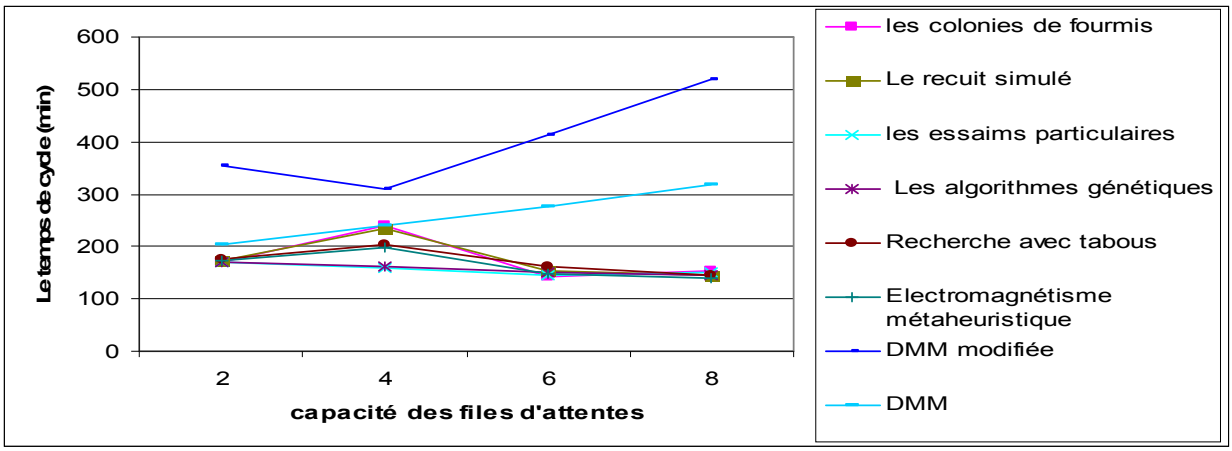

Figure 3.31: Temps de cycle pour un taux de création de pièces = 1/20 (1/min).

La figure et tableau 3.31 montrent l'augmentation du temps de cycle pour les règles DMM et DMM modifiée par rapport aux métaheuristiques étudiées pour un taux de création de pièces égale à 1/20 car le système se sature rapidement si on utilise les règles DMM et DMM modifiée.

3.4.2.3. Les en-cours

Le nombre de pièces qui restent dans le système est plus grand pour les métaheuristiques étudiées et la règle DMM modifiée vis-à-vis de la règle DMM, pour un taux de création de pièces supérieur à 1/30 et de petite taille de files d'attente (voir figure 3.32 et 3.33).

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	3.95	4.09	4.55	5.4	8.46	8.43	8.43	8.46
Le recuit simulé	3.93	4.1	4.54	5.44	7.95	7.94	7.99	8.05
Les essaims particulaires	3.94	4.09	4.55	5.39	8.67	8.68	8.68	8.7
Les algorithmes génétiques	3.92	4.1	4.58	5.34	8.83	8.87	8.85	8.81
Recherche avec tabous	3.94	4.1	4.54	5.36	8.41	8.4	8.41	8.39
Electromagnétisme métaheuristique	3.93	4.09	4.56	5.41	8.65	8.61	8.61	8.6
DMM modifiée	4.55	7.79	28.62	29.82	30.51	28.04	28.64	27.2
DMM	4.6	5.9	6.11	4.85	5.99	5.29	4.67	0.46

Tableau 3.32: Les en-cours pour une capacité de file d'attente = 2.

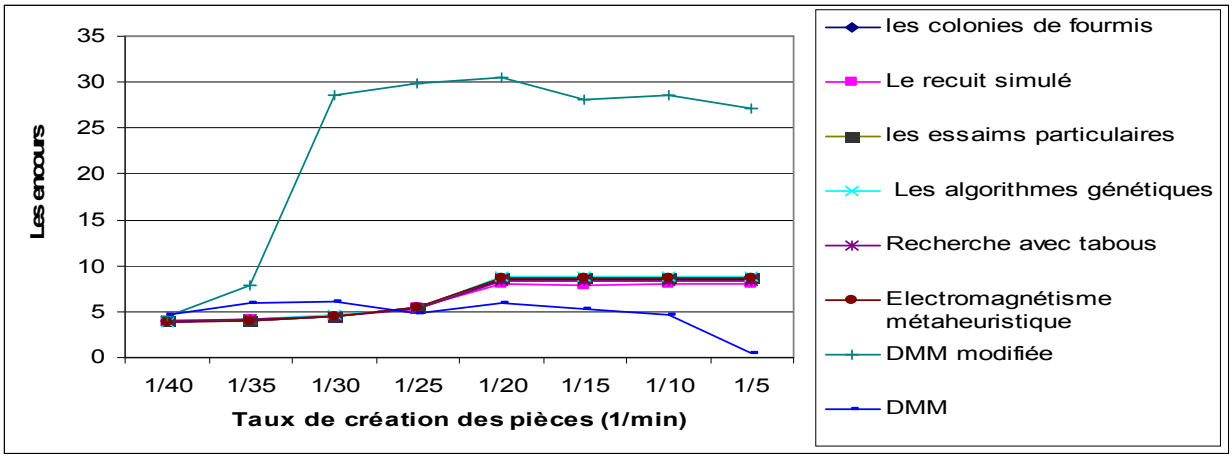

Figure 3.32 : Les en-cours pour une capacité de file d'attente = 2.

La figure 3.32 montre aussi que le recuit simulé donne des résultats meilleurs concernant les en-cours par rapport aux autres métaheuristiques et DMM modifiée pour une taille de files d'attente égale à 2 et un taux d'arrivée supérieur à 1/25.

Pour un taux d'arrivée de pièces supérieur à 1/35, l'augmentation des en-cours pour la règle DMM modifiée apparaît clairement par rapport aux métaheuristiques et la règle DMM.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	8.46	13.37	12.3	14.13
Le recuit simulé	7.95	13.15	12.79	14.39
Les essaims particulaires	8.67	10.13	12.47	14.43
Les algorithmes génétiques	8.83	11.42	12.28	14.36
Recherche avec tabous	8.41	11.48	12.43	14.34
Electromagnétisme métaheuristique	8.65	10.62	12.18	14.02
DMM modifiée	30.51	40.86	68.07	106.12
DMM	5.99	11.2	13.9	17.8

Tableau 3.33: Les en-cours pour un taux de création de pièces = 1/20 (1/min).

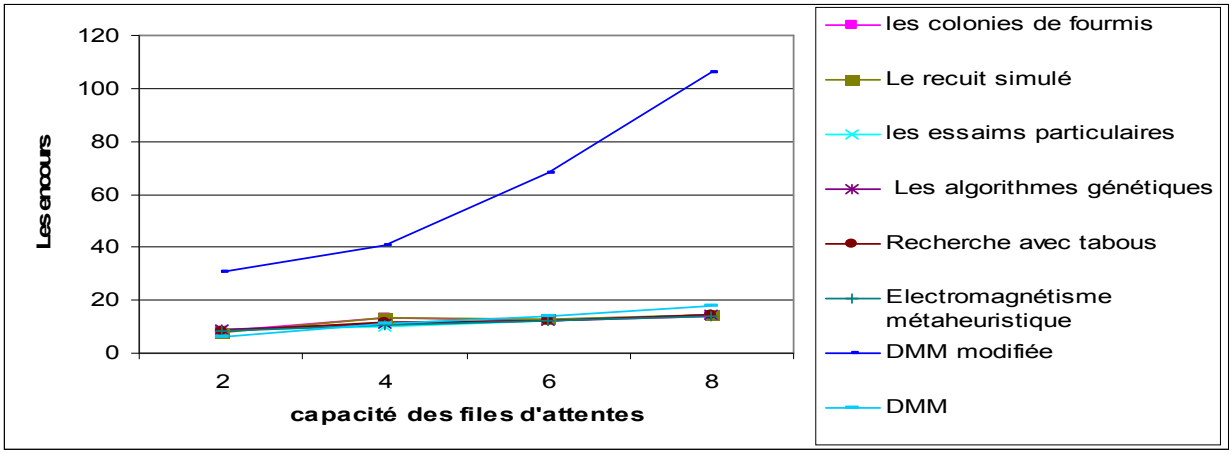

Figure 3.33: Les en-cours pour un taux de création de pièces = 1/20 (1/min).

La figure 3.33 montre que si on compare les métaheuristiques et les règles DMM et DMM modifiée, on remarque que pour une taille de file d'attente égale à 2, DMM est la plus efficace, si on s'intéresse aux en-cours. Si la taille de file d'attente augmente, nous pouvons dire que les métaheuristiques étudiées ont amélioré les résultats qui concerne les en-cours (les essaims particulaires sont les meilleurs pour une taille égale à 4 tandis que l'électromagnétisme est la meilleure pour une taille supérieure à 4), et DMM modifiée donne toujours les plus mauvais résultats concernant les en-cours.

3.4.2.4. Taux d'utilisation des machines

Comme nous pouvons bien le remarquer sur les figures (3.34 à 3.38), avec un taux de création de pièces supérieur à 1/25, le taux d'utilisation des machines est toujours plus important en utilisant les métaheuristiques, sauf l'utilisation de la machine TP. Ces figures montrent aussi que certaines métaheuristiques sont meilleures que d'autres mais il n'existe pas une métaheuristique plus efficace pour toutes les machines, et pour tous les taux.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	14.71	17.58	20.24	24.95	26.3	26.15	26.18	26.33
Le recuit simulé	14.79	17.72	19.79	24.88	22.81	22.66	22.72	22.95
Les essaims particulaires	14.29	17.3	19.78	24.57	26.83	26.87	26.86	27.2
Les algorithmes génétiques	14.7	17.18	20.07	25.02	27.1	27.72	28.05	28.04
Recherche avec tabous	14.58	17.49	20.17	25.1	23.51	23.56	23.55	23.45
Electromagnétisme métaheuristique	14.7	17.56	20.09	25.2	27.13	25.63	25.63	25.75
DMM modifiée	12.79	14.67	16.89	17.35	17.06	17.25	17.5	17.33
DMM	12.66	14.56	16.99	10.86	8.23	9.91	7.96	8.72

Tableau 3.34: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 2.

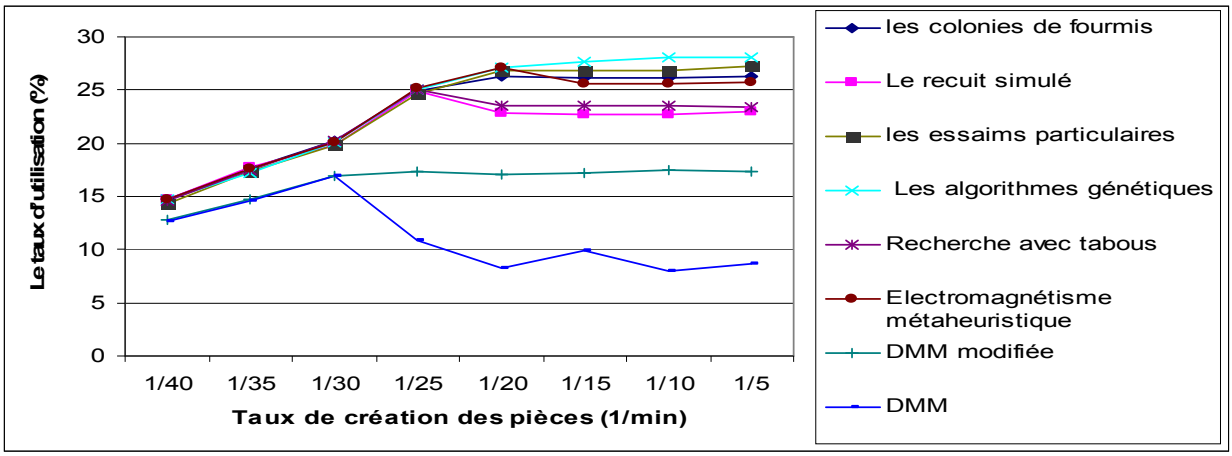

Figure 3.34: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 2.

La figure et tableau 3.34 montrent aussi que pour une capacité de file d'attente égale à 2, l'utilisation des machines FV_1 et FV_2 est plus importante pour DMM modifiée par rapport à DMM et les algorithmes génétiques sont les plus efficaces pour un taux de création de pièces supérieur à 1/20. Pour un intervalle de création de pièces compris entre [1/25, 1/20], l'électromagnétisme est la meilleure, pour un taux de création égale à 1/30 les colonies de fourmis sont les plus efficaces et le recuit simulé donne la meilleure utilisation de ces deux machines pour un taux de création inférieur à 1/30.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	26.3	26.53	34.23	34.7
Le recuit simulé	22.81	26.3	33.49	34.29
Les essaims particulaires	26.83	33.81	34.04	34.49
Les algorithmes génétiques	27.1	33.14	34.28	34.11
Recherche avec tabous	23.51	29.54	34.07	34.07
Electromagnétisme métaheuristique	27.13	29.66	34.27	34.11
DMM modifiée	17.06	23.09	23.72	23.96
DMM	8.23	19.53	23.46	24.3

Tableau 3.35: Le taux d'utilisation des machines FV_1 et FV_2 pour un taux de création de pièces= 1/20 (1/min).

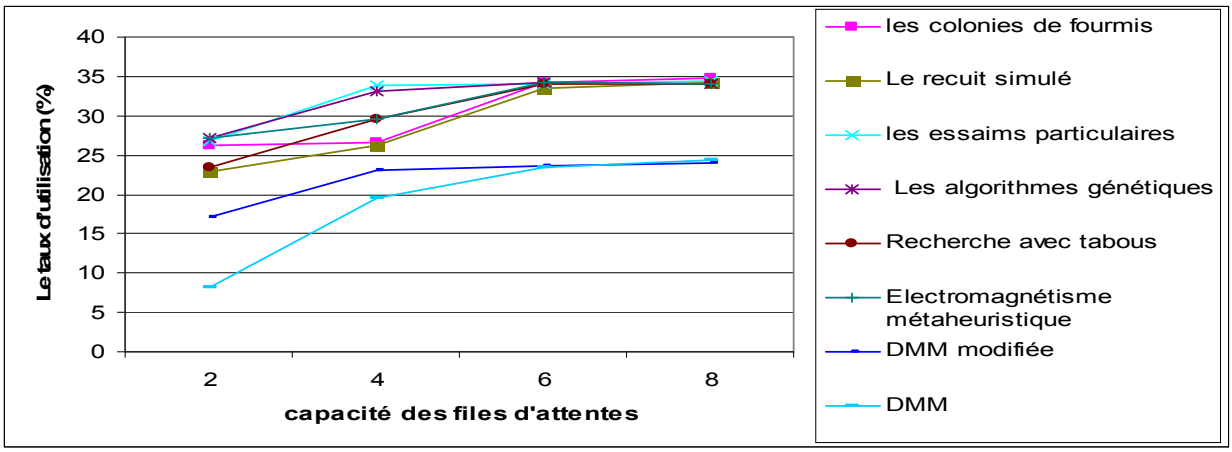

Figure 3.35: Le taux d'utilisation des machines FV_1 et FV_2 pour un taux de création de pièces= 1/20 (1/min).

La figure et tableau 3.35 nous montrent que si on compare les métaheuristiques pour un taux de création égale à 1/20, nous ne pouvons pas dire qu'une métaheuristique est la meilleure pour toutes les capacités, la meilleure utilisation des machines FV_1 et FV_2 est donnée par l'électromagnétisme pour une taille de file d'attente égale à 2, les essaims particulaires pour une taille égale à 4, les algorithmes génétiques pour une taille de file d'attente égale à 6 et les colonies de fourmis pour une taille de file d'attente égale à 8.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	48.29	54.58	63.9	76.13	80.44	80.2	80.24	80.5
Le recuit simulé	48.23	54.47	64.25	76.2	78.73	78.35	78.53	79.84
Les essaims particulaires	48.64	54.8	64.26	76.45	86.71	86.63	86.61	86.9
Les algorithmes génétiques	48.3	54.9	64.04	76.1	88.13	88.44	88.63	88.67
Recherche avec tabous	48.4	54.65	63.95	76.03	82.75	82.8	82.76	82.75
Electromagnétisme métaheuristique	48.30	54.60	64.01	75.95	86.16	84.6	84.65	84.96
DMM modifiée	49.88	56.94	66.78	77.16	77.77	78.13	78.5	78.75
DMM	49.96	57.23	66.81	42.71	33.02	39.3	31.21	34.83

Tableau 3.36: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 2.

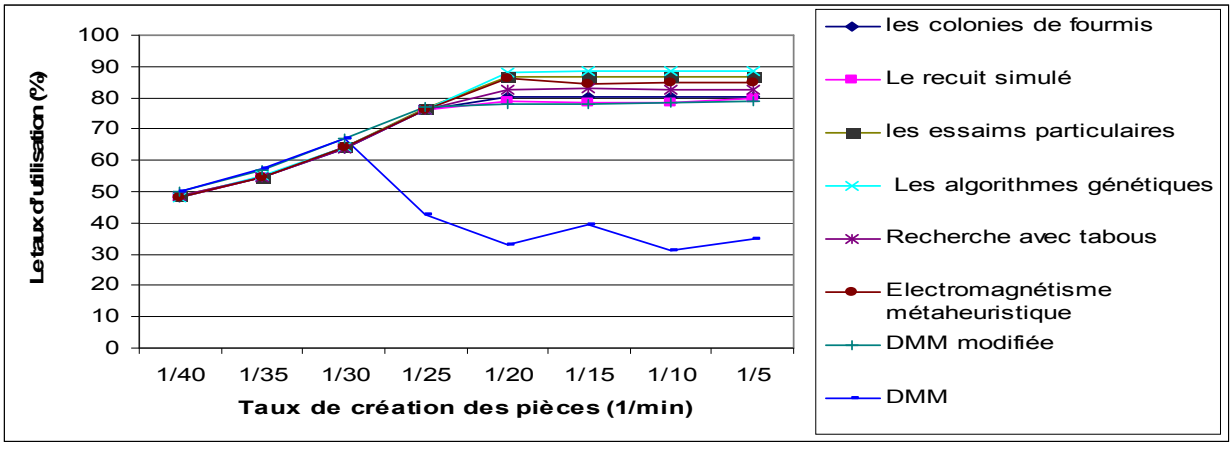

Figure 3.36: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 2.

A partir de la figure et tableau 3.36, nous pouvons remarquer que pour une capacité de file d'attente égale à 2, l'utilisation des machines T_1 et T_2 est plus importante pour les algorithmes génétiques par rapport à DMM, DMM modifiée et les autres métaheuristiques pour un taux de création de pièces supérieur à 1/25. Pour un taux de création égale 1/25, les essaims particulaires dépassent les autres règles et métaheuristiques. Pour un taux de création inférieur ou égale à 1/30, la règle DMM est la meilleure.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	1.292	1.415	1.673	1.954	2.068	2.065	2.065	2.069
Le recuit simulé	1.286	1.404	1.709	1.96	2.162	2.153	2.158	2.201
Les essaims particulaires	1.32	1.43	1.71	1.985	2.3	2.295	2.295	2.291
Les algorithmes génétiques	1.293	1.447	1.687	1.95	2.34	2.33	2.322	2.324
Recherche avec tabous	1.303	1.423	1.679	1.943	2.294	2.294	2.293	2.297
Electromagnétisme métaheuristique	1.293	1.417	1.685	1.935	2.226	2.27	2.273	2.28
DMM modifiée	1.47	1.73	2.72	22.21	25.27	24.31	23.55	23.73
DMM	1.48	1.77	2.56	2.98	2.88	2.79	2.28	3.01

Tableau 3.37: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 2.

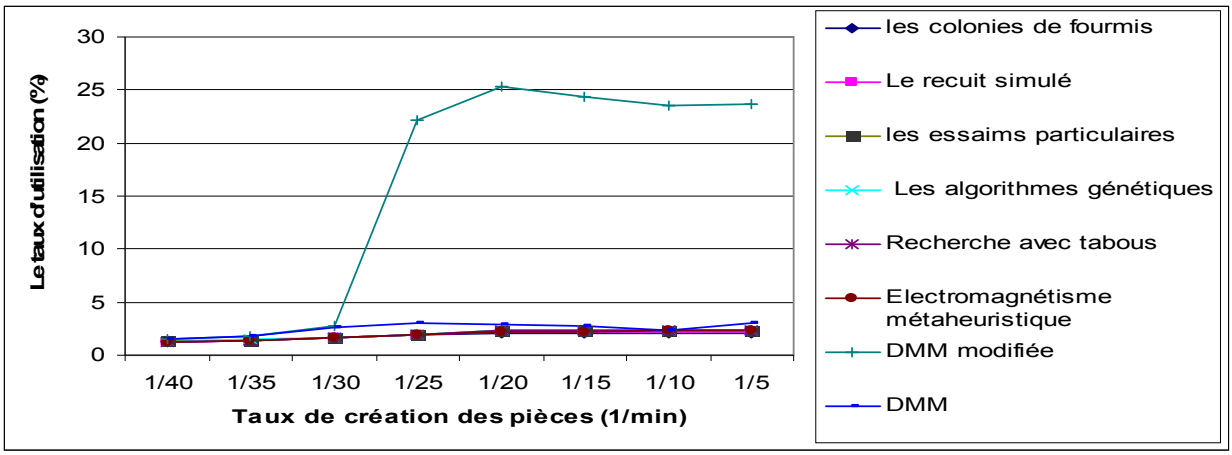

Figure 3.37: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 2.

Comme nous l'avons bien remarqué (voir la figure et tableau 3.37) l'utilisation de la machine TP est meilleure pour la règle DMM modifiée par rapport à la règle DMM et les métaheuristiques pour un taux de création supérieur à 1/35, et dans l'intervalle [1/40, 1/35] la règle DMM est la meilleure. Si on compare les métaheuristiques, les algorithmes génétiques donnent une utilisation de la machine TP supérieure à celle des autres métaheuristiques pour un système saturé, et qu'en dessous de taux de création 1/20 l'utilisation est meilleure pour les essaims particulaires sauf pour un taux de création égale à 1/35 où les algorithmes génétiques sont les meilleurs.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	22.92	25.12	29.7	34.7	36.72	36.66	36.67	36.74
Le recuit simulé	22.82	24.93	30.32	34.8	38.34	38.19	38.28	39.02
Les essaims particulaires	23.52	25.51	30.34	35.24	40.82	40.73	40.72	40.66
Les algorithmes génétiques	22.93	25.69	29.94	34.62	41.63	41.36	41.22	41.25
Recherche avec tabous	23.11	25.26	29.8	34.5	40.68	40.68	40.65	40.73
Electromagnétisme métaheuristique	22.94	25.16	29.9	34.37	40.18	40.28	40.33	40.44
DMM modifiée	17.99	20.49	23.91	24.52	24.02	24.43	24.64	24.49
DMM	18.08	20.58	23.95	15.31	11.67	14.1	11.11	12.29

Tableau 3.38: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente =2.

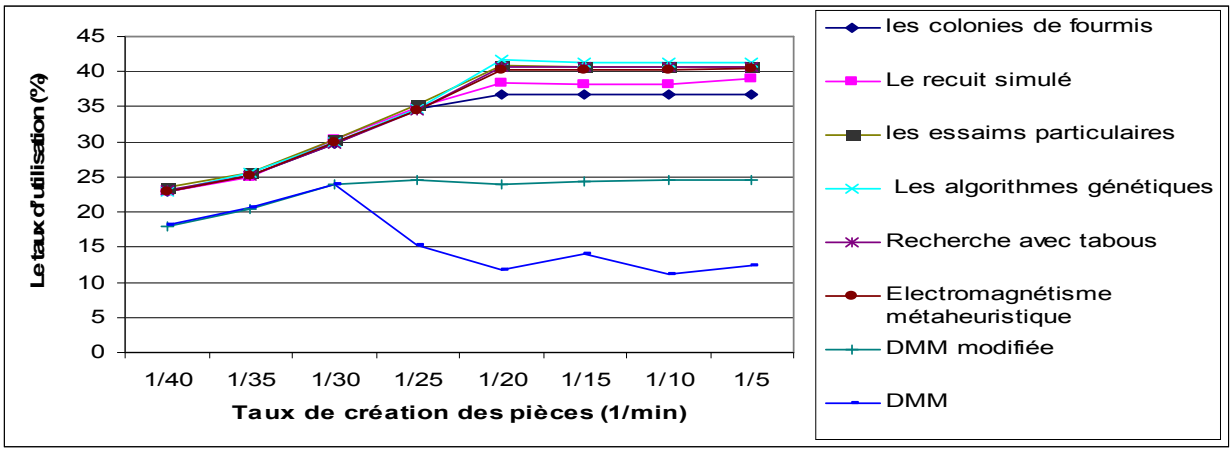

Figure 3.38: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente = 2.

Les figures et tableaux 3.38 montrent que pour une capacité de files d'attente égale à 2, si le taux de création des pièces est supérieur à 1/25, le taux d'utilisation des machines FH_1 et FH_2 est plus important pour les algorithmes génétiques par rapport aux autres métaheuristiques et les règles DMM et DMM modifiée, et qu'en dessous de taux de création 1/20 l'utilisation de ces machines est meilleure pour les essaims particulaires sauf pour un taux de création égale à 1/35 où les algorithmes génétiques sont les meilleurs.

3.4.2.5. Le taux d'utilisation de l'AGV

Les figures 3.39 et 3.40 nous montrent que pour un système saturé, de petites files d'attente (inférieur à 6) et avec la présence de pannes, le taux d'utilisation de l'AGV est plus important pour les algorithmes génétiques. Si on augmente les files d'attente la règle DMM modifiée est la meilleure.

Pour un système non saturé, l'utilisation de l'AGV est meilleure pour les essaims particulaires sauf pour un taux de création égale à 1/35 où les algorithmes génétiques dépassent les autres.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	17.39	19.57	22.95	27.27	28.82	28.74	28.75	28.84
Le recuit simulé	17.37	19.52	23.13	27.3	28.46	28.33	28.3	28.88
Les essaims particulaires	17.57	19.68	23.13	27.42	31.2	31.16	31.15	31.24
Les algorithmes génétiques	17.4	19.73	23.02	27.25	31.78	31.79	31.83	31.85
Recherche avec tabous	17.45	19.61	22.97	27.21	29.96	29.98	29.96	29.97
Electromagnétisme métaheuristique	17.4	19.58	23	27.17	30.96	30.49	30.51	30.62
DMM modifiée	14.98	17.55	21.19	24.28	24.07	24.54	24.67	24.54
DMM	15	17.61	21.06	14.2	10.94	12.98	10.41	11.59

Tableau 3.39: Le taux d'utilisation de l'AGV pour une capacité de file d'attente =2.

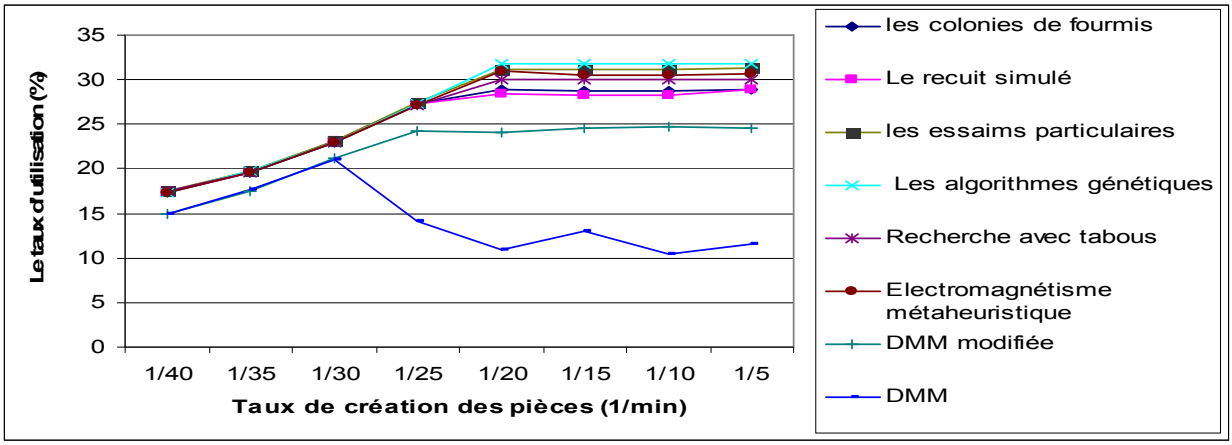

Figure 3.39: Le taux d'utilisation de l'AGV pour une capacité de file d'attente =2.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	28.82	31.43	32.88	32.69
Le recuit simulé	28.46	30.7	32.71	32.85
Les essaims particulaires	31.2	32.68	32.95	32.77
Les algorithmes génétiques	31.78	32.82	32.86	32.93
Recherche avec tabous	29.96	32.46	32.77	32.94
Electromagnétisme métaheuristique	30.96	31.66	32.86	32.92
DMM modifiée	24.07	32.46	33.58	34.08
DMM	10.94	25.28	30.66	31.68

Tableau 3.40: Le taux d'utilisation de l'AGV pour un taux de création de pièces =1/20 (1/min).

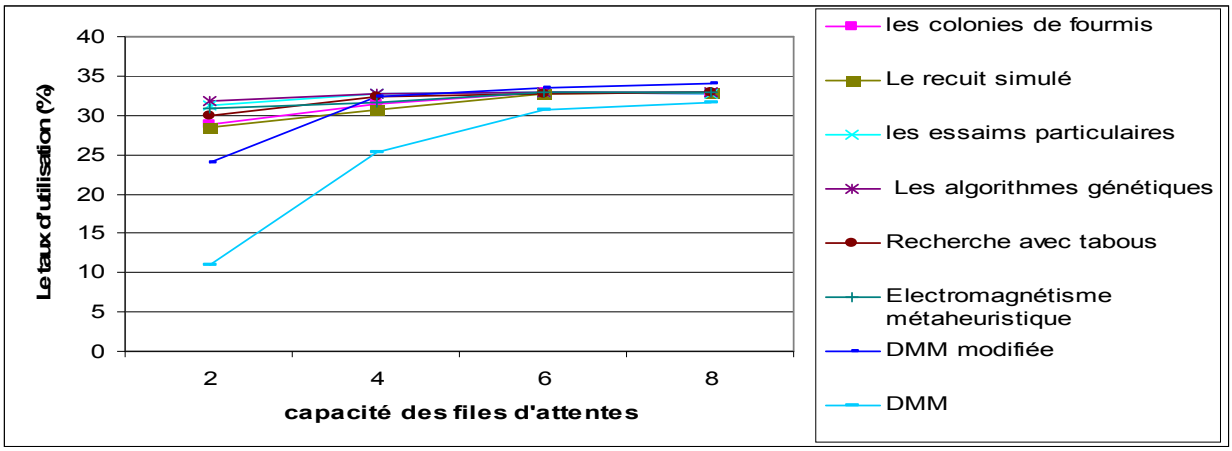

Figure 3.40 : Le taux d'utilisation de l'AGV pour un taux de création de pièces =1/20 (1/min).

3.5. Conclusion

Dans ce chapitre qui est une perspective du travail présenté à ICEEDT' 08 **[Souier 08],** nous avons adapté les métaheuristiques étudiées pour résoudre le problème de sélection de routage en temps réel dans un FMS.

Ces approches ont été comparées avec les règles DMM et DMM modifiée, afin de tirer des idées sur l'efficacité de chaque métaheuristique, pour choisir la meilleure à travers la simulation.

Nous avons d'abord commencé par varier le temps d'arrivée (le taux de création) des pièces et garder la capacité des files d'attente fixe puis nous avons fixé le temps de création des pièces et nous avons varié les capacités de files d'attente. Ensuite, nous avons simulé le système avec les deux critères de simulation mentionnés, mais en ajoutant de pannes aux machines du système.

De cette simulation un certain nombre de conclusions peuvent être faites. Nous résumons ici, l'essentiel :

- Les résultats obtenus ont montré que la plupart des métaheuristiques que nous avons étudiée ont donné de meilleures performances pour le taux de production, le taux d'utilisation des machines et du système de transport pour un système flexible de production saturé surtout pour de petites capacités de files d'attente même avec la présence de pannes.
- Les résultats obtenus après la simulation montrent aussi que ces techniques permettent d'améliorer le temps de cycle pour les petites files d'attente (taille de files d'attente égale à 2)
- Ces techniques permettent d'améliorer les en-cours pour de petites files d'attente pour le cas sans panne, et de grandes files d'attente pour le cas avec panne.
- Nous pouvons remarquer que le système n'est pas saturé pour les métaheuristiques et saturé pour les règles DMM et DMM modifiée pour un taux de création de pièces égale à 1/20 ce qui veut dire que le système sature moins rapidement si on utilise les métaheuristiques.
- L'efficacité de ces métaheuristiques varie d'un objectif à un autre et d'un critère à un autre par exemple pour le taux de production les algorithmes génétiques sont les plus efficaces pour un taux de production supérieur à 1/20.
- Pour le temps de cycle, nous ne pouvons pas dire qu'une métaheuristique donne un temps de cycle meilleur que les autres pour toutes les capacités de files d'attente, mais toutes les métaheuristiques améliorent les temps de cycle obtenus par les règles DMM et DMM modifiée pour un système saturé et de petites files d'attente. Dans le cas sans panne, pour un système saturé et de petites files d'attente, les temps de cycle obtenus par les colonies de fourmis sont généralement inférieurs à ceux obtenus par les autres règles et métaheuristiques.
- Pour les en-cours, dans le cas sans panne, le recuit simulé est le meilleur pour un taux de création superieur à 1/25 et de petites files d'attente, pour de grandes files d'attente et un système saturé, la règle DMM est la plus efficace mais le recuit simulé et les colonies de fourmis sont meilleurs que les autres métaheuristiques. Dans le cas avec panne, nous ne

pouvons pas dire qu'une métaheuristique est la meilleure pour toutes les capacités de files d'attente.

- Pour le taux d'utilisation des machines, les algorithmes génétiques sont les plus efficaces pour un taux de création de pièces supérieur à 1/20 sauf la machine TP dont l'utilisation est meilleure pour DMM modifiée que pour la règle DMM et les métaheuristiques. Tandis que pour les machines FH_1 et FH_2 et pour de grandes files d'attente, la recherche tabou donne la meilleure utilisation pour ces deux machines.
- Pour le taux d'utilisation de l'AGV, les algorithmes génétiques sont les meilleurs pour un taux de création supérieur à 1/20 et de petites files d'attente.

Durant notre travail, nous avons supposé que si une pièce entre dans la station de chargement, on ne peut pas changer son routage, et nous n'avons aucune garantie que les résultats trouvés sont les optimaux donc ils peuvent être améliorés si on ré-ordonnance les pièces contenues dans la station de chargement, le chapitre suivant concerne l'adaptation des métaheuristiques pour résoudre le problème en temps réel sans et avec pannes des machines avec le ré-ordonnancement des pièces contenues dans la station de chargement avec les premières pièces de la file infinie.

Chapitre 4

Métaheuristiques pour la sélection de routages alternatifs en temps réel -avec ré ordonnancement de la station de chargement-

4.1. Introduction

En combinant les mérites de production flow shop et job shop, un FMS fournit une technologie prometteuse pour la production mi-volume, mi-variété. En raison de l'évolution de l'informatique répartie et de l'écoulement de la matière, les FMS sont difficiles à concevoir.

De ce point de vue les problèmes liées à la technologie des FMS sont relativement complexes comparés aux systèmes de production traditionnels. C'est la raison pour laquelle les problèmes d'ordonnancement dans ces systèmes sont généralement NP complet.

L'une des premières études sur l'ordonnancement des systèmes flexibles de production est le travail de Nof et al **[Nof 79]** où il démontre l'importance et l'effet des décisions d'ordonnancement sur les performances des systèmes de production. La définition classique et traditionnelle dit que l'ordonnancement est une activité faite en temps différé où les opérations de production sont ordonnancées avant le début de la production.

Mais les systèmes flexibles de production (FMS) peuvent changer rapidement de produits et de séquences de produits sans perdre leurs productivités, à condition qu'il soit possible d'obtenir le bon produit, palette, support, ou outil, à la bonne place au bon moment. Donc afin de prendre en compte les changements de l'état de l'atelier les décisions d'ordonnancement doivent être prise en temps réel.

Dans ce chapitre qui est une continuation du chapitre précédent, nous nous proposons d'analyser l'impact du ré-ordonnancement des pièces contenues dans la station de chargement sur les performances du système flexible de production si on utilise les métaheuristiques pour la sélection des routages alternatifs, nous commençons par l'adaptation des métaheuristiques à la résolution de ce problème en temps réel avec ré-ordonnancement de la station de chargement, ensuite nous procéderons à une comparaison de ces techniques à la base des critères et objectifs définis dans le chapitre précédent et nous présentons ainsi les résultats obtenus après plusieurs simulations du modèle en gardant les mêmes conditions opératoires prise en compte dans le chapitre précédent.

4.2. Adaptation des algorithmes des métaheuristiques

Durant le cas sans ré-ordonnancement, si une pièce entre dans la station de chargement on l'affecte à un routage selon son type et l'état du système à ce moment en utilisant les algorithmes présentés dans le chapitre précédent. Mais des qu'elle est dans la station de chargement on ne peut

pas changer son routage malgré que l'état du système peut changer avant qu'elle passe à la première machine du routage choisi, ce qui va influencer sur les performances du système de production.

Pour cela nous proposons de ré-ordonnancer les pièces (modifier leurs routages) existantes dans la station de chargement. Notre proposition vise à garder le même principe des algorithmes décrits précédemment, avec la permission de changer les routages des pièces qui sont à l'intérieur de la station de chargement avec les premières pièces de la file infinie à chaque passage d'une nouvelle pièce à cette station et en fonction du nouvel état de l'atelier. Pour ce faire, dans chaque algorithme nous effectuons à chaque itération les modifications suivantes :

- En plus de *n* premières pièces contenues dans la file infinie, la solution est composée de routages de pièces contenues dans la station de chargement.
- Les opérations de construction de nouvelles solutions tel que les opérateurs de croisement mutation, recherche local …, doivent prendre en considération toutes ces pièces.

4.3. Résultats et interprétations

Pour pouvoir analyser l'influence de ré-ordonnancement des pièces contenues dans la station de chargement sur les performances du système flexible de production lorsque les métaheuristiques sont utilisées pour la sélection des routages alternatifs et le système est en état de saturation, nous avons proposé des études en simulation avec des variations sur les critères du système étudié utilisés dans le chapitre précédent (le taux de création de pièces et la taille des files d'attente d'entrée et de sortie des stations).

Cette section est réservée à la présentation des résultats trouvés et leurs interprétations.

4.3.1. étude comparative sans introduction de pannes

4.3.1.1. Taux de production

Les graphes des figures 4.1 à 4.3 présentent le taux de sortie des pièces en fonction du taux de création des pièces et pour différentes capacités de files d'attente.

Les figures 4.4 et 4.5 présentent le taux de sortie des pièces du système en fonction des capacités de files d'attente et en maintenant le temps de création fixe.

Les graphes des figures 4.5 et 4.6 présentent le taux de sortie des pièces en fonction du taux de création des pièces obtenu par les algorithmes génétiques avec et sans ré-ordonnancement et pour différentes capacités de files d'attente.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.99	95.37	67.01	44.82	22.36
Le recuit simulé	99.99	99.99	99.99	99.99	94.95	62.45	41.86	20.98
Les essaims particulaires	99.99	99.99	99.99	99.99	99.04	71.34	47.63	23.78
Les algorithmes génétiques	99.99	99.99	99.99	99.99	99.99	72.26	48.03	24.08
Recherche avec tabous	99.99	99.99	99.99	99.99	99.99	66.07	44.04	22.05
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	98.94	67.76	45.29	22.70
DMM modifiée	99.99	99.99	99.98	99.71	84.47	60.73	41.67	21.15
DMM	99.99	99.99	99.97	81.4	24.65	32.05	15.43	8.88

Tableau 4.1 : Taux de sortie des pièces pour une capacité de file d'attente = 2.

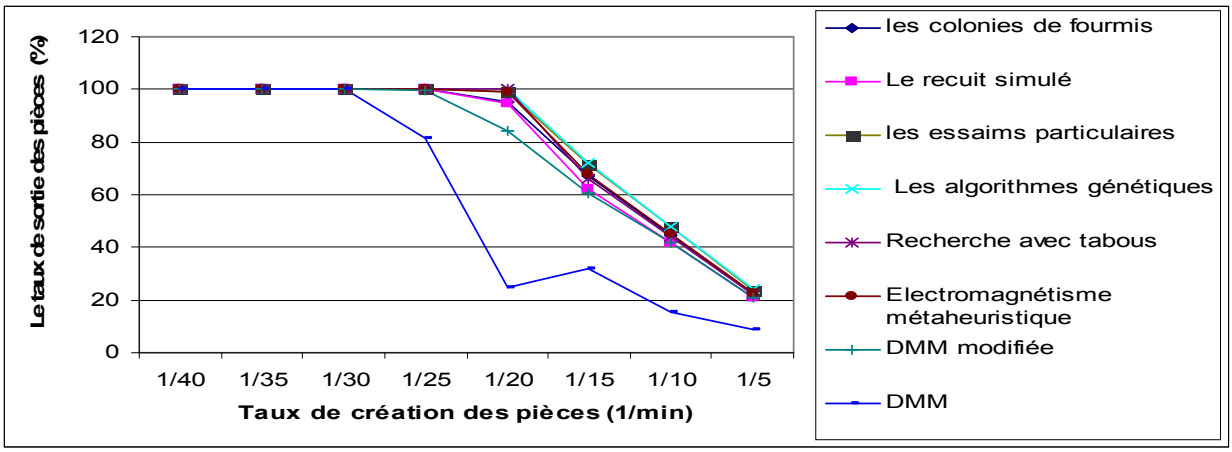

Figure 4.1 : Taux de sortie des pièces pour une capacité de file d'attente = 2.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.99	99.99	69.69	46.66	23.26
Le recuit simulé	99.99	99.99	99.99	99.99	99.99	68.32	45.76	22.95
Les essaims particulaires	99.99	99.99	99.99	99.99	99.99	73.26	48.84	24.43
Les algorithmes génétiques	99.99	99.99	99.99	99.99	99.99	72.72	48.67	24.26
Recherche avec tabous	99.99	99.99	99.99	99.99	99.99	71.52	47.67	23.83
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	99.99	69.43	46.45	23.22
DMM modifiée	99.99	99.99	99.99	99.99	93.11	69.79	46.77	23.44
DMM	99.99	99.99	99.99	99.99	91.94	53.33	41.3	18.42

Tableau 4.2 : Taux de sortie des pièces pour une capacité de file d'attente = 6.

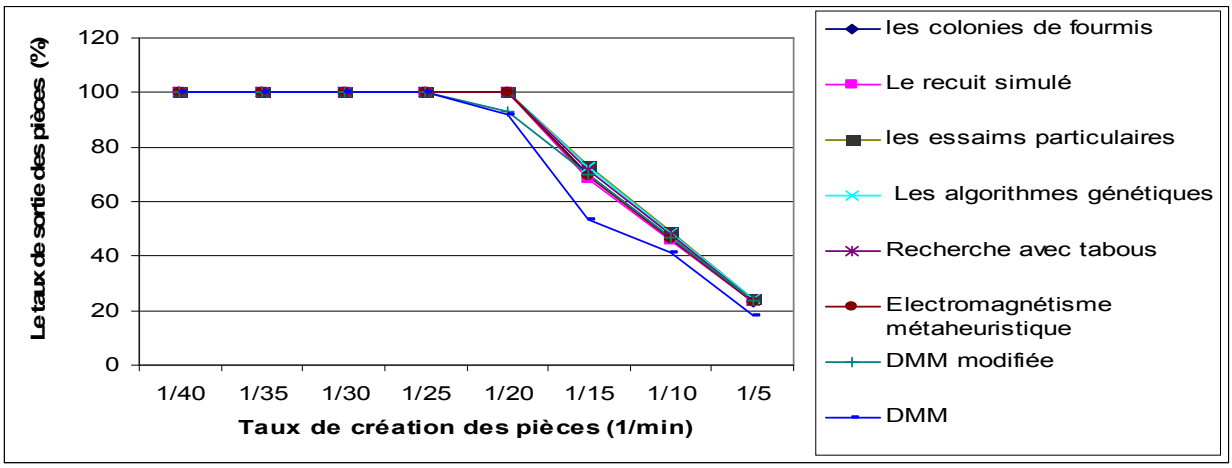

Figure 4.2 : Taux de sortie des pièces pour une capacité de file d'attente = 6.

Les figures et les tableaux 4.1, 4.2 montrent que pour des capacités de files d'attente inférieurs à 6 et un taux de création de pièces supérieur à 1/20, les algorithmes génétiques sont toujours les plus efficaces par rapport aux autres métaheuristiques et les méthodes DMM et DMM modifiée, dans cet intervalle les essaims particulaires sont les meilleurs pour une taille de files d'attente égale à 6. Pour un taux de création de pièces égale à 1/20, les algorithmes génétiques et la recherche avec tabous sont les meilleurs pour une capacité de files d'attente égale à 2. Et hors de ces intervalles les résultats obtenus par les métaheuristiques sont les mêmes.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.99	99.99	70.33	46.85	23.42
Le recuit simulé	99.99	99.99	99.99	99.99	99.99	70.29	47.05	23.81
Les essaims particulaires	99.99	99.99	99.99	99.99	99.99	73.79	49.18	24.61
Les algorithmes génétiques	99.99	99.99	99.99	99.99	99.99	72.84	48.72	24.28
Recherche avec tabous	99.99	99.99	99.99	99.99	99.99	72.33	48.15	24.09
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	99.99	70.63	47.11	23.62
DMM modifiée	99.99	99.99	99.99	99.99	94.25	70.83	47.32	23.51
DMM	99.99	99.99	99.99	99.99	95.71	71.35	47.3	24.01

Tableau 4.3 : Taux de sortie des pièces pour une capacité de file d'attente = 8.

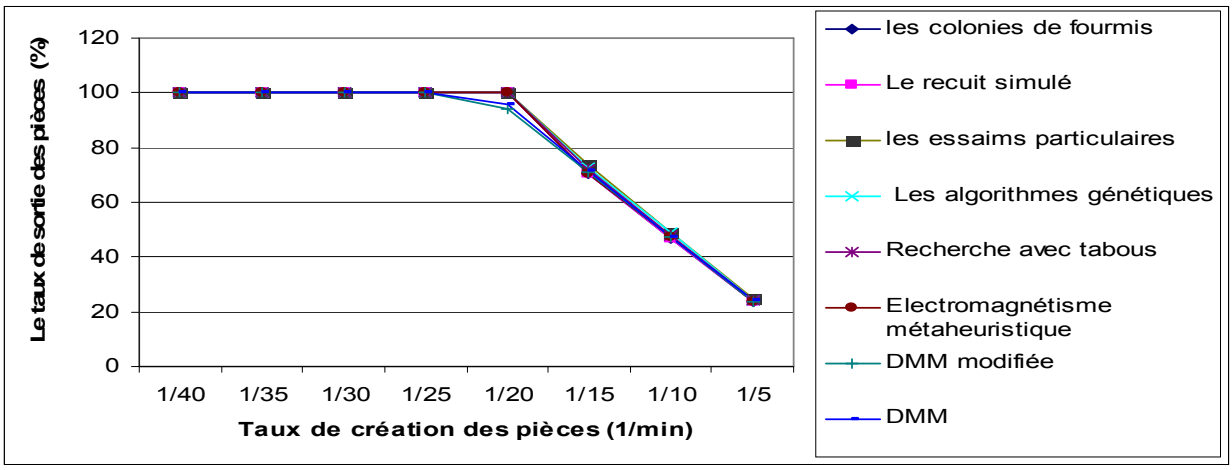

Figure 4.3 : Taux de sortie des pièces pour une capacité de file d'attente = 8.

La courbe et tableau 4.3 montrent que pour une capacité de files d'attente égale à 8 et un taux de création supérieur à 1/20, les essaims particulaires sont les meilleurs. Et hors de cet intervalle, les résultats concernant le taux de production obtenus par les métaheuristiques sont identiques.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	22.36	23.16	23.26	23.42
Le recuit simulé	20.98	22.01	22.95	23.81
Les essaims particulaires	23.78	24.16	24.43	24.61
Les algorithmes génétiques	24.08	24.17	24.26	24.28
Recherche avec tabous	22.05	23.22	23.83	24.09
Electromagnétisme métaheuristique	22.70	23.07	23.22	23.62
DMM modifiée	21.15	22.58	23.44	23.51
DMM	8.88	15.67	18.42	24.01

Tableau 4.4 : Taux de sortie des pièces pour un taux de création de pièces = 1/5 (1/min).

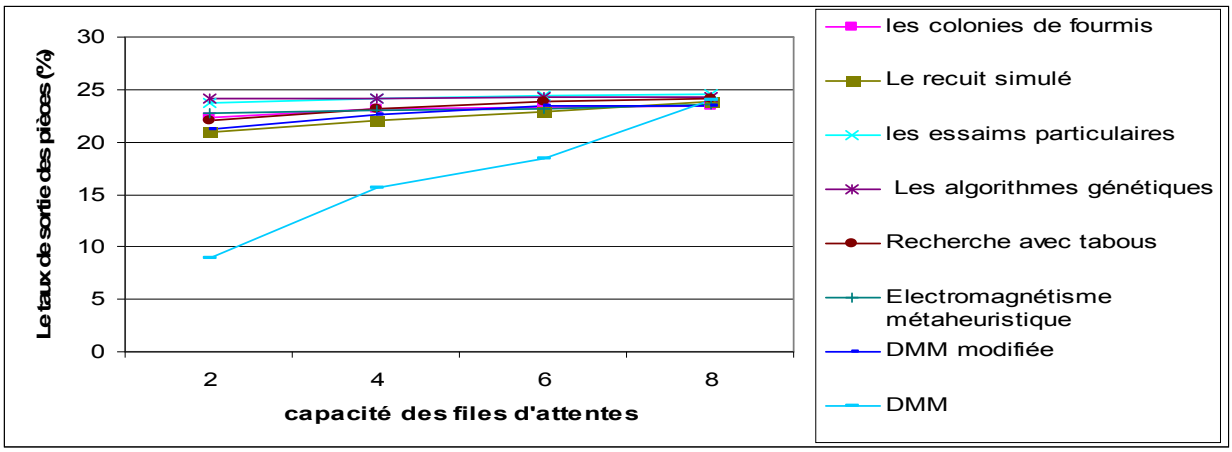

Figure 4.4 : Taux de sortie des pièces pour un taux de création de pièces = 1/5 (1/min).

La figure et tableau 4.4 nous montrent qu'en variant les capacités de files d'attente et en maintenant le temps de création fixe, les algorithmes génétiques sont les meilleurs pour un système très saturé et de petites files d'attente et si leur capacité augmente les essaims particulaires sont les meilleurs.

Comme le chapitre précédent, nous avons remarqué que pour un taux de création égale à 1/20 le système sature moins rapidement si on utilise les métaheuristiques.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	99.99	99.99	99.99	99.99	98.44	71.08	47.25	23.7
Avec ré-ordonnancement	99.99	99.99	99.99	99.99	99.99	72.26	48.03	24.08

Tableau 4.5 : Taux de sortie des pièces obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

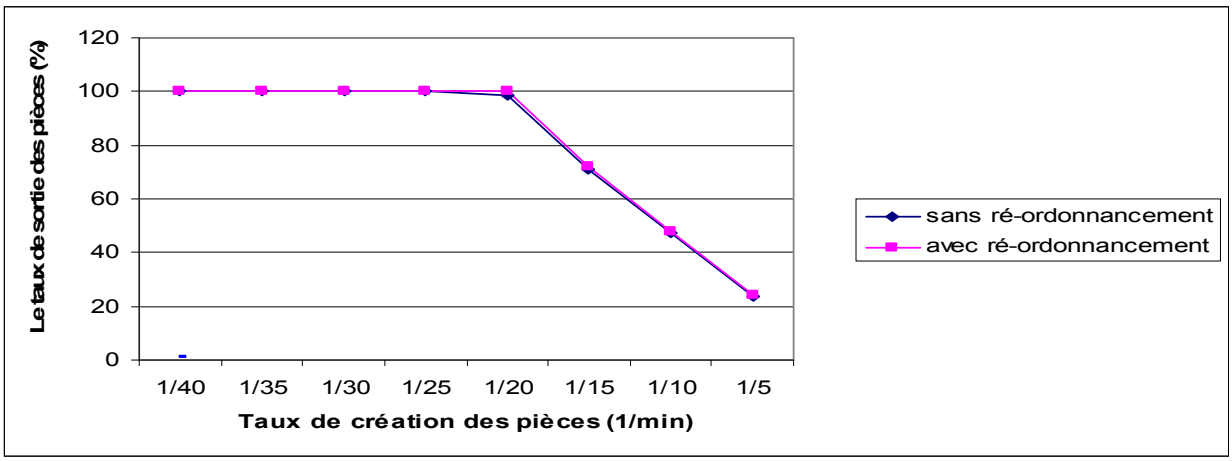

Figure 4.5 : Taux de sortie des pièces obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.5 nous montrent que le taux de sortie des pièces obtenu par les algorithmes génétiques avec ré-ordonnancement est supérieur à celui trouvé sans ré-ordonnancement pour un taux de création de pièces supérieur à 1/25 et une capacité de files d'attente égale à 2. Hors de cet intervalle, les résultats obtenus dans les deux cas sont identiques.

A partir de figures, les tableaux 4.1, 4.2, 4.5 et les résultats présentés dans le chapitre précédent, nous pouvons remarquer que pour de petites files d'attente et des taux de création de pièces supérieurs ou égales à 1/15, les résultats obtenus par les métaheuristiques avec ré-ordonnancement de pièces contenues dans la station de chargement sont meilleurs que ceux obtenus sans ré-ordonnancement.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	99.99	99.99	99.99	99.99	99.99	72.76	48.59	24.24
Avec ré-ordonnancement	99.99	99.99	99.99	99.99	99.99	72.84	48.72	24.28

Tableau 4.6 : Taux de sortie des pièces obtenu par les algorithmes génétiques pour une capacité de file d'attente = 8.

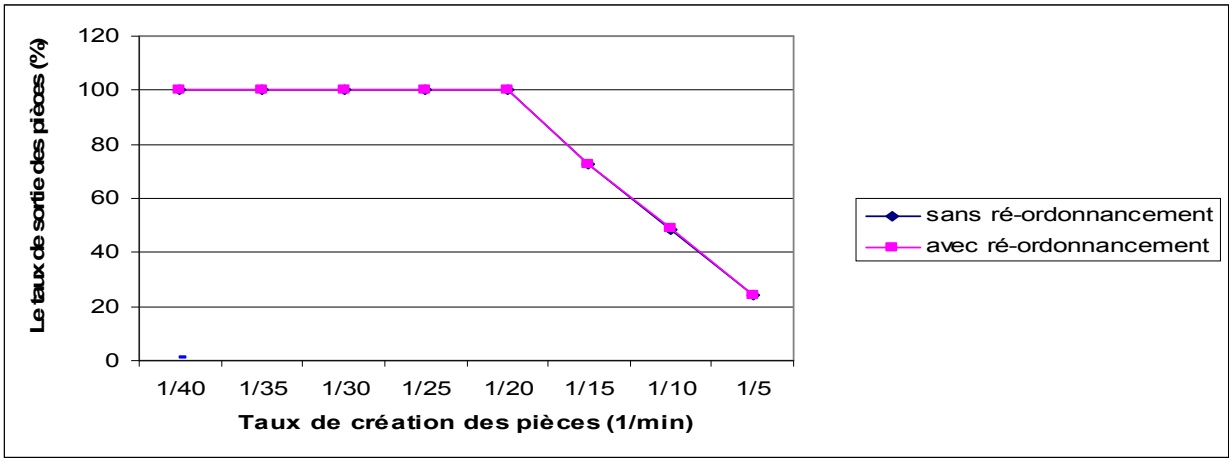

Figure 4.6 : Taux de sortie des pièces obtenu par les algorithmes génétiques pour une capacité de file d'attente = 8.

La figure et le tableau 4.6 nous montrent que même si la capacité de files d'attente augmente, le taux de sortie de pièces obtenu par les algorithmes génétiques avec ré-ordonnancement est toujours supérieur à celui trouvé sans ré-ordonnancement pour un système saturé. Pour un système non saturé, les résultats obtenus dans les deux cas sont identiques.

Pour de grandes capacités de files d'attente et un taux de création de pièces supérieur à 1/20, les résultats obtenus par les métaheuristiques étudiées restent meilleurs avec ré-ordonnancement des pièces contenues dans la station de chargement que ceux obtenus dans le chapitre précédent (sauf l'électromagnétisme avec une taille de file = 8). Et qu'en dessous du taux de création 1/15, le taux de production est pratiquement le même pour chaque métaheuristique dans les deux cas (voir les figures et les tableaux 3.5, 4.3 et 4.6).

4.3.1.2. Le temps de cycle

Dans cette sous section, nous allons montrer l'impact du taux de création des pièces, la capacité des files d'attente et le ré-ordonnancement de la station de chargement sur le temps de cycle.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	93.99	82.64	90.16	96.2	139.2	170.2	166.8	170.6
Le recuit simulé	89	90.8	89.5	96.5	135.1	167.2	168.7	173.4
Les essaims particulaires	94.6	82.3	89.8	98.4	129.7	172.2	166.7	173.8
Les algorithmes génétiques	92.4	83.1	90	98.6	115.5	171.8	169.8	170.7
Recherche avec tabous	93.3	81.3	90.5	95.7	130.8	167.2	173.2	171
Electromagnétisme métaheuristique	89	82.3	88.9	97.6	121.5	172.9	169.9	170.1
DMM modifiée	81.9	87.8	101.6	155.8	203.3	204.7	207.2	204.2
DMM	81.3	88.1	102.1	153.7	187.8	176.5	186.3	187.2

Tableau 4.7: Temps de cycle pour une capacité de file d'attente = 2.

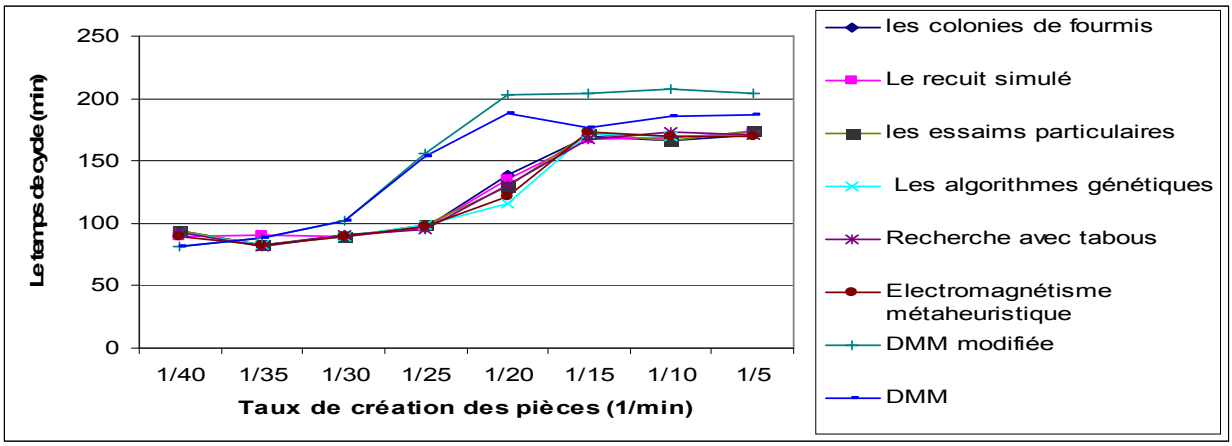

Figure 4.7: Temps de cycle pour une capacité de file d'attente = 2

La courbe de figure 4.7 et le tableau 4.7 montrent que pour une taille de file d'attente égale à 2, les temps de cycle de la méthode DMM et DMM modifiée sont supérieurs à ceux obtenus par toutes les métaheuristiques étudiées pour un taux de création de pièces supérieur à 1/35, mais nous ne pouvons pas dire qu'une métaheuristique est la meilleure pour tous les taux.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	89.1	84.6	90.3	98.6	123.9	242.9	241	240
Le recuit simulé	90.3	84.6	88.7	98	116	248.7	250.8	249.3
Les essaims particulaires	92	83	91.6	96.7	125.4	255.2	254.1	252.5
Les algorithmes génétiques	89.2	82.5	88.5	98.9	132.4	259.1	256.9	256.7
Recherche avec tabous	90.9	83.4	89.8	96.9	130.9	250.8	249.6	252.3
Electromagnétisme métaheuristique	91.4	82	86.2	93.2	121.8	251.2	247.6	256
DMM modifiée	82.02	86.63	98.07	130.0	309.8	310.4	309.9	311.8
DMM	82.03	86.69	98.07	129.9	245.2	247.2	231.0	246.3

Tableau 4.8: Temps de cycle pour une capacité de file d'attente = 4.

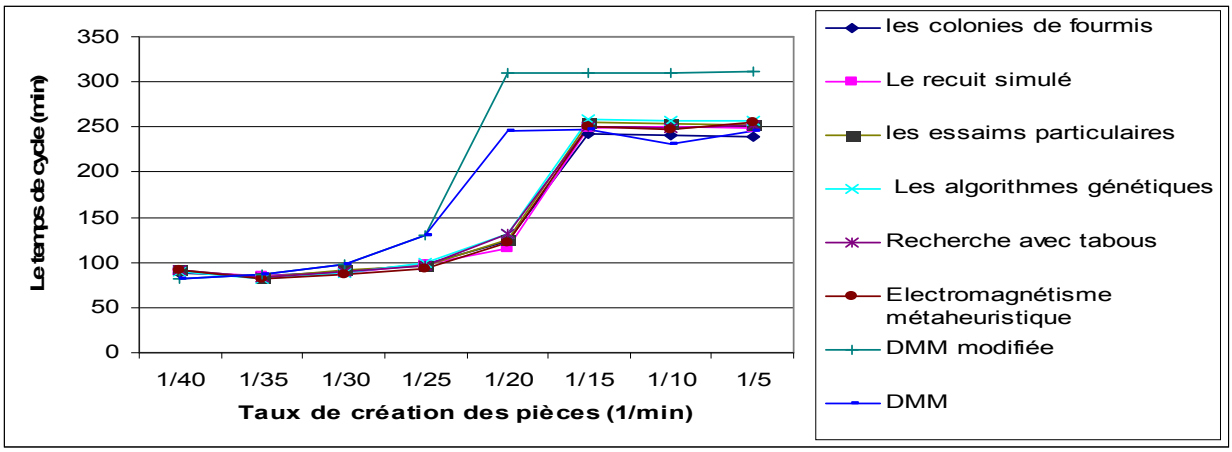

Figure 4.8: Temps de cycle pour une capacité de file d'attente = 4.

La courbe de figure 4.8 et le tableau 4.8 montrent que pour une taille de file d'attente égale à 4 et un taux de création supérieur à 1/20, les temps de cycle obtenus par DMM et les colonies de fourmis sont meilleurs que ceux de DMM modifiée et les autres métaheuristiques, pour un taux de création égale à 1/20 le recuit simulé est le plus efficace, l'électromagnétisme est la meilleure pour un taux compris entre 1/25 et 1/35 et DMM modifiée pour un taux égale 1/40.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	95	82.8	92.3	93.3	120.1	307.2	311.6	309.5
Le recuit simulé	87.8	83.7	90.9	98.4	120.9	330.9	331.7	321.9
Les essaims particulaires	89.9	84.3	90.9	95.9	123.6	331.9	340.9	338.4
Les algorithmes génétiques	90.2	83.6	89.6	97.8	122.3	342	338	339.9
Recherche avec tabous	94.4	83.6	89.6	97.1	125.3	341.5	340.4	338.4
Electromagnétisme métaheuristique	94.3	79.9	86.9	99.7	126.4	336.4	333.7	336.9
DMM modifiée	81.26	86.6	97.57	126.8	416.7	416.7	414.7	414.3
DMM	81.26	86.67	97.65	127.2	276.9	279.2	269.9	283.9

Tableau 4.9: Temps de cycle pour une capacité de file d'attente = 6.

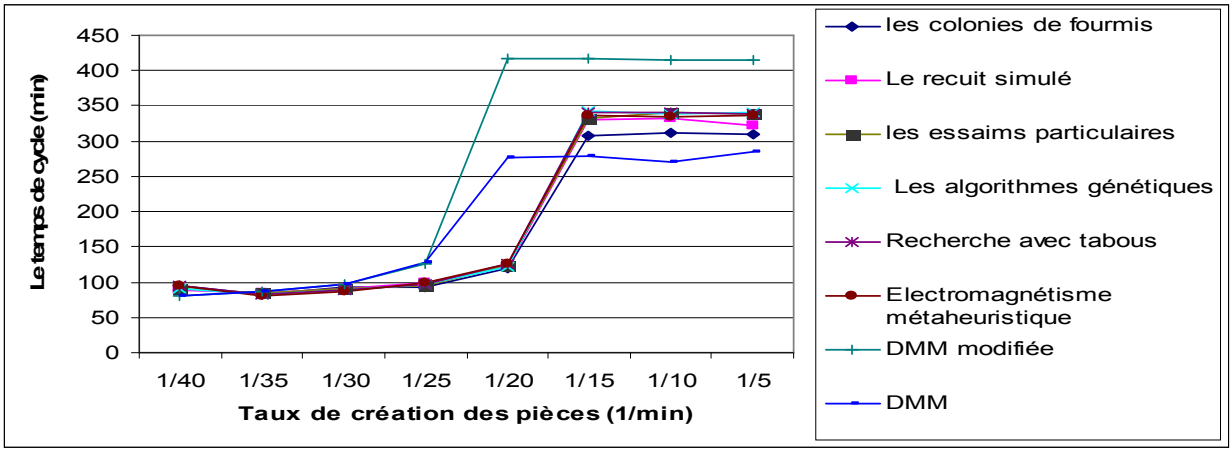

Figure 4.9: Temps de cycle pour une capacité de file d'attente = 6.

La courbe du figure et tableau 4.9 montrent que pour une taille de file d'attente supérieure à 4, les temps de cycle de la méthode DMM sont inférieurs à ceux des métaheuristiques étudiées pour

un taux de création de pièces supérieur à 1/20, dans cet intervalle les colonies de fourmis sont les plus efficaces si on les compare avec les autres métaheuristiques.

Dans l'intervalle ([1/35, 1/20]), les temps de cycles obtenus par toutes les métaheuristiques sont meilleurs que ceux de DMM et DMM modifiée, mais nous ne pouvons pas dire qu'une métaheuristique est la meilleure. DMM et DMM modifiée sont les meilleurs pour un taux de création de pièces égale à 1/40.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	170.6	240	309.5	376.4
Le recuit simulé	173.4	249.3	321.9	417.1
Les essaims particulaires	173.8	252.5	338.4	423
Les algorithmes génétiques	170.7	256.7	339.9	422.9
Recherche avec tabous	171	252.3	338.4	416.6
Electromagnétisme métaheuristique	170.1	256	336.9	411.3
DMM modifiée	204.2	311.8	414.3	551.1
DMM	187.2	246.3	283.9	321.3

Tableau 4.10: Temps de cycle pour un taux de création de pièces =1/5 (1/min)

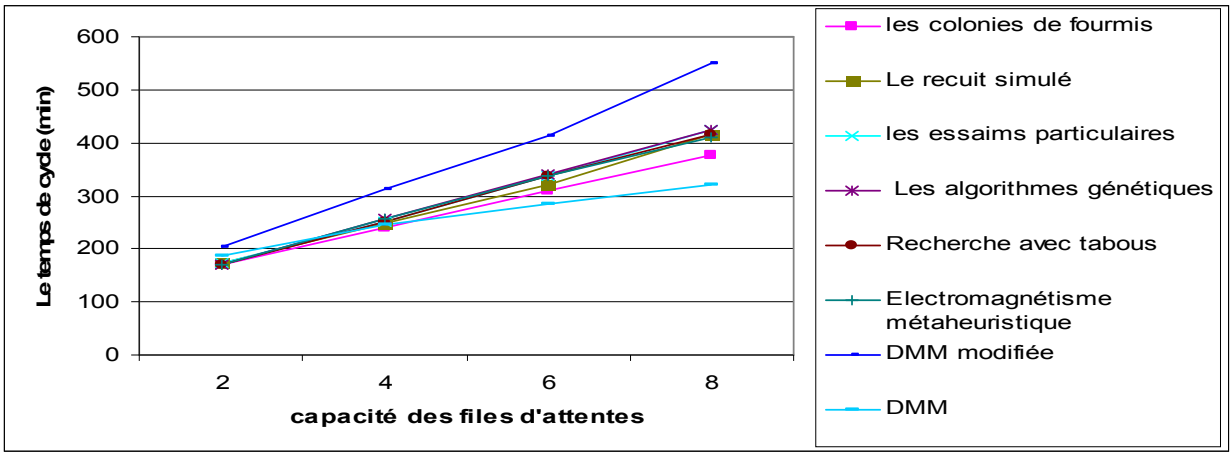

Figure 4.10: Temps de cycle pour un taux de création de pièces =1/5 (1/min)

La figure et le tableau 4.10 montrent que si on varie les capacités de files d'attente, en gardant le taux de création fixe, nous pouvons dire que les métaheuristiques ont pu améliorer le temps de cycle pour un système très saturé et de petites files d'attente où les colonies de fourmis donnent les meilleurs temps de cycle.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	89.6	81.8	88.6	98.7	132.7	169.8	168.9	168.1
Avec ré-ordonnancement	92.4	83.1	90	98.6	115.5	171.8	169.8	170.7

Tableau 4.11 : Temps de cycle obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

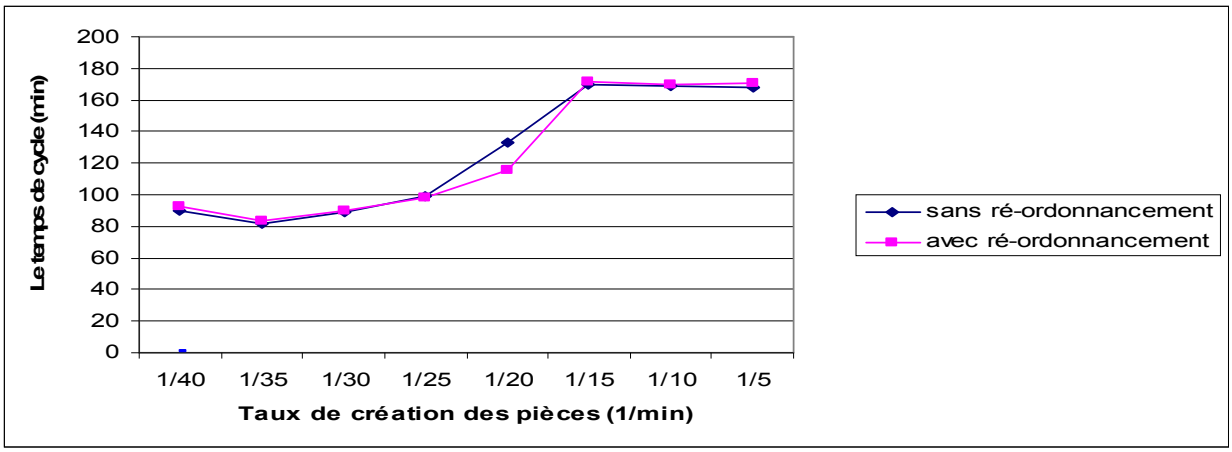

Figure 4.11 : Temps de cycle obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

Capacité de file d'attente	2	4	6	8
Sans ré-ordonnancement	168.1	249.9	338.1	417
Avec ré-ordonnancement	170.7	256.7	339.9	422.9

Tableau 4.12 : Temps de cycle obtenu par les algorithmes génétiques un taux de création de pièces =1/5 (1/min)

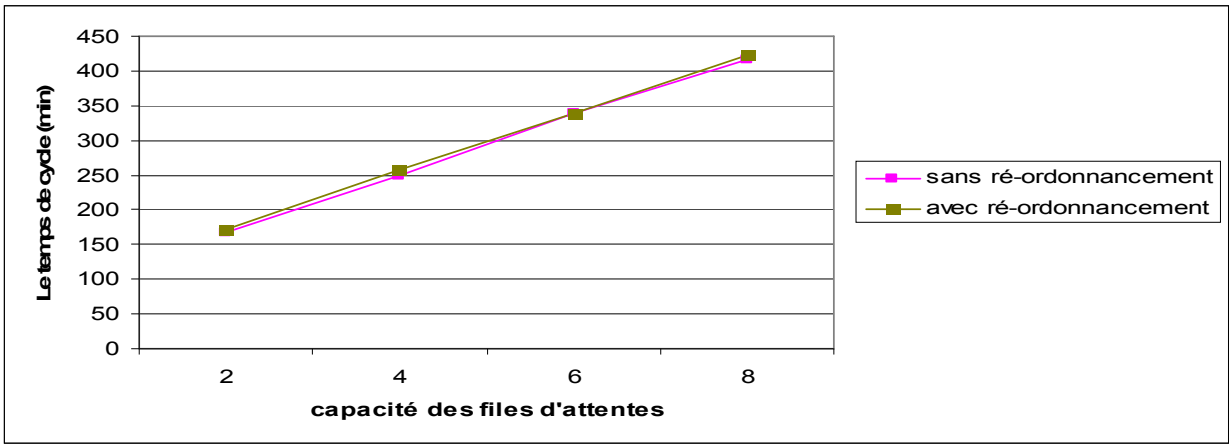

Figure 4.12 : Temps de cycle obtenu par les algorithmes génétiques pour un taux de création de pièces =1/5 (1/min)

Les figures 4.11 et 4.12 nous montrent que le temps de cycle obtenu par les algorithmes génétiques avec ré-ordonnancement est supérieur à celui trouvé sans ré-ordonnancement pour un taux d'arrivée de pièces supérieur à 1/20, même si on varie la capacité de files d'attente.

Pour chaque métaheuristique, si on compare le temps de cycle avec et sans ré-ordonnancement de la station de chargement, nous pouvons remarquer l'augmentation du temps de cycle dans le cas avec ré-ordonnancement pour la plupart des cas d'un système saturé (voir les figures 4.7 à 4.12 et les résultats du chapitre précédent).

4.3.1.3. Les en-cours

La figure 4.13 nous montre que les en-cours obtenus par les métaheuristiques sont inférieurs à ceux obtenus par les règles DMM et DMM modifiée, pour un système saturé et de petites files d'attente.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	3.9	4.05	4.47	5.23	7.1	8.09	8.09	8.1
Le recuit simulé	3.91	4.06	4.45	5.29	7.66	8.13	8.17	8.16
Les essaims particulaires	3.91	4.04	4.47	5.25	7.35	8.92	8.93	8.95
Les algorithmes génétiques	3.88	4.03	4.45	5.22	7.65	9.01	9.02	9.02
Recherche avec tabous	3.89	4.03	4.45	5.24	7.38	8.34	8.36	8.37
Electromagnétisme métaheuristique	3.88	4.03	4.47	5.28	7.18	8.67	8.68	8.65
DMM modifiée	6.79	12.95	15.79	19.01	17.84	18.19	16.86	18.12
DMM	8.37	7.79	14.33	15.96	15.82	15.4	12.82	13.67

Tableau 4.13: Les en-cours pour une capacité de file d'attente = 2.

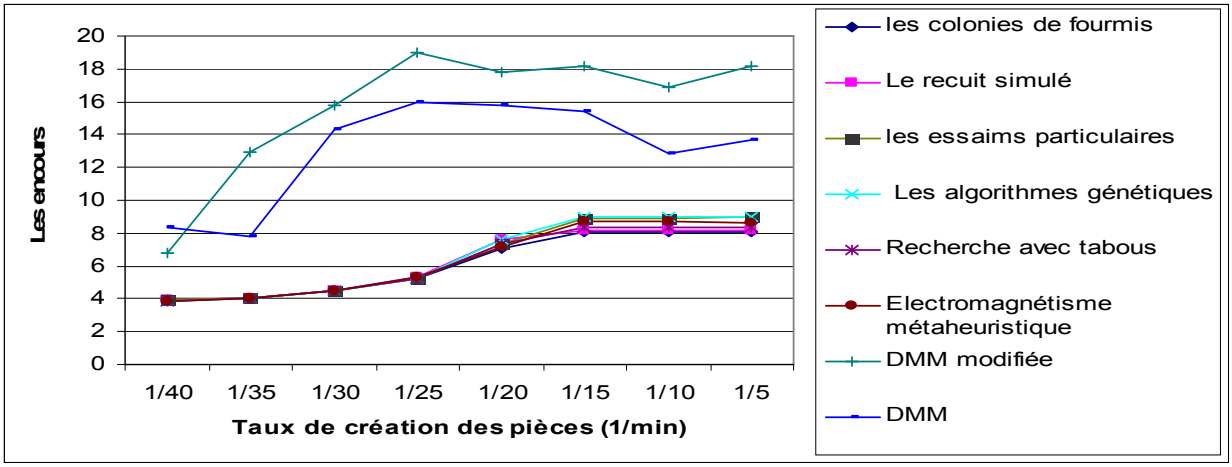

Figure 4.13: Les en-cours pour une capacité de file d'attente = 2.

La figure et le tableau 4.13 montrent aussi que pour une taille de files d'attente égale à 2, les colonies de fourmis sont les plus efficaces pour un taux de création supérieur à 1/25, et hors de cet intervalle il n'existe pas une métaheuristique qui donne les meilleurs résultats pour tous les taux.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	7.9	8.04	8.47	9.28	10.98	18.24	18.32	18.3
Le recuit simulé	7.89	8.04	8.48	9.21	11.5	19.08	19.17	19.25
Les essaims particulaires	7.9	8.04	8.47	9.23	11.34	19.99	20.17	20.09
Les algorithmes génétiques	7.89	8.03	8.46	9.27	11.1	20.07	20.05	20.07
Recherche avec tabous	7.89	8.02	8.41	9.25	11.16	19.77	19.79	19.78
Electromagnétisme métaheuristique	7.91	8.05	8.46	9.29	11.25	19	19.06	19.13
DMM modifiée	3.79	4.8	6.89	46.96	50.99	49.87	43.81	42.46
DMM	3.78	4.8	6.83	14	15.4	14.1	15.3	11.4

Tableau 4.14: Les en-cours pour une capacité de file d'attente = 6.

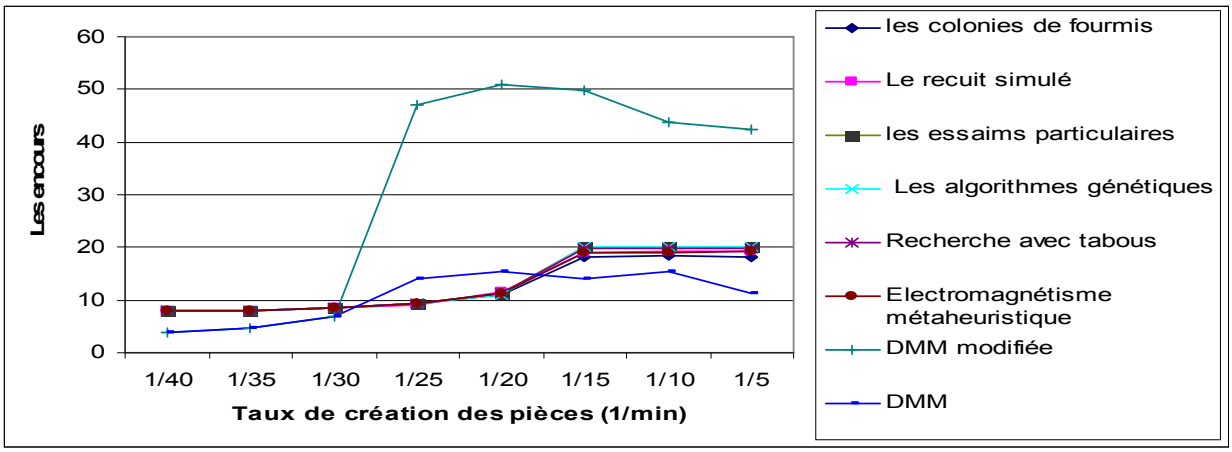

Figure 4.14: Les en-cours pour une capacité de file d'attente = 6.

A partir des figures et tableaux 4.14, nous pouvons remarquer que si la taille des files d'attente égale à 6, le nombre de pièces restantes obtenu par la règle DMM est inférieur à ceux obtenus par la règle DMM modifiée et les métaheuristiques, pour un taux de création compris entre [1/15, 1/5] et [1/40, 1/30]. Et hors de cet intervalle (dans l'intervalle [1/25, 1/20]), les colonies de fourmis sont les meilleures pour un taux égale à 1/20 et le recuit simulé est le plus efficace pour un taux égale à 1/25.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	3.9	4.04	4.46	5.23	7.45	8.9	8.85	8.85
Avec ré-ordonnancement	3.88	4.03	4.45	5.22	7.65	9.01	9.02	9.02

Tableau 4.15 : Les en-cours obtenus par les algorithmes génétiques pour une capacité de file d'attente = 2.

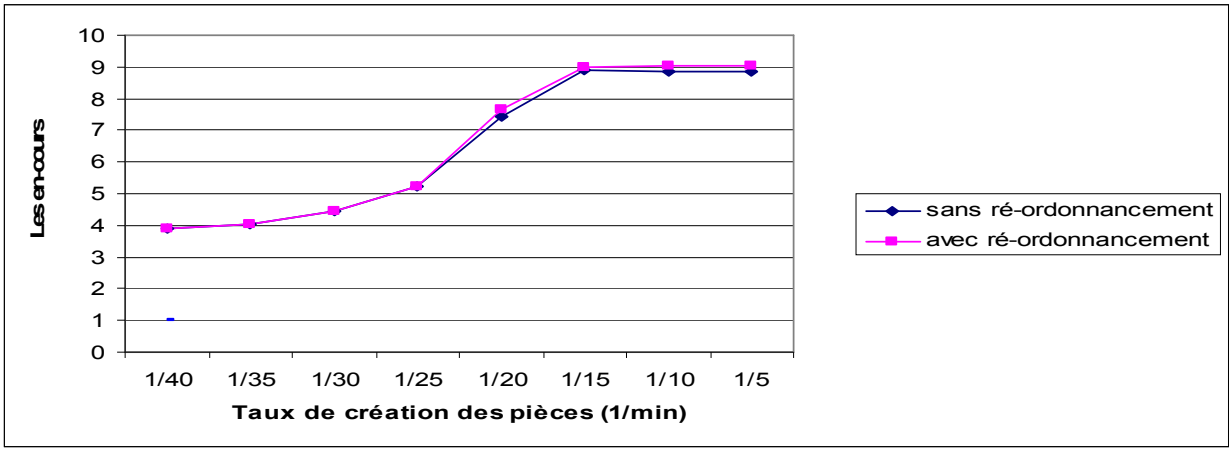

Figure 4.15 : Les en-cours obtenus par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.15 nous montrent que le nombre de pièces restantes obtenu par les algorithmes génétiques avec ré-ordonnancement est supérieur à celui trouvé sans ré-ordonnancement pour un taux de création de pièces supérieur à 1/25, et une capacité de files d'attente égale à 2. Hors de cet intervalle, les résultats obtenus dans les deux cas sont presque identiques.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	7.89	8.02	8.45	9.2	11.31	18.46	18.56	18.56
Avec ré-ordonnancement	7.9	8.04	8.47	9.28	10.98	18.24	18.32	18.3

Tableau 4.16 : Les en-cours obtenus par les colonies de fourmis pour une capacité de file d'attente = 6.

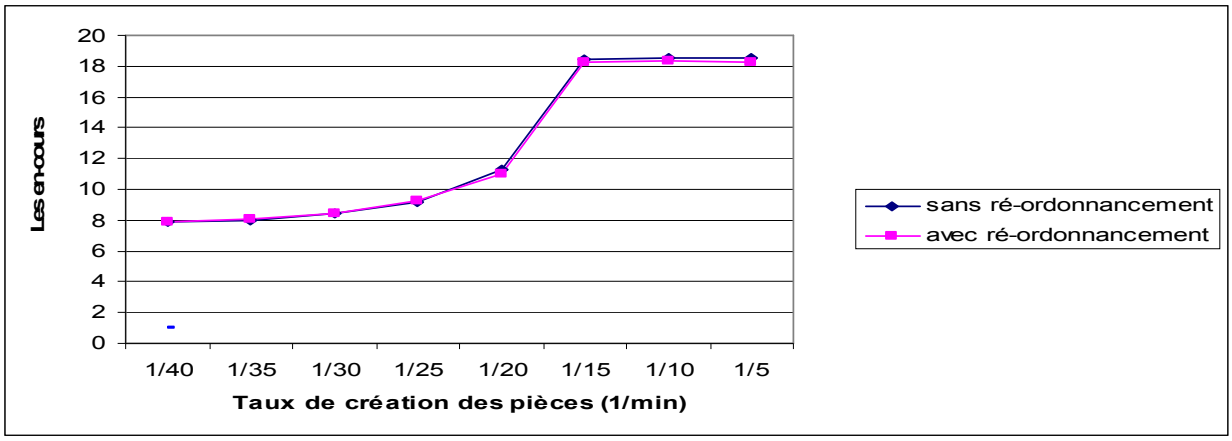

Figure 4.16 : Les en-cours obtenus par les colonies de fourmis pour une capacité de file d'attente = 6.

La figure et le tableau 4.15 nous montrent que le nombre de pièces restantes obtenu par les colonies de fourmis avec ré-ordonnancement est inférieur à celui trouvé sans ré-ordonnancement pour une capacité de files d'attente égale à 6, et un taux de création de pièces supérieur ou égale à 1/20. Hors de cet intervalle, les résultats obtenus dans les deux cas sont presque identiques.

Pour chaque métaheuristique, si on compare le nombre de pièces restantes avec et sans ré-ordonnancement de la station de chargement, pour un taux de création supérieur à 1/20, nous pouvons dire que le ré-ordonnancement augmente les en-cours, sauf pour la recherche tabou avec de petites files d'attente et les colonies de fourmis. Hors de cet intervalle, nous ne pouvons pas dire qu'un cas est meilleur que l'autre (voir les figures 4.13 à 4.16 et les résultats du chapitre précédent).

4.3.1.4. Taux d'utilisation des machines

Dans cette sous section, nous allons montrer l'impact du taux de création des pièces, la capacité des files d'attente et le ré-ordonnancement de la station de chargement sur le taux d'utilisation des machines.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	14.4	17.42	20.41	24.99	30.21	25.98	26.03	25.99
Le recuit simulé	15.24	16.71	19.87	24.64	30.08	23.35	23.46	23.48
Les essaims particulaires	14.9	17.06	20.16	25	32.68	28.7	28.85	28.7
Les algorithmes génétiques	14.43	17.64	20.2	24.45	31.71	29.28	29.09	29.35
Recherche avec tabous	14.30	17.35	20.24	25.2	32.99	24.82	24.8	24.85
Electromagnétisme métaheuristique	14.8	17.81	20.08	24.69	32.52	26.25	26.38	26.39
DMM modifiée	12.7	14.61	16.99	20.32	21.65	17.52	21.13	21.58
DMM	12.69	14.58	17.04	16.65	6.26	10.98	7.88	9.08

Tableau 4.17: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 2.

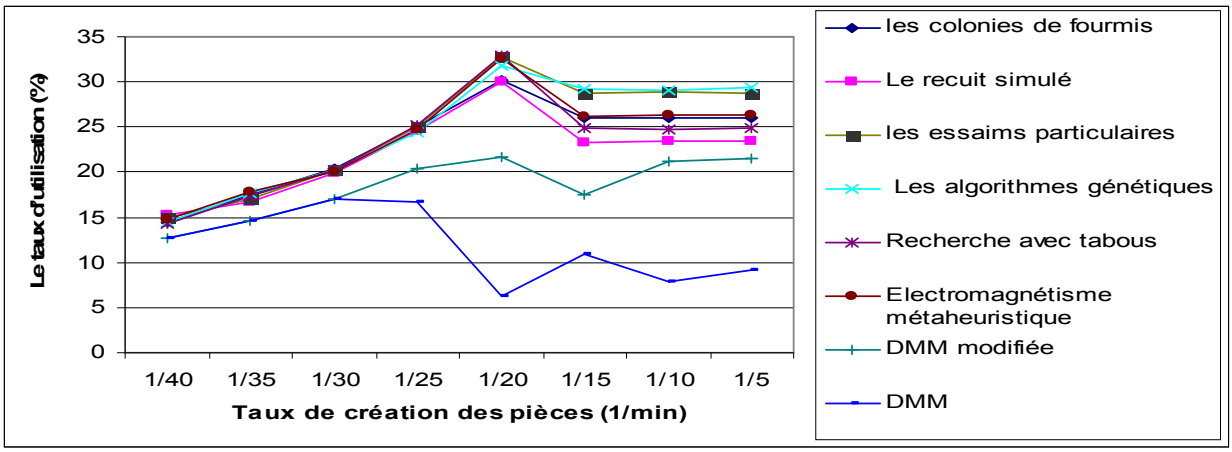

Figure 4.17: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 2.

D'après la courbe de la figure 4.17, on remarque que si on s'intéresse au taux d'utilisation des machines FV_1 et FV_2 pour un taux de création supérieur à 1/20, et une taille de files d'attente égale à 2 les algorithmes génétiques sont plus efficaces que les règles DMM, DMM modifiée et les autres métaheuristiques.

Pour un système non saturé, nous ne pouvons pas dire qu'une métaheuristique est meilleure par rapport aux autres et qu'une règle est meilleure par rapport à d'autre pour tous les taux de création des pièces.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	14.62	17.67	19.7	25.54	31.94	28.6	28.36	28.72
Avec ré-ordonnancement	14.43	17.64	20.2	24.45	31.71	29.28	29.09	29.35

Tableau 4.18 : Le taux d'utilisation des machines FV_1 et FV_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

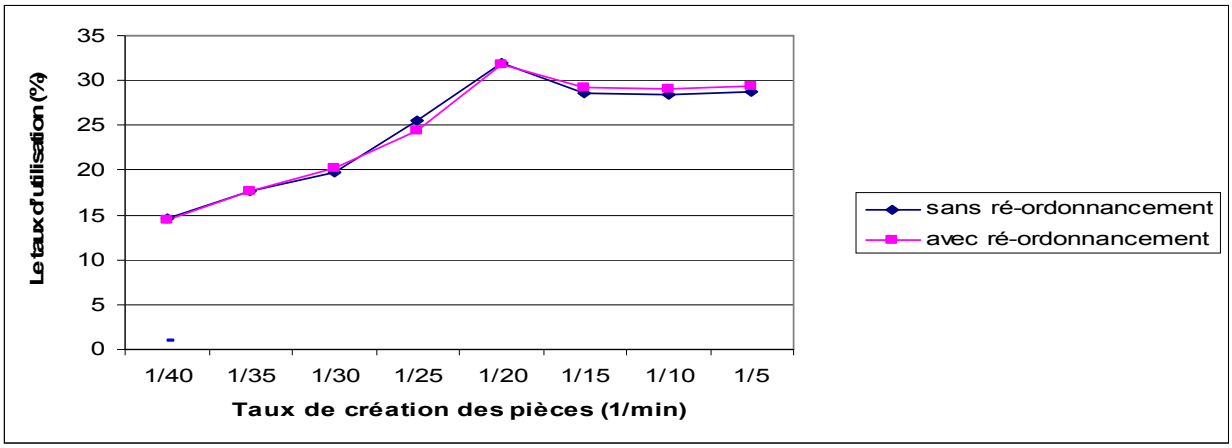

Figure 4.18 : Le taux d'utilisation des machines FV_1 et FV_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.18 nous montrent l'augmentation du taux d'utilisation des machines FV_1 et FV_2 obtenu par les algorithmes génétiques avec ré-ordonnancement par rapport à celui trouvé sans ré-ordonnancement pour un taux d'arrivée de pièces supérieur à 1/20, et une capacité de

files d'attente égale à 2. Hors de cet intervalle, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

A partir de figures et les tableaux 4.17, 4.18 et les résultats présentés dans le chapitre précédent, nous pouvons remarquer que pour de petites files d'attente et un taux de création de pièces supérieur ou égale à 1/15, les résultats obtenus concernant le taux d'utilisation des machines FV_1 et FV_2 trouvés par les métaheuristiques avec ré-ordonnancement de pièces contenues dans la station de chargement sont meilleurs que ceux obtenus sans ré-ordonnancement (sauf les colonies de fourmis).

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	48.54	54.71	63.76	76.11	90.41	86.56	86.87	86.69
Le recuit simulé	47.87	55.28	64.19	76.39	90.01	81.36	81.83	82.05
Les essaims particulaires	48.14	54.99	63.96	76.11	92.85	91.33	91.38	91.34
Les algorithmes génétiques	48.52	54.54	63.92	76.55	93.25	92.33	92.14	92.26
Recherche avec tabous	48.62	54.77	63.89	75.95	93.75	85.98	85.99	86.12
Electromagnétisme métaheuristique	48.21	54.39	64.03	76.35	92.77	87.55	87.72	87.99
DMM modifiée	49.97	56.94	66.59	79.85	84.52	82.72	83.58	84.7
DMM	49.94	57	66.56	65.05	24.62	42.59	30.9	35.5

Tableau 4.19: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 2.

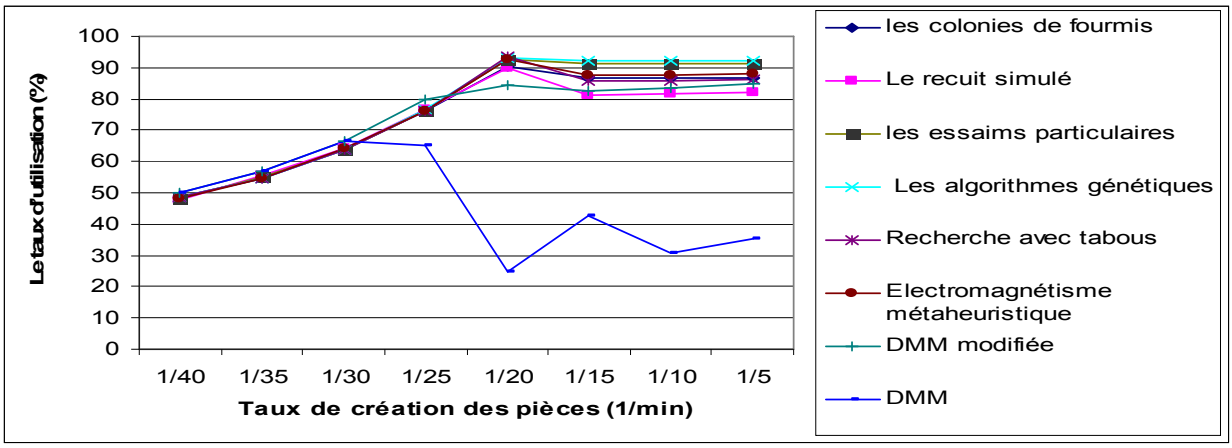

Figure 4.19: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 2.

La courbe de la figure et le tableau 4.19 montrent aussi que le taux d'utilisation des machines T_1 et T_2 est plus important pour les métaheuristiques étudiées que pour DMM et DMM modifiée pour un taux de création important (supérieur à 1/25), et que les algorithmes génétiques sont les plus efficaces pour ces machines pour un taux de création de pièces supérieur à 1/20. Pour un taux de création égale à 1/20, la recherche tabou est la meilleure.

Hors de cet intervalle, DMM et DMM modifiée sont plus efficace que les métaheuristiques.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	48.36	54.51	64.32	75.67	92.72	90.98	90.85	90.96
Avec ré-ordonnancement	48.52	54.54	63.92	76.55	93.25	92.33	92.14	92.26

Tableau 4.20: Le taux d'utilisation des machines T_1 et T_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

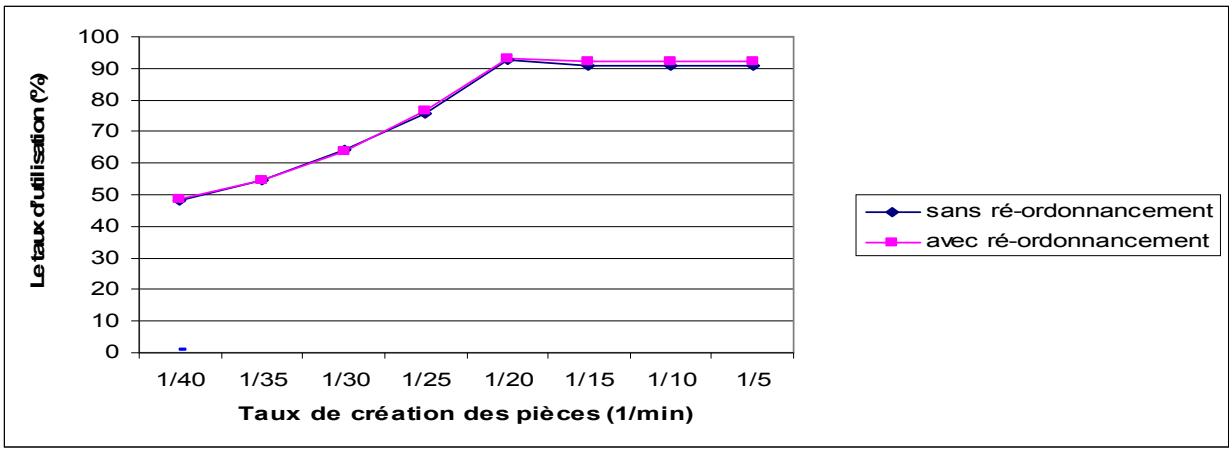

Figure 4.20: Le taux d'utilisation des machines T_1 et T_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.18 nous montrent l'augmentation du taux d'utilisation des machines T_1 et T_2 obtenu par les algorithmes génétiques avec ré-ordonnancement par rapport à celui trouvé sans ré-ordonnancement pour une capacité de files d'attente égale à 2, et un taux de création de pièces supérieur à 1/25. Hors de cet intervalle, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

Les taux d'utilisation des machines T_1 et T_2 obtenus par les métaheuristiques avec ré-ordonnancement de pièces contenues dans la station de chargement sont meilleurs que ceux obtenus sans ré-ordonnancement pour un taux de création de pièces supérieur ou égale à 1/15 (Voir les figures et les tableaux 4.19, 4.20, 3.20).

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	1.317	1.428	1.659	2.645	2.294	2.335	2.345	2.339
Le recuit simulé	1.249	1.485	1.703	1.979	2.283	2.245	2.259	2.267
Les essaims particulaires	1.277	1.457	1.679	1.952	2.277	2.403	2.397	2.403
Les algorithmes génétiques	1.314	1.411	1.676	1.995	2.34	2.416	2.418	2.41
Recherche avec tabous	1.324	1.434	1.673	1.935	2.299	2.365	2.366	2.37
Electromagnétisme métaheuristique	1.284	1.397	1.686	1.97	2.286	2.363	2.363	2.374
DMM modifiée	1.46	1.69	2.09	3.82	5.35	8.6	6.06	5.57
DMM	1.45	1.68	2.09	2.79	1.25	2.03	1.6	1.91

Tableau 4.21: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 2

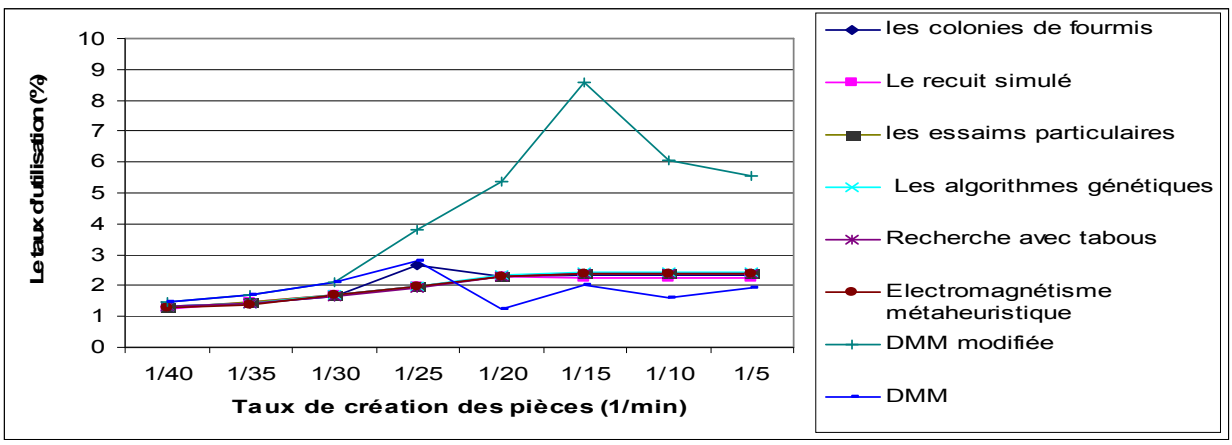

Figure 4.21: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 2.

Le taux d'utilisation de la toupie (voir la figure 4.21) est plus faible pour la méthode DMM et les métaheuristiques que celui obtenu par la méthode DMM modifiée pour des taux de création supérieurs à 1/30, pour les autres taux de création les taux d'utilisation pour DMM et DMM modifiée sont pratiquement égaux.

Si on s'intéresse aux métaheuristiques, nous pouvons remarquer pour un taux de création de pièces supérieur à 1/25 que les algorithmes génétiques sont les plus efficaces pour une taille de file d'attente égale à 2.

Pour un système non saturé, nous ne pouvons pas dire qu'une métaheuristique est meilleure par rapport aux autres et qu'une règle est plus efficace par rapport à d'autre pour tous les taux de création des pièces.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	1.299	1.408	1.711	1.907	2.307	2.393	2.399	2.387
Avec ré-ordonnancement	1.314	1.411	1.676	1.995	2.34	2.416	2.418	2.41

Tableau 4.22: Le taux d'utilisation de la machine TP obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

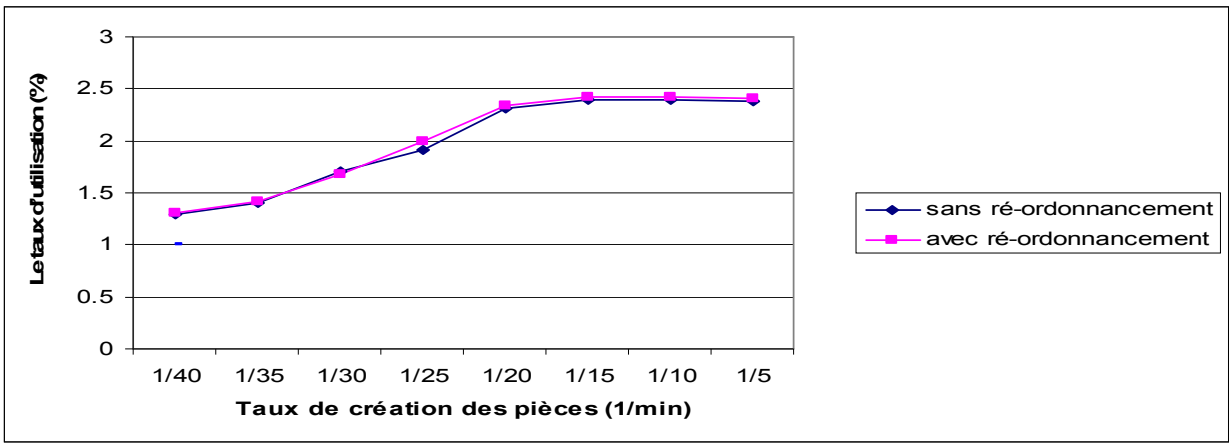

Figure 4.22: Le taux d'utilisation de la machine TP obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

Nous pouvons remarquer (voir les figures 4.21, 4.22 et 3.22) pour un taux de création de pièces supérieur ou égale à 1/15, que le ré-ordonnancement des pièces contenues dans la station de chargement améliore le taux d'utilisation de la machine TP pour toutes les métaheuristiques.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	23.36	25.35	29.46	34.65	40.73	41.41	41.6	41.5
Le recuit simulé	22.18	26.34	31.72	35.14	40.55	39.8	40.05	40.19
Les essaims particulaires	22.66	25.85	29.81	34.65	40.47	42.64	42.56	42.64
Les algorithmes génétiques	23.32	25.05	29.75	35.42	41.57	42.89	42.91	42.78
Recherche avec tabous	23.49	25.45	29.69	34.37	40.87	41.94	41.97	42.03
Electromagnétisme métaheuristique	22.79	24.8	29.92	35.07	40.62	41.91	41.93	42.12
DMM modifiée	18.05	20.61	23.96	28.63	30.34	29.15	30.16	30.4
DMM	18.07	20.61	23.94	23.46	8.87	15.33	11.11	12.78

Tableau 4.23: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente = 2.

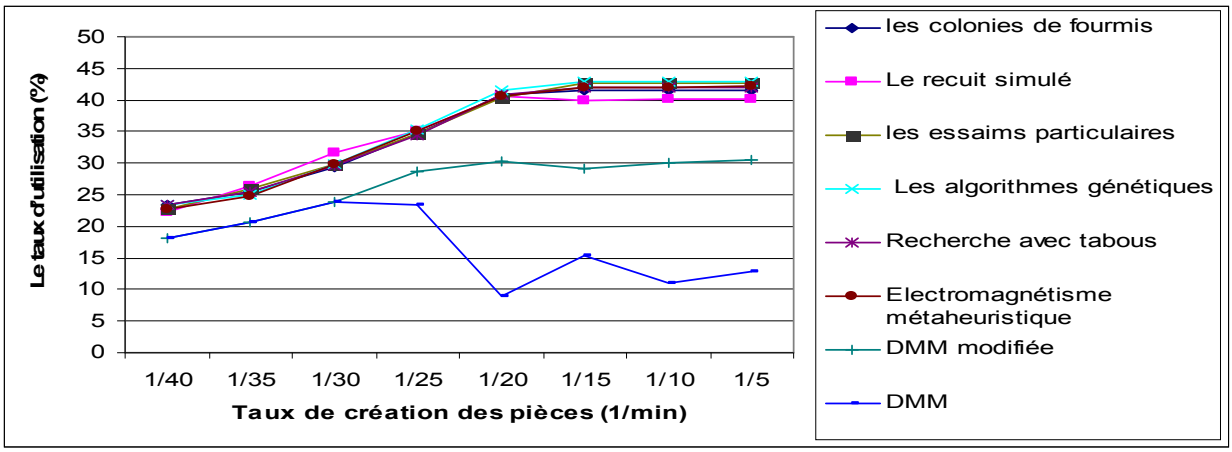

Figure 4.23: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente = 2.

Les machines FH_1 et FH_2 sont moins utilisées dans la méthode DMM et DMM modifiée que dans les métaheuristiques (voir la figure 4.23).

Si on s'intéresse aux métaheuristiques, nous pouvons remarquer pour un taux de création de pièces supérieur à 1/25 que les algorithmes génétiques sont les plus efficaces pour une taille de file d'attente égale à 2.

Pour un système non saturé, nous ne pouvons pas dire qu'une métaheuristique est meilleure par rapport aux autres et qu'une règle est plus efficace par rapport à d'autre pour tous les taux de création des pièces.

Hors de cet intervalle, DMM et DMM modifiée sont plus efficaces que les métaheuristiques.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	23.05	25	30.45	33.88	40.99	42.47	42.57	42.36
Avec ré-ordonnancement	23.32	25.05	29.75	35.42	41.57	42.89	42.91	42.78

Tableau 4.24: Le taux d'utilisation des machines FH_1 et FH_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

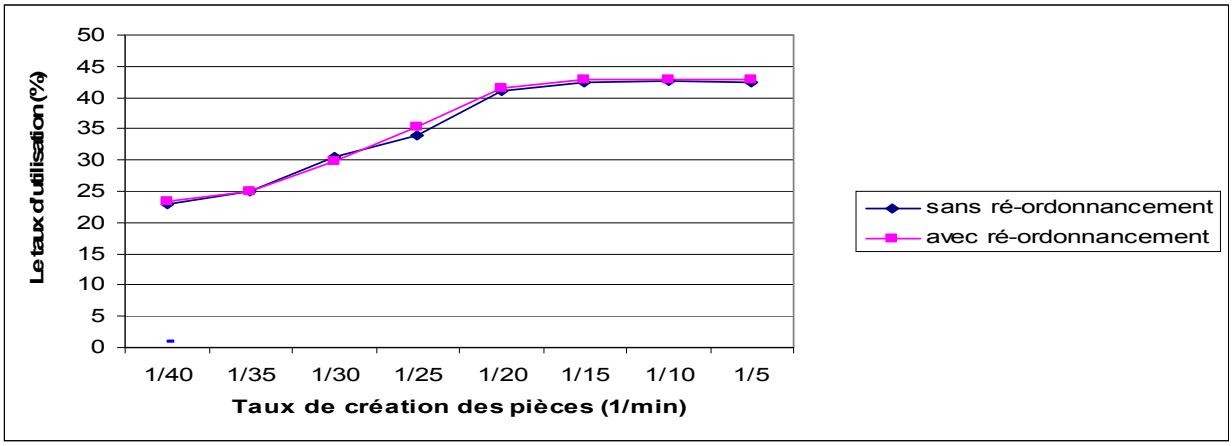

Figure 4.24: Le taux d'utilisation des machines FH_1 et FH_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.24 nous montrent que le taux d'utilisation des machines FH_1 et FH_2 obtenu par les algorithmes génétiques avec ré-ordonnancement est supérieur à celui trouvé sans ré-ordonnancement pour une capacité de files d'attente égale à 2, et un taux de création de pièces supérieur ou égale à 1/25. Hors de cet intervalle, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

Les figures et les tableaux 4.23, 4.24 et les résultats présentés dans le chapitre précédent montrent que pour un taux de création de pièces supérieur ou égale à 1/15, les résultats concernant le taux d'utilisation des machines FH_1 et FH_2 obtenus par les métaheuristiques avec ré-ordonnancement de pièces contenues dans la station de chargement sont meilleurs que ceux obtenus sans ré-ordonnancement.

4.3.1.5. Taux d'utilisation de l'AGV

Dans cette sous section, nous allons montrer l'impact du taux de création des pièces, la capacité des files d'attente et le ré-ordonnancement de la station de chargement sur le taux d'utilisation de l'AGV.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	17.52	19.64	22.88	27.26	32.33	31.22	31.33	31.26
Le recuit simulé	17.18	19.92	23.1	27.39	32.19	29.44	29.61	29.69
Les essaims particulaires	17.32	19.78	22.98	27.26	33.05	32.82	32.83	32.82
Les algorithmes génétiques	17.51	19.55	22.96	27.47	33.62	33.16	33.1	33.12
Recherche avec tabous	17.56	19.67	22.94	27.18	33.37	31.1	31.1	31.15
Electromagnétisme métaheuristique	17.36	19.48	23.01	27.38	33.07	31.58	31.63	31.74
DMM modifiée	14.98	17.46	21	27.31	30.26	29.16	30.19	30.44
DMM	15	17.4	20.93	21.67	8.43	14.35	10.55	12.08

Tableau 4.25: Le taux d'utilisation de l'AGV pour une capacité de file d'attente = 2.

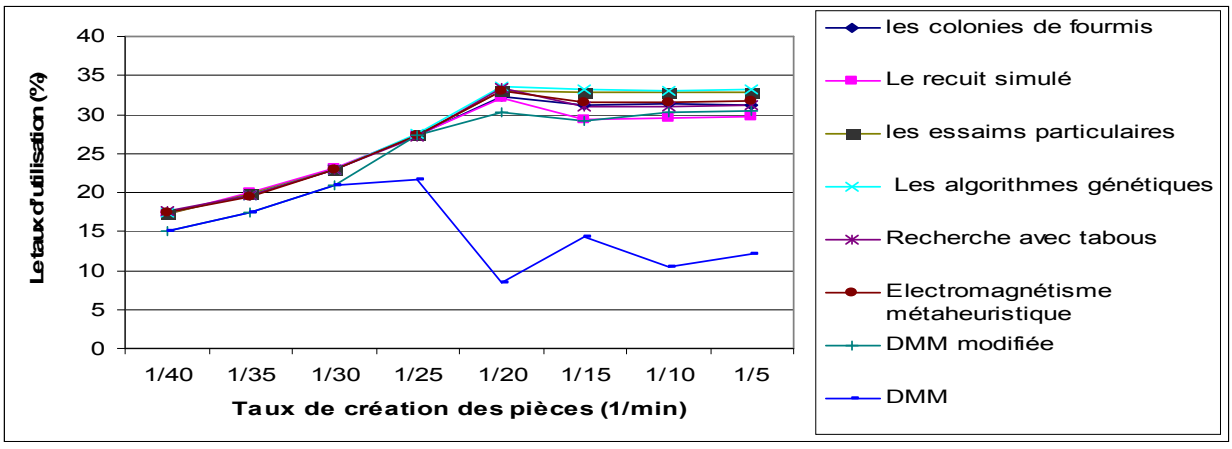

Figure 4.25: Le taux d'utilisation de l'AGV pour une capacité de file d'attente = 2.

La figure et tableau 4.25 montrent que pour une taille de file d'attente égale à 2, et un taux de création de pièces supérieur ou égale à 1/25, les taux d'utilisation d'AGV pour les algorithmes génétiques sont meilleurs que ceux obtenus par les autres métaheuristiques et les règles DMM et DMM modifiée.

Pour un taux de création égale à 1/30, les essaims particulaires sont les meilleurs avec un peu de différence par rapport aux algorithmes génétiques. Pour un taux de création égale à 1/35, le recuit simulé est le plus efficace, la recherche tabou est la meilleure pour un taux de création égale à 1/40.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	17.43	19.54	23.16	27.04	33.07	32.69	32.67	32.68
Avec ré-ordonnancement	17.51	19.55	22.96	27.47	33.62	33.16	33.1	33.12

Tableau 4.26: Le taux d'utilisation de l'AGV obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

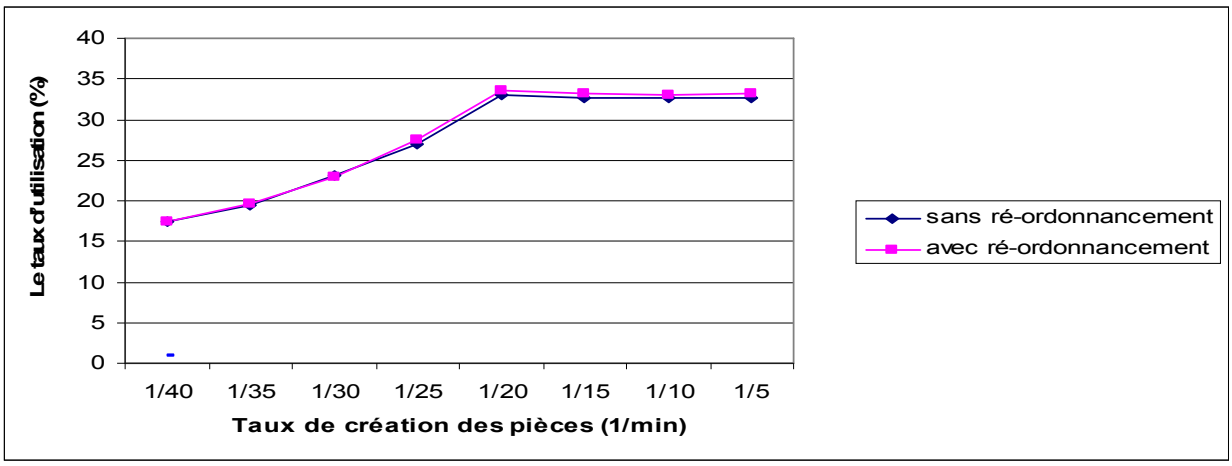

Figure 4.26: Le taux d'utilisation de l'AGV obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.26 nous montrent que le taux d'utilisation de l'AGV obtenu par les algorithmes génétiques avec ré-ordonnancement est supérieur à celui trouvé sans ré-ordonnancement pour une capacité de files d'attente égale à 2, et un taux de création de pièces supérieur ou égale à 1/20. Hors de cet intervalle, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

Les figures et les tableaux 4.25, 4.26 et les résultats du chapitre précédent montrent que pour un système saturé, l'utilisation de l'AGV est plus importante pour chaque métaheuristique dans le cas avec ré-ordonnancement que pour le cas sans ré-ordonnancement.

4.3.2. Etude comparative avec introduction de pannes

L'objectif de cette sous section est simulaire à celui de la précédente mais cette fois les pannes des machines sont prises en compte.

4.3.2.1. Taux de production

La simulation du système en introduisant des pannes, nous montre que les taux de production obtenus par les métaheuristiques étudiées avec ré-ordonnancement de pièces contenues dans la station de chargement sont nettement supérieurs à ceux obtenus sans ré-ordonnancement pour une capacité de file d'attente égale à 2 et un système saturé (voir les figures 4.27, 4.28 et 3.28).

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	99.99	99.99	99.99	99.99	87.08	65.24	43.66	21.83
Le recuit simulé	99.99	99.99	99.99	99.99	81.49	60.99	40.72	20.37
Les essaims particulaires	99.99	99.99	99.99	99.99	92.67	69.57	46.26	23.22
Les algorithmes génétiques	99.99	99.99	99.99	99.99	93.3	70.08	46.72	23.38
Recherche avec tabous	99.99	99.99	99.99	99.99	85.8	64.27	42.93	21.51
Electromagnétisme métaheuristique	99.99	99.99	99.99	99.99	90.14	66.29	44.35	22.17
DMM modifiée	99.99	99.97	99.62	85.02	66.7	50.91	34.29	17.04
DMM	99.99	99.97	99.95	53.07	32.46	29.29	15.43	8.55

Tableau 4.27: Taux de sortie des pièces pour une capacité de file d'attente = 2.

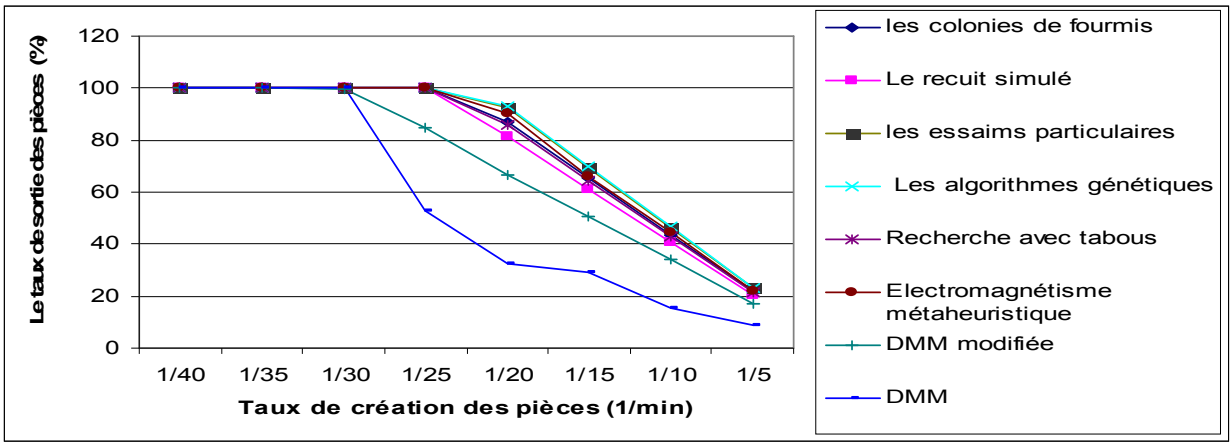

Figure 4.27: Taux de sorite des pièces pour une capacité de file d'attente = 2.

La figure 4.27 nous montre aussi que le taux de production pour une taille de file d'attente égale à 2, et un taux de création supérieur à 1/25, est plus important pour les algorithmes génétiques que pour les autres métaheuristiques, et hors de cet intervalle les taux de production obtenus par les métaheuristiques sont identiques.

De ces résultats avec et sans ré-ordonnancement de la station de chargement (avec et sans présence de pannes), nous pouvons conclure que le système sature moins rapidement si on utilise les métaheuristiques par rapport aux règles DMM et DMM modifiée.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	99.99	99.99	99.99	99.99	92	69.05	46.22	23.12
Avec ré-ordonnancement	99.99	99.99	99.99	99.99	93.3	70.08	46.72	23.38

Tableau 4.28: Taux de sortie de pièces obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

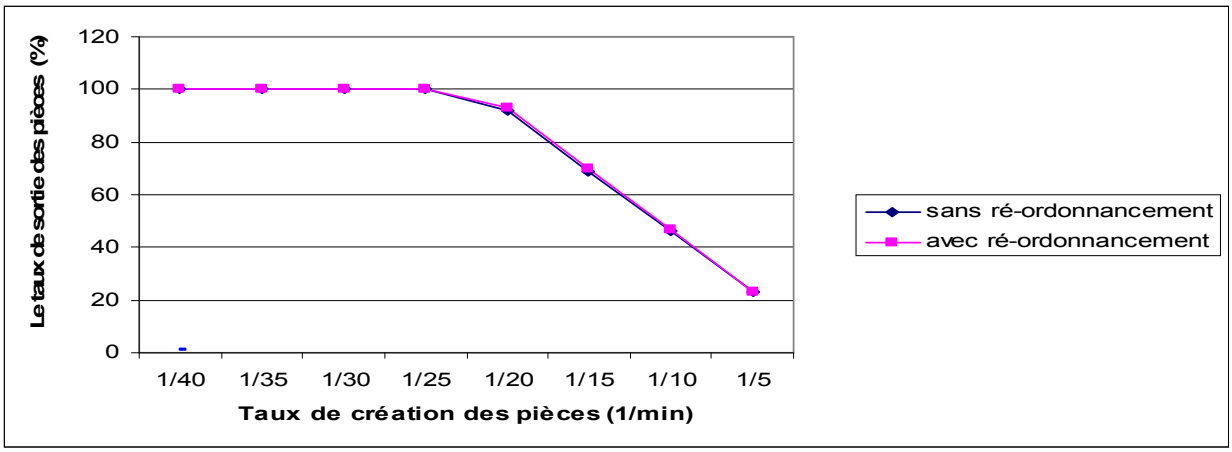

Figure 4.28: Taux de sortie de pièces obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.28 nous montrent que pour une capacité de files d'attente égale à 2 et un taux de création de pièces supérieur ou égale à 1/20, le taux de production obtenu par les algorithmes génétiques avec ré-ordonnancement est meilleur que celui trouvé sans ré-ordonnancement. Hors de cet intervalle, les résultats obtenus dans les deux cas sont identiques.

4.3.2.2. Le temps de cycle

D'après les résultats illustrés dans la figure 4.29, nous remarquons que les temps de cycle de la règle DMM et DMM modifiée sont supérieurs à ceux obtenus par les métaheuristiques étudiées, sauf pour un taux de création égale à 1/40.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	94.8	86.2	92	103	173.4	174.6	173.7	174.3
Le recuit simulé	94	85.1	94.6	102.8	173	174.2	176.7	180.4
Les essaims particulaires	93.7	84.1	89.5	103.8	172.4	178.9	174.4	172.3
Les algorithmes génétiques	90.8	86.5	91.8	101.5	178.8	177.4	178.3	175.4
Recherche avec tabous	93.7	85.9	93	100.1	176.3	170	170.7	170.9
Electromagnétisme métaheuristique	95.5	85.4	91.1	103.8	171.3	172.1	173.7	172.8
DMM modifiée	91.1	99.1	119.9	325.1	353.3	346.8	339.8	341.1
DMM	91.3	101.6	120.9	185.3	203.7	193.8	199.6	203

Tableau 4.29: Temps de cycle pour une capacité de file d'attente = 2.

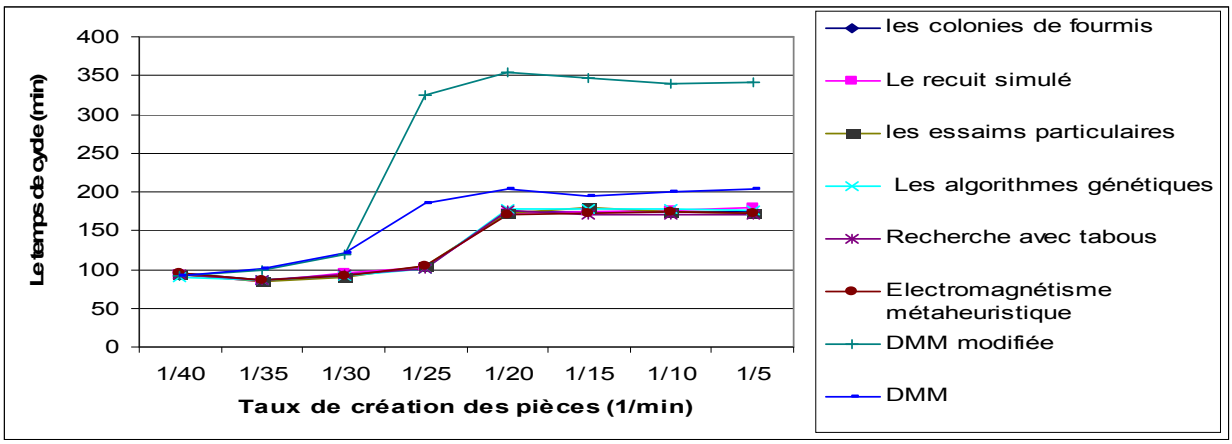

Figure 4.29: Temps de cycle pour une capacité de file d'attente = 2.

La figure et tableau 4.29 montrent que pour un taux de création de pièces supérieur ou égale à 1/15 et une taille de file d'attente égale à 2, la recherche tabou est la plus efficace si on s'intéresse au temps de cycle. Pour un taux de création égale à 1/20, l'électromagnétisme est la meilleure.

Hors de cet intervalle, nous ne pouvons pas dire qu'une métaheuristique est la plus efficace par rapport aux autres pour tous les taux de création des pièces.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	94.2	85.7	93.3	104.6	170.8	174.6	175.2	176.2
Avec ré-ordonnancement	90.8	86.5	91.8	101.5	178.8	177.4	178.3	175.4

Tableau 4.28: Temps de cycle obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

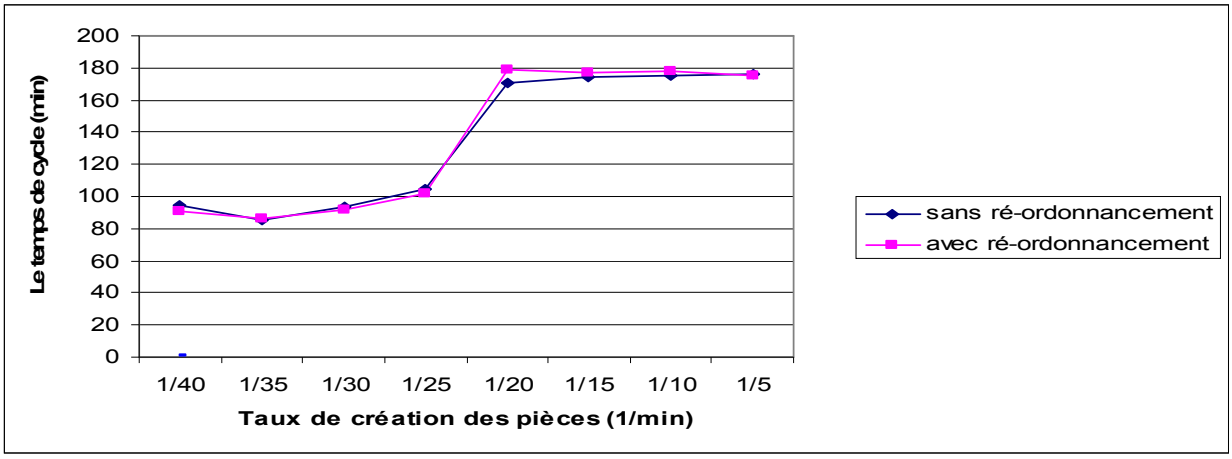

Figure 4.28: Temps de cycle obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.28 montrent l'augmentation du temps de cycle obtenu par les algorithmes génétiques avec ré-ordonnancement par rapport à celui obtenu sans ré-ordonnancement pour la plupart des cas d'un système saturé. Pour un système non saturé, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

Pour chaque métaheuristique, si on compare le temps de cycle si on ré-ordonnance la station de chargement avec celui trouvé sans ré-ordonnancement, nous pouvons dire que le ré-ordonnancement augmente le temps de cycle pour la plupart des cas d'un système saturé.

4.3.2.3. Les en-cours

La figure suivante montre que le nombre de pièces qui restent dans le système est plus grand pour les métaheuristiques étudiées et la règle DMM modifiée vis-à-vis de la règle DMM, pour un taux de création de pièces supérieur à 1/30 et de petite taille de files d'attente.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	3.94	4.1	4.54	5.35	8.11	8.1	8.09	8.13
Le recuit simulé	3.92	4.08	4.53	5.47	8.07	8.05	8.07	8.09
Les essaims particulaires	3.94	4.1	4.56	5.4	8.9	8.92	8.92	8.92
Les algorithmes génétiques	3.92	4.09	4.56	5.4	9.96	9.98	8.98	8.99
Recherche avec tabous	3 .94	4.1	4.56	5.45	8.33	8.3	8.33	8.34
Electromagnétisme métaheuristique	3.94	4.09	4.54	5.47	8.66	8.66	8.68	8.66
DMM modifiée	4.55	7.79	28.62	29.82	30.51	28.04	28.64	27.2
DMM	4.6	5.9	6.11	4.85	5.99	5.29	4.67	0.46

Tableau 4.31: Les en-cours pour une capacité de file d'attente = 2.

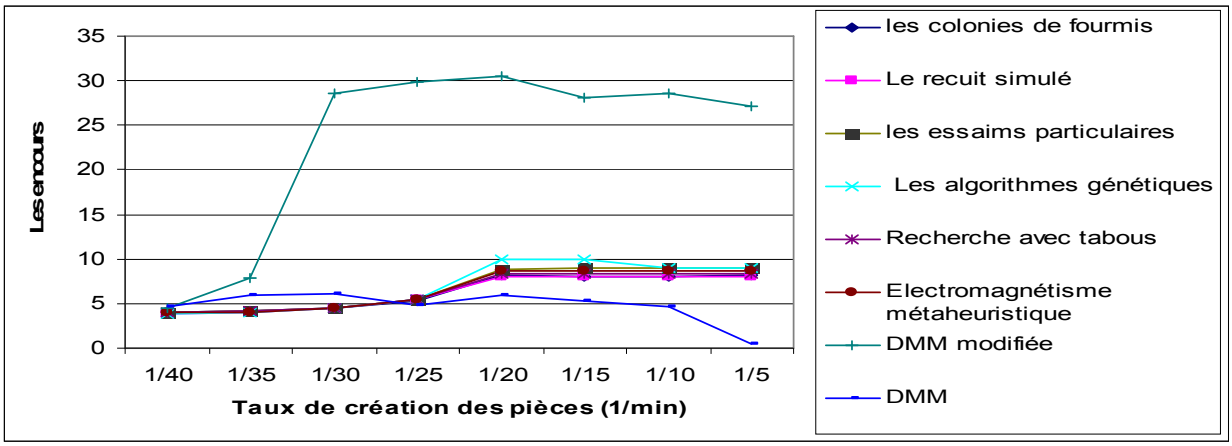

Figure 4.31 : Les en-cours pour une capacité de file d'attente = 2.

La figure 4.31 montre aussi que le recuit simulé donne des résultats meilleurs concernant les en-cours par rapport aux autres métaheuristiques et DMM modifiée pour une taille de files d'attente égale à 2 et un taux de création supérieur à 1/25. Et pour un taux de création de pièces supérieur à 1/35, l'augmentation des en-cours pour la règle DMM modifiée apparaît clairement par rapport aux métaheuristiques et la règle DMM.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	3.95	4.09	4.55	5.4	8.46	8.43	8.43	8.46
Avec ré-ordonnancement	3.94	4.1	4.54	5.35	8.11	8.1	8.09	8.13

Tableau 4.32: Les en-cours obtenus par les colonies de fourmis pour une capacité de file d'attente = 2.

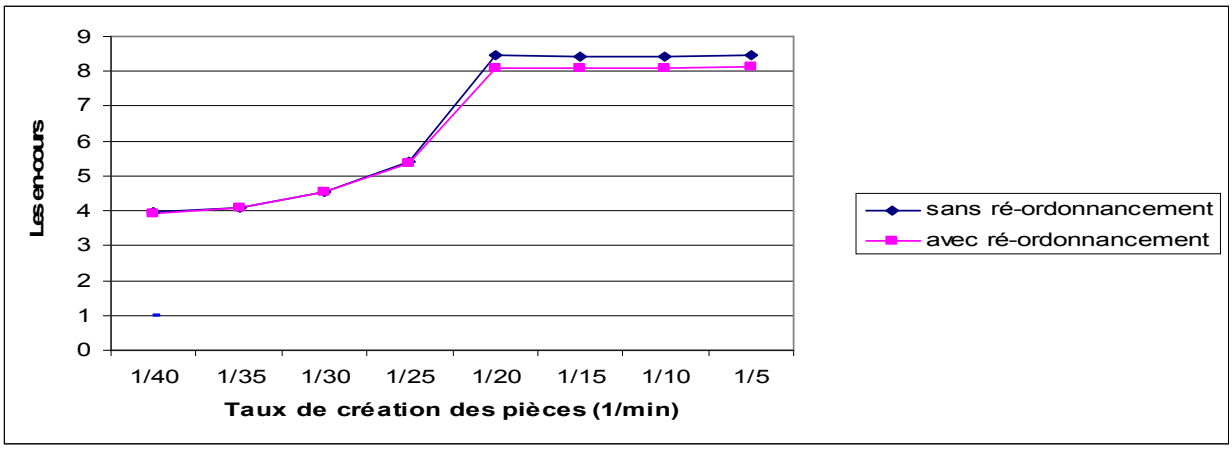

Figure 4.32: Les en-cours obtenus par les colonies de fourmis pour une capacité de file d'attente = 2.

Pour les colonies de fourmis, si on compare le nombre de pièces restantes si on ré-ordonnance la station de chargement avec celui trouvé sans ré-ordonnancement, nous pouvons dire que le ré-ordonnancement a amélioré les performances concernant les en-cours, pour une taille de file d'attente égale à 2 et un taux de création de pièces supérieur ou égale à 1/20. Hors de cet intervalle, les résultats obtenus dans les deux cas sont presque identiques. (Voir la figure 4.32).

Pour chaque métaheuristique (sauf pour la recherche tabou et les colonies de fourmis), le ré-ordonnancement augmente les en-cours même avec la présence de pannes, pour un taux de création de pièces supérieur à 1/25 et de petites files d'attente (voir les figures et les tableaux 4.31, 4.32 et 3.32).

4.3.2.4. Taux d'utilisation des machines

Comme nous pouvons bien le remarquer sur les figures (4.33 à 4.37), avec un taux de création de pièces supérieur à 1/25, le taux d'utilisation des machines est toujours plus important en utilisant les métaheuristiques, sauf l'utilisation de la machine TP. Ces figures montrent aussi que certaines métaheuristiques sont meilleures que d'autres mais il n'existe pas une métaheuristique plus efficace pour toutes les machines et tous les taux.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	14.7	17.38	20.12	25.12	25.34	25.3	25.36	25.35
Le recuit simulé	14.73	17.62	20.24	25.18	22.84	22.77	22.66	22.77
Les essaims particulaires	14.79	17.42	20.14	25.01	27.95	28.07	27.78	28.12
Les algorithmes génétiques	14.89	17.3	20.16	24.93	28.1	28.2	28.15	28.23
Recherche avec tabous	14.71	17.62	20.28	25	24.17	24.14	24.23	24.21
Electromagnétisme métaheuristique	14.75	17.34	20.03	24.86	27.06	25.69	25.79	25.79
DMM modifiée	12.79	14.67	16.89	17.35	17.06	17.25	17.5	17.33
DMM	12.66	14.56	16.99	10.86	8.23	9.91	7.96	8.72

Tableau 4.33: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 2.

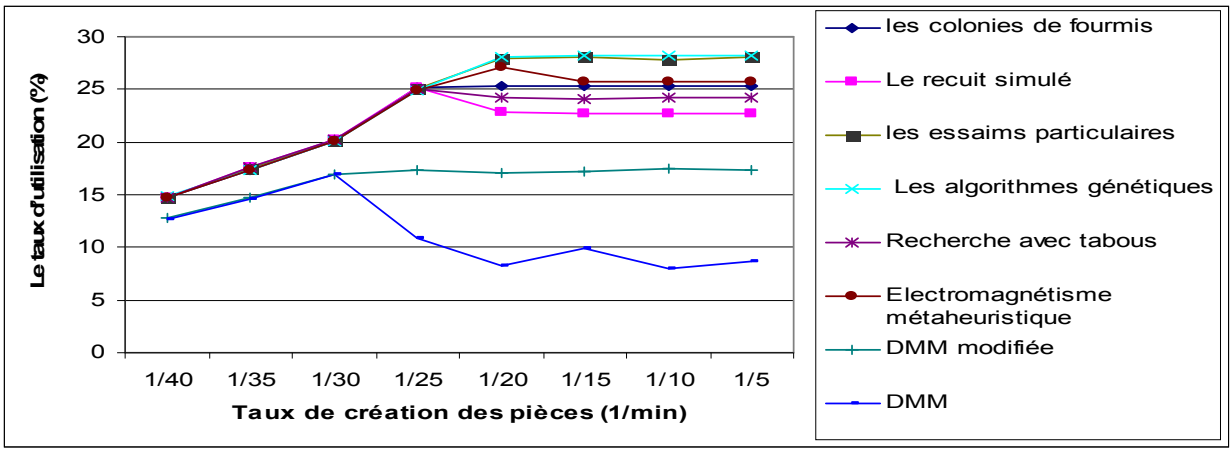

Figure 4.33: Le taux d'utilisation des machines FV_1 et FV_2 pour une capacité de file d'attente = 2.

La figure et le tableau 4.33 montrent aussi que pour une capacité de file d'attente égale à 2, l'utilisation des machines FV_1 et FV_2 est plus importante pour DMM modifiée par rapport à DMM et les algorithmes génétiques sont les plus efficaces pour un taux de création de pièces supérieur à 1/25. Pour un système non saturé, la règle DMM modifiée est toujours plus efficace que DMM sauf pour un taux de création égale à 1/30 et nous ne pouvons pas dire qu'une métaheuristique est meilleure que les autres pour tous les taux (le recuit simulé est le meilleur pour un taux de création égale à 1/25, la recherche tabou est la plus efficace pour un taux compris entre 1/35 et 1/30, l'électromagnétisme est la meilleure pour un taux de création égale à 1/40).

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	14.7	17.18	20.07	25.02	27.1	27.72	28.05	28.04
Avec ré-ordonnancement	14.89	17.3	20.16	24.93	28.1	28.2	28.15	28.23

Tableau 4.34: Le taux d'utilisation des machines FV_1 et FV_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

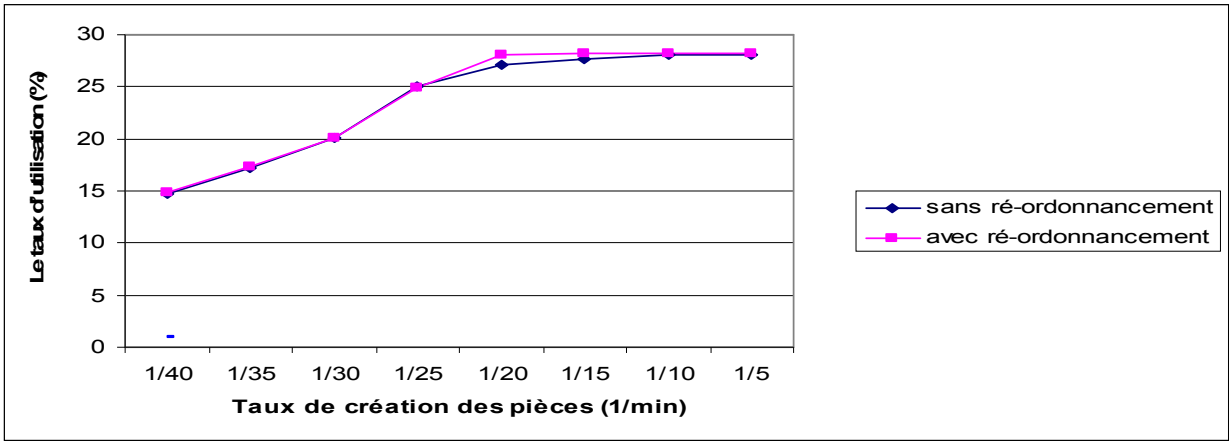

Figure 4.34 : Le taux d'utilisation des machines FV_1 et FV_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.34 nous montrent l'augmentation du taux d'utilisation des machines FV_1 et FV_2 obtenu par les algorithmes génétiques avec ré-ordonnancement par rapport à celui

trouvé sans ré-ordonnancement pour un taux de création de pièces supérieur à 1/25, et une capacité de files d'attente égale à 2. Hors de cet intervalle, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

A partir de figures et les tableaux 4.33, 4.34 et les résultats présentés dans le chapitre précédent, nous pouvons remarquer que pour de petites tailles de files d'attente et un taux de création de pièces supérieur ou égale à 1/20, les résultats obtenus concernant le taux d'utilisation de machines FV_1 et FV_2 par certaines métaheuristiques (les essaims particulaires, les algorithmes génétiques, la recherche tabou et l'électromagnétisme) avec ré-ordonnancement de pièces contenues dans la station de chargement sont meilleurs que ceux obtenus sans ré-ordonnancement.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	48.3	54.74	63.99	76.01	84.34	84.27	84.63	84.64
Le recuit simulé	48.28	54.55	63.9	75.97	79.63	79.49	79.16	79.69
Les essaims particulaires	48.23	54.71	63.97	76.1	88.96	88.99	88.93	89.12
Les algorithmes génétiques	48.15	54.8	63.96	76.16	89.61	89.71	89.74	89.79
Recherche avec tabous	48.29	54.54	63.86	76.11	83.74	83.65	83.79	84.02
Electromagnétisme métaheuristique	48.26	54.77	64.06	76.21	86.64	85.64	85.94	85.91
DMM modifiée	49.88	56.94	66.78	77.16	77.77	78.13	78.5	78.75
DMM	49.96	57.23	66.81	42.71	33.02	39.3	31.21	34.83

Tableau 4.35: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 2.

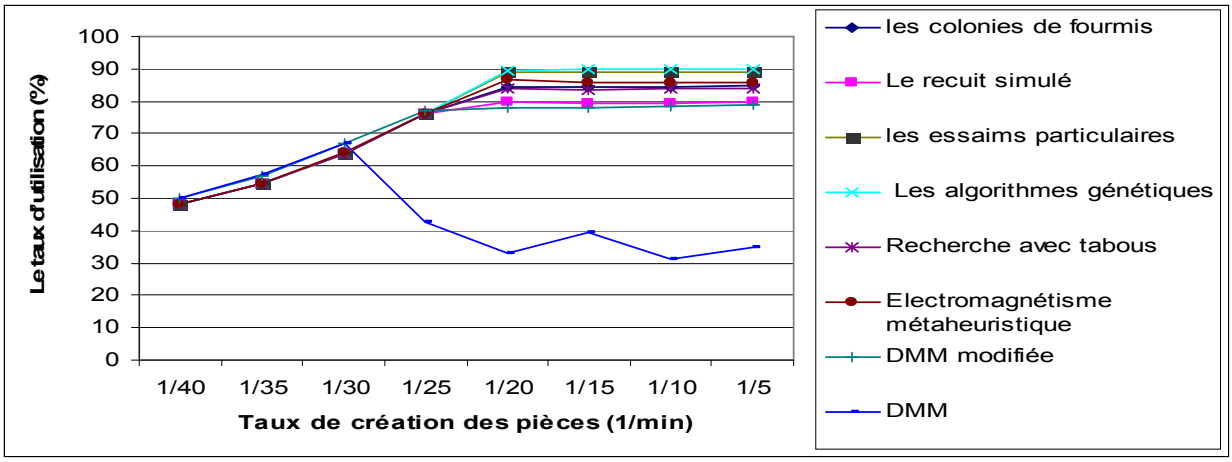

Figure 4.35: Le taux d'utilisation des machines T_1 et T_2 pour une capacité de file d'attente = 2.

A partir de la figure et tableau 4.35, nous povons remarquer que pour une capacité de file d'attente égale à 2, l'utilisation des machines T_1 et T_2 est plus importante pour les algorithmes génétiques par rapport à DMM, DMM modifiée et les autres métaheuristiques pour un taux de création de pièces supérieur à 1/25.

Hors de cet intervalle, les règles DMM et DMM modifiée donnent les meilleurs taux d'utilisation, et si on veut sélectionner la meilleure métaheuristique l'électromagnétisme est la plus efficace pour un taux compris entre 1/30 et 1/25, les algorithmes génétiques sont les meilleurs pour un taux de création des pièces égale à 1/35 et les colonies de fourmis dépassent les autres pour un taux égale à 1/40.

Les taux d'utilisation des machines T_1 et T_2 obtenus par les métaheuristiques avec ré-ordonnancement de pièces contenues dans la station de chargement sont meilleurs que ceux obtenus sans ré-ordonnancement pour de petites files d'attente et un taux de création de pièces supérieur ou égale à 1/20, sauf ceux obtenus par le recuit simulé pour un taux égale à 1/5 (Voir les figures 4.35 et 3.36).

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	1.293	1.431	1.683	1.941	2.274	2.272	2.284	2.285
Le recuit simulé	1.29	1.412	1.673	1.937	2.198	2.195	2.186	2.203
Les essaims particulaires	1.286	1.428	1.681	1.95	2.34	2.336	2.347	2.339
Les algorithmes génétiques	1.277	1.438	1.68	1.956	2.36	2.359	2.363	2.362
Recherche avec tabous	1.292	1.412	1.67	1.951	2.304	2.302	2.307	2.314
Electromagnétisme métaheuristique	1.289	1.434	1.69	1.962	2.287	2.31	2.318	2.317
DMM modifiée	1.47	1.73	2.72	22.21	25.27	24.31	23.55	23.73
DMM	1.48	1.77	2.56	2.98	2.88	2.79	2.28	3.01

Tableau 4.36: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 2.

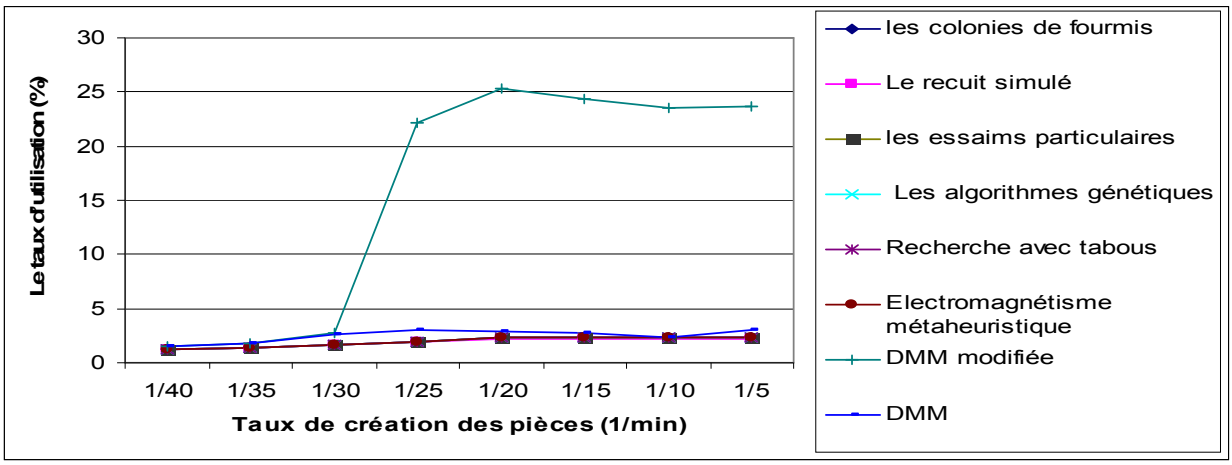

Figure 4.36: Le taux d'utilisation de la machine TP pour une capacité de file d'attente = 2.

Comme nous l'avons bien remarqué (voir la figure et le tableau 4.36) l'utilisation de la machine TP est meilleure pour la règle DMM modifiée par rapport à la règle DMM et les métaheuristiques pour un taux de création supérieur à 1/35, et dans l'intervalle [1/40, 1/35] la règle DMM est la meilleure.

Si on compare les métaheuristiques, les algorithmes génétiques donnent une utilisation de la machine TP supérieure à celle des autres métaheuristiques pour un système saturé, et qu'en dessous de taux de création 1/20 l'utilisation est meilleure pour l'électromagnétisme dans l'intervalle [1/30,1/25], les algorithmes génétiques pour un taux de création égale à 1/30 et les colonies de fourmis pour un taux égale à 1/40.

Nous pouvons remarquer (voir les figures et les tableaux 4.36 et 3.37) pour un taux de création de pièces supérieur ou égale à 1/20 et de petites files d'attente, que le ré-ordonnancement de pièces contenues dans la station de chargement améliore le taux d'utilisation de la machine TP pour toutes les métaheuristiques.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	22.95	25.4	29.87	34.47	40.33	40.31	40.52	40.54
Le recuit simulé	22.9	25.08	29.7	34.4	38.97	38.91	38.77	39.06
Les essaims particulaires	22.81	25.35	29.83	34.63	41.54	41.46	41.66	41.51
Les algorithmes génétiques	22.67	25.52	29.81	34.73	41.88	41.88	41.94	41.91
Recherche avec tabous	22.92	25.07	29.64	34.65	40.86	40.81	40.85	41.03
Electromagnétisme métaheuristique	22.86	22.46	29.99	34.83	40.58	40.98	41.17	41.1
DMM modifiée	17.99	20.49	23.91	24.52	24.02	24.43	24.64	24.49
DMM	18.08	20.58	23.95	15.31	11.67	14.1	11.11	12.29

Tableau 4.37: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente =2.

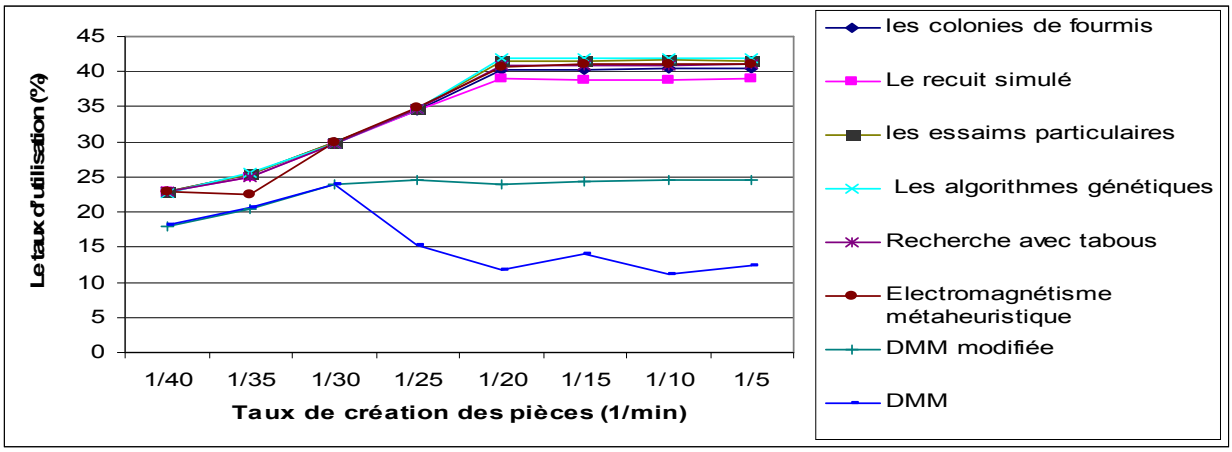

Figure 4.37: Le taux d'utilisation des machines FH_1 et FH_2 pour une capacité de file d'attente = 2.

La figure et le tableau 4.37 montrent que pour une capacité de files d'attente égale à 2, si le taux de création des pièces est supérieur à 1/25, le taux d'utilisation des machines FH_1 et FH_2 est plus important pour les algorithmes génétiques que pour les métaheuristiques et les règles DMM et DMM modifiée, et qu'en dessous de taux de création 1/20 l'utilisation de ces machines est meilleure pour l'électromagnétisme dans l'intervalle [1/30,1/25] et les algorithmes génétiques pour un taux de création égale à 1/30 et les colonies de fourmis pour un taux égale à 1/40.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	22.93	25.69	29.94	34.62	41.63	41.36	41.22	41.25
Avec ré-ordonnancement	22.67	25.52	29.81	34.73	41.88	41.88	41.94	41.91

Tableau 4.38: Le taux d'utilisation des machines FH_1 et FH_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

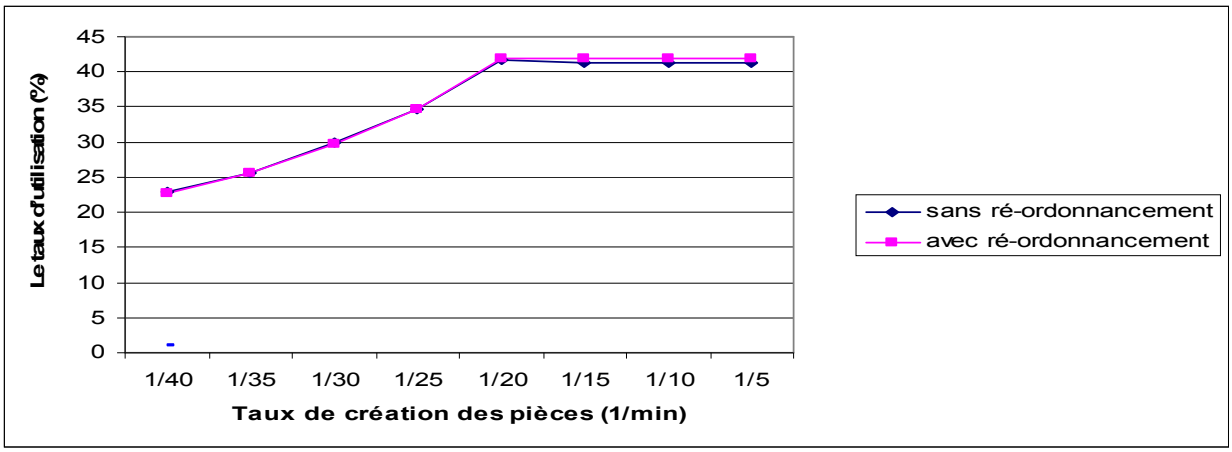

Figure 4.38: Le taux d'utilisation des machines FH_1 et FH_2 obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

La figure et le tableau 4.38 nous montrent que le taux d'utilisation des machines FH_1 et FH_2 obtenu par les algorithmes génétiques avec ré-ordonnancement est supérieur à celui trouvé sans ré-ordonnancement pour une capacité de files d'attente égale à 2, et un taux de création de pièces supérieur à 1/25. Hors de cet intervalle, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

Les figures et les tableaux 4.37, 4.38 et 3.38 montrent que pour de petites capacités de files d'attente et un taux de création de pièces supérieur ou égale à 1/20, les résultats concernant le taux d'utilisation des machines FH_1 et FH_2 obtenus par toutes les métaheuristiques étudiées avec ré-ordonnancement de pièces contenues dans la station de chargement sont meilleurs que ceux obtenus sans ré-ordonnancement.

4.3.2.5. Le taux d'utilisation de l'AGV

Les figures 4.39 et 4.40 et les résultats du chapitre précédent nous montrent que le ré-ordonnacement de la station de chargement améliore le taux d'utilisation de l'AGV pour un système saturé avec présence de pannes.

Nous pouvons remarquer (voir la figure 4.39) que pour un système saturé et de petites files d'attente, le taux d'utilisation de l'AGV est plus important pour les algorithmes génétiques même avec la présence de pannes.

Pour un système non saturé, il y'a un peu de différence entre les métaheuristiques, dans l'intervalle ([1/30,1/25]), l'utilisation de l'AGV est meilleure pour l'électromagnétisme, pour un taux de création égale à 1/35 les algorithmes génétiques sont les plus efficaces, pour un taux de création égale à 1/40 les colonies de fourmis dépassent les autres.

Taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	17.4	19.65	23	27.2	30.54	30.39	30.52	30.53
Le recuit simulé	17.39	19.56	22.95	27.18	28.81	28.76	28.64	28.84
Les essaims particulaires	17.36	19.64	22.98	27.25	31.97	31.97	31.97	32.01
Les algorithmes génétiques	17.32	19.69	22.98	27.28	32.21	32.24	32.25	32.27
Recherche avec tabous	17.39	19.56	22.93	27.25	30.29	30.25	30.3	30.39
Electromagnétisme métaheuristique	17.38	19.67	23.03	27.31	31.15	30.88	30.99	30.98
DMM modifiée	14.98	17.55	21.19	24.28	24.07	24.54	24.67	24.54
DMM	15	17.61	21.06	14.2	10.94	12.98	10.41	11.59

Tableau 4.39: Le taux d'utilisation de l'AGV pour une capacité de file d'attente =2.

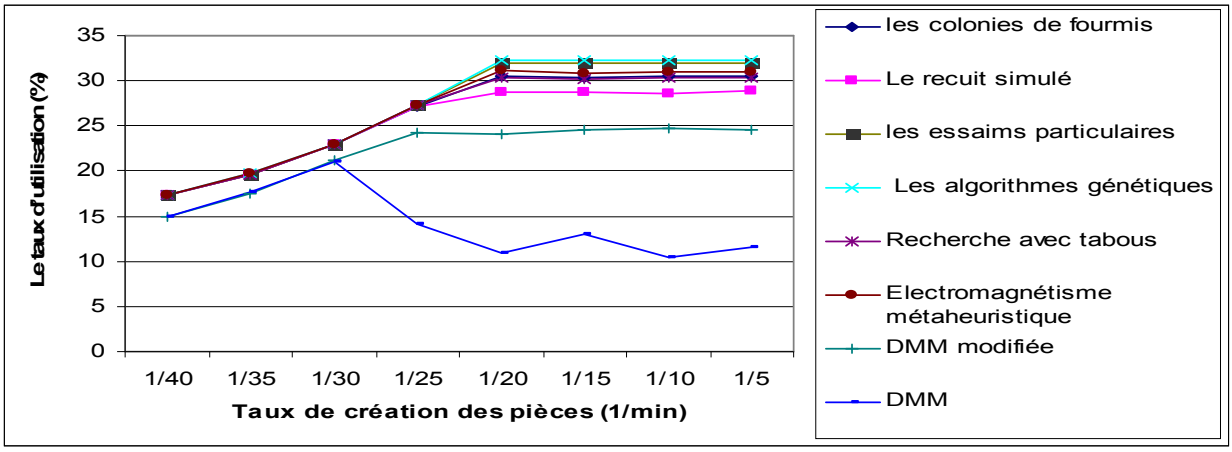

Figure 4.39: Le taux d'utilisation de l'AGV pour une capacité de file d'attente =2

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	17.4	19.73	23.02	27.25	31.78	31.79	31.83	31.85
Avec ré-ordonnancement	17.32	19.69	22.98	27.28	32.21	32.24	32.25	32.27

Tableau 4.40: Le taux d'utilisation de l'AGV obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

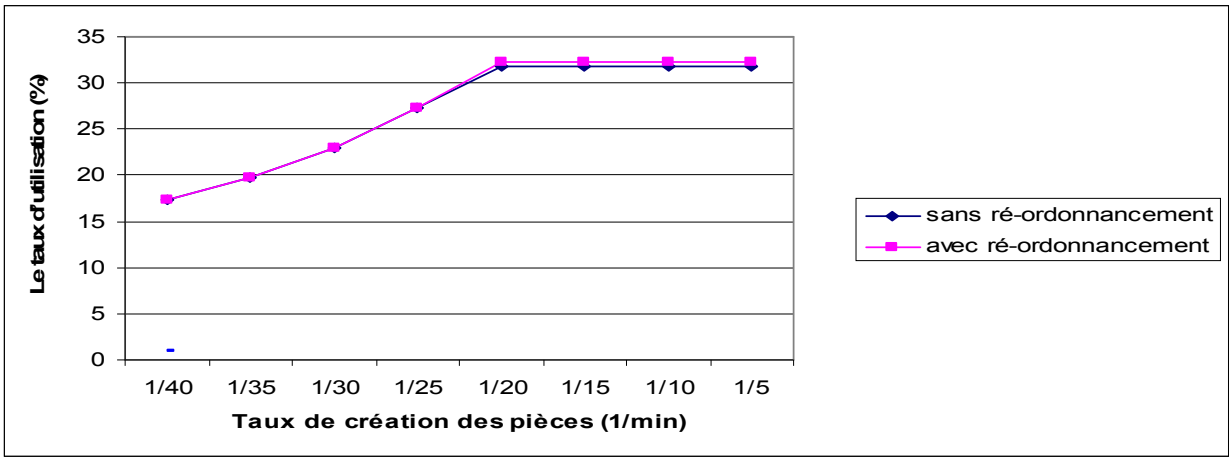

Figure 4.40: Le taux d'utilisation de l'AGV obtenu par les algorithmes génétiques pour une capacité de file d'attente = 2.

Pour les algorithmes génétiques, si on compare le taux d'utilisation de l'AGV si on ré-ordonnance la station de chargement avec celui trouvé sans ré-ordonnancement pour un taux de

création de pièces supérieur à 1/25, nous pouvons dire que le ré-ordonnancement a amélioré les performances concernant l'utilisation de l'AGV. Hors de cet intervalle, les résultats obtenus dans les deux cas sont presque identiques. (Voir la figure 4.40).

4.4. Conclusion

Dans ce chapitre qui est une perspective du chapitre précédent, nous avons adapté les métaheuristiques étudiées pour résoudre le problème de sélection de routage en temps réel dans un FMS avec ré-ordonnancement de la station de chargement et présenté une étude comparative entre ces techniques et les règles de sélection de routages alternatifs DMM et DMM modifiée.

Nous nous intéressons dans cette étude aux différences entre le taux de sortie des pièces, temps de cycle, les en-cours, le taux d'utilisation des machines et de système de transport.

L'analyse présentée dans ce chapitre démontre l'effet du ré-ordonnancement des pièces contenues dans la station de chargement sur les performances du système flexible de production si on utilise les métaheuristiques pour la sélection de routages alternatifs,

De cette analyse nous avons tiré les remarques suivantes :

- L'efficacité de ces métaheuristiques varie d'un objectif à un autre et d'un critère à un autre par exemple pour un taux de production supérieur à 1/20, le taux de production le plus important est trouvé par les algorithmes génétiques pour de petites files d'attente et les essaims particulaires pour de grandes files d'attente.
- Pour le temps de cycle nous ne pouvons pas dire qu'une métaheuristique donne un temps de cycle meilleur que les autres pour toutes les capacités de files d'attente, mais toutes les métaheuristiques améliorent les temps de cycle obtenus par les règles DMM et DMM modifiée pour un système saturé et de petites files d'attente. Dans le cas avec panne, pour un taux de création de pièces supérieur à 1/20 et de petites files d'attente, les temps de cycle obtenus par la recherche tabou sont généralement inférieurs à ceux obtenus par les autres règles et métaheuristiques.
- Ces techniques permettent d'améliorer les en-cours pour les petites files d'attente pour le cas sans panne, et de grandes files d'attente pour le cas avec panne. Dans le cas sans panne, les colonies de fourmis sont les meilleures pour un taux de création superieur à 1/25 et de petites files d'attente, pour de grandes files d'attente et un système saturé, la règle DMM est la plus efficace. Dans le cas avec panne, nous ne pouvons pas dire qu'une métaheuristique est la meilleure pour toutes les capacités de files d'attente.
- Pour le taux d'utilisation des machines, les algorithmes génétiques sont les plus efficaces pour un taux de création de pièces supérieur à 1/20 et de petites files d'attente sauf la machine TP dont l'utilisation est meilleure pour DMM modifiée que pour les autres règles et métaheuristiques.
- Pour le taux d'utilisation de l'AGV, les algorithmes génétiques sont les meilleurs pour un système saturé et de petites files d'attente.

Si on compare les résultats de ce chapitre avec ceux trouvés dans le précédent on peut extraire les conclusions suivantes :

- Les résultats obtenus ont montré que pour chaque métaheuristique que nous avons étudiée, le ré-ordonnancement améliore les performances pour le taux de production, le taux d'utilisation des machines et du système de transport pour un système flexible de production saturé surtout pour de petites capacités de files d'attente, même avec la présence de pannes.
- Pour chaque métaheuristique, pour un taux de création de pièces supérieur à 1/20, le ré-ordonnancement augmente les en-cours, sauf pour la recherche tabou avec de petites files d'attente et les colonies de fourmis.
- Le ré-ordonnancement augmente le temps de cycle pour la plupart des cas d'un système saturé.

Conclusions et perspectives

Les problèmes d'ordonnancement d'ateliers concernent la planification de l'utilisation des ressources disponibles en hommes et en machines, afin de mieux contrôler les coûts et de maîtriser les délais de fabrication des productions décidées.

Ce travail de mémoire se situe dans le cadre de ces problèmes, et plus précisément la manipulation de routages alternatifs en temps réel. Ces problèmes posent des cas extrêmement complexes, et appartiennent à la classe des problèmes NP-difficile pour les quels il n'existe pas d'algorithme polynomial optimal.

Les métaheuristiques sont des algorithmes de type stochastique visant à résoudre une large gamme de ces types de problèmes. L'aspect itératif de ces techniques peut rendre leur utilisation pour résoudre les problèmes complexes en temps réel difficile mais devant le succès rencontré par ces techniques dans le cadre de résolution des autres problèmes NP-difficiles, tout au long de ce travail notre but a été leur adaptation pour résoudre notre problème de sélection de routages alternatifs en temps réel.

Par conséquent, nous avons proposé des algorithmes sur la base de certaines métaheuristiques (les colonies de fourmis, les algorithmes génétiques, le recuit simulé, la recherche tabou, les essaims particulaires, électromagnétisme métaheuristique) pour résoudre le problème afin d'avoir une idée sur leur efficacité et de sélectionner la plus fiable. Pour cela, nous les avons appliqué sur un système flexible de production et nous avons choisi les règles DMM, et DMM modifiée comme une base de comparaison des performances de nos algorithmes.

➢ La synthèse de notre travail

Le travail réalisé a été orienté autour de deux axes :

- La sélection de routages alternatifs en temps réel.
- L'adaptation des métaheuristiques pour la résolution des problèmes NP-difficiles.

Le premier chapitre nous a permit d'exposer la problématique d'ordonnancement dans les systèmes flexibles de production, et de souligner quelques points essentiels : les systèmes flexibles de production, les problèmes d'ordonnancement, l'ordonnancement de type job shop, l'ordonnancement en temps réel…

Dans le deuxième chapitre, après avoir présenté de manière générale les métaheuristiques, en incluant leurs définitions, leurs domaines d'utilisation, leurs caractéristiques et leur classification, nous avons présenté en détail chaque métaheuristique étudiée, nous nous sommes intéressés plus particulièrement à sa définition, son origine et son algorithme de base.

Ensuite, le troisième chapitre présente le modèle FMS de test et les algorithmes proposés pour résoudre le problème de sélection de routages alternatifs en temps réel sans ré-ordonnancement des pièces contenues dans la station de chargement. Des simulations des méthodes DMM et DMM modifiée, et des algorithmes des métaheuristiques étudiées ont été effectuées dans un Core (TM) 2 Duo CPU avec 2.2 GHZ et 1 GO de RAM, en variant le temps d'arrivée (le taux de création) des pièces et en gardant les capacités des files d'attente fixes (et inversement), en tenant en compte les deux cas (avec et sans présence de pannes), en vue de maximiser le taux de sortie des pièces, d'utilisation des machines et de système de transport et de minimiser le temps de cycle et le nombre des en-cours. Les résultats de ces simulations ont mené aux conclusions suivantes:

- La plupart des métaheuristiques que nous avons étudiée ont donné de meilleures performances pour le taux de production, le taux d'utilisation des machines et du système de transport pour un système flexible de production saturé surtout pour de petites capacités de files d'attente même avec la présence de pannes.
- Ces techniques permettent d'améliorer le temps de cycle pour de petites files d'attente (taille de files d'attente égale à 2) et les en-cours pour de petites files d'attente pour le cas sans panne, et de grandes files d'attente pour le cas avec panne.
- Le système sature moins rapidement si on utilise les métaheuristiques.
- L'efficacité de ces métaheuristiques varie d'un objectif à un autre et d'un critère à un autre.

Dans le dernier chapitre, après une analyse de l'impact du ré-ordonnancement des pièces contenues dans la station de chargement sur les performances du système en gardant les mêmes conditions opératoires prises en compte dans le troisième chapitre, nous avons tiré les conclusions suivantes :

- Les résultats obtenus ont montré que pour chaque métaheuristique que nous avons étudiée, le ré-ordonnancement améliore les performances pour le taux de production, le taux d'utilisation des machines et du système de transport pour un système flexible de production saturé surtout pour de petites capacités de files d'attente, même avec la présence de pannes.
- Pour chaque métaheuristique, pour un taux d'arrivée de pièces supérieur à 1/20, le ré-ordonnancement augmente les en-cours, sauf pour la recherche tabou avec de petites files d'attente et les colonies de fourmis.
- Le ré-ordonnancement augmente le temps de cycle pour la plupart des cas d'un système saturé.

➢ Perspectives

L'ordonnancement temps réel est un domaine qui reste ouvert, et dont l'importance ne cesse de croître. Notre travail doit être approfondie et enrichie pour proposer d'autres solutions pratiques au problème de sélection de routages alternatifs en temps réel. Donc, les perspectives ouvertes par ces travaux sont multiples :

- **L'utilisation des métaheuristiques hybrides**

D'après les résultats trouvés, nous avons remarqué que l'efficacité des métaheuristiques étudiées varie d'un objectif à un autre et d'un critère à un autre. Ainsi, une métaheuristique qui a pu

améliorer les performances par rapport à un critère donné et dans le même temps, a donné de mauvais résultats pour un autre critère.

Toutes les métaheuristiques partagent l'avantage qu'elles prêtent à toutes sortes d'extensions. Mais il est souvent impossible de prévoir avec certitude l'efficacité d'une technique donnée quand elle est appliquée.

Donc les métaheuristiques hybrides qui s'efforcent de tirer partie des avantages spécifiques de métaheuristiques différentes en les combinant peuvent améliorer les performances de notre système.

- **Le ré-ordonnancement en temps réel des pièces en cour de traitement**

L'étude expérimentale a montré que le nombre des en-cours est trop petit par rapport au grand nombre des pièces sorties du système, ce qui peut être justifié par la présence de machines goulots (particulièrement T_1 et T_2) donc il est possible d'approfondir le principe de ré-ordonnancement en temps réel en permettant de changer les routages de toutes ou certaines pièces qui sont à l'intérieur des files d'entrée et de sortie des machines, afin de pouvoir minimiser ou éviter ces impasses.

- **Combinaison avec les autres règles de priorités**

Il existe un grand nombre de règles de priorité utilisées dans la recherche et dans l'industrie. Les règles les plus connues sont First In First Out (FIFO), Shortest Processing Time (SPT), Earliest Due Date (EDD), Minimum Slack Time (MST), Slack per Operation (SLACK/OPN), Critical Ratio, Cost Over Time (COVERT)…

Durant notre travail nous avons choisi la règle FIFO pour le séquencement de chaque machine, or l'efficacité des règles de priorité dépend étroitement des critères de performance évalués et des conditions opératoires effectives dans l'atelier …

Ainsi, nous suggérons pour une sélection de routages alternatifs en temps réel de combiner les algorithmes proposés avec d'autres règles de priorité ou avec des approches de sélection dynamique des règles de priorités.

- **L'utilisation des autres coefficients ou fonctions objectifs**

Dans notre travail, pour DMM, et DMM modifiée la dissimilitude entre les routages occupés a été maximisée, à l'aide des métaheuristiques nous avons équilibré la charge des routages non pas en terme de nombre de pièces mais en terme de temps opératoire (maximiser le produit des charges de routages). Mais, nous n'avons aucune garantie que les coefficients utilisés sont optimaux.

L'effet des différents coefficients pourra donc être considéré dans les futurs travaux, en les analysant un par un ou en combinant deux ou plusieurs coefficients.

- **La généralisation des solutions proposées**

Dans notre travail, nous avons remarqué qu'une métaheuristique peut améliorer les performances par rapport à un objectif donné et dans le même temps, donner de très mauvais résultats pour un autre objectif. En plus, les résultats de simulation trouvés sont fortement dépendants de la structure du système étudié, et surtout des données. Donc, si certains conditions opératoires changent tel que la modification du nombre de machines, ou des routages identifiées

pour un ou plusieurs types de pièces, la disposition des stations …, les métaheuristiques appliquées jusque là peuvent ne plus être appropriées à la nouvelle situation.

Pour remédier à ces inconvénients, nous proposons d'aborder le problème avec d'autres ateliers, afin de pouvoir mieux cerner le problème et éventuellement trouver des conclusions générales et fournir un ensemble de techniques ou des méthodes de diagnostic et de correction des symptômes à l'utilisateur qui, en fonction de ses critères et objectifs d'optimisation pourra choisi telle ou telle métaheuristique.

Annexe A

Le temps de calcul

Le temps de calcul est un critère très important dans la mesure des performances des métaheuristiques.

L'analyse présentée dans cet annexe montre la variation du temps de calcul en fonction du taux de création de pièces et en fonction des capacités de files d'attente.

Pour chaque métaheuristique, le temps de calcul est mesuré en milliseconde qui reste très faible par rapport aux durées opératoires du problème, mais il existe un peu de différences entre les métaheuristiques si on s'intéresse au temps de calcul.

A.1. Cas sans ré-ordonnancement de la station de chargement

A.1.1. Étude comparative sans présence de pannes

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	0.016	0.015	0.023	0.022	0.024	1.59	1.63	4.38
Le recuit simulé	0.021	0.021	0.015	0.015	0.111	0.121	0.111	0.114
Les essaims particulaires	0.029	0.026	0.029	0.027	0.045	0.583	0.616	0.584
Les algorithmes génétiques	0.031	0.029	0.021	0.027	0.030	3.231	3.262	3.325
Recherche avec tabous	0.030	0.024	0.027	0.031	0.026	0.716	0.702	0.701
Electromagnétisme métaheuristique	0.025	0.022	0.028	0.032	0.037	0.66	0.638	0.632

Tableau A.1: Temps de calcul pour une capacité de file d'attente = 2.

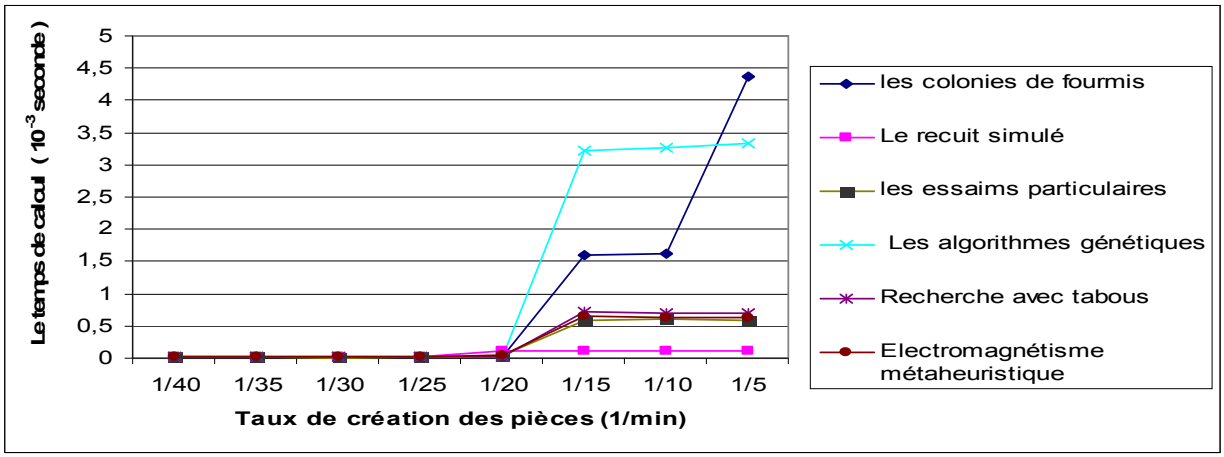

Figure A.1: Temps de calcul pour une capacité de file d'attente = 2

La courbe de figure A.1 et le tableau A.1 montrent que pour une taille de file d'attente égale à 2, les temps de calcul des colonies de fourmis et les algorithmes génétiques sont supérieurs à ceux

des autres métaheuristiques étudiées pour un taux de création de pièces supérieur à 1/20 où le recuit simulé a le meilleur temps de calcul.

Pour un taux de création de pièces inférieur ou égale à 1/20, nous pouvons dire que les temps de calcul sont pratiquement identiques pour toutes les métaheuristiques.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	4.38	2.824	4.89	6.934
Le recuit simulé	0.114	0.1574	0.1737	0.2192
Les essaims particulaires	0.584	0.703	0.863	1.084
Les algorithmes génétiques	3.325	3.798	4.149	4.813
Recherche avec tabous	0.701	1.033	1.38	1.804
Electromagnétisme métaheuristique	0.632	0.691	0.841	0.922

Tableau A.2: Temps de calcul pour un taux de création de pièces =1/5 (1/min)

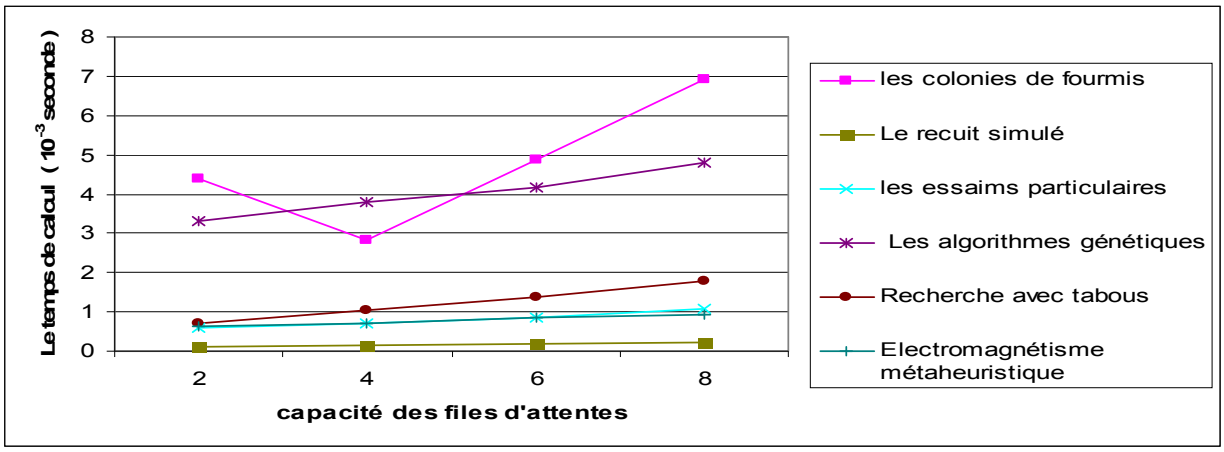

Figure A.2: Temps de calcul pour un taux de création de pièces =1/5 (1/min)

La figure et le tableau A.2 montrent que si on varie les capacités de files d'attente en gardant le taux de création fixe, nous pouvons remarquer l'augmentation du temps de calcul pour les colonies de fourmis et les algorithmes génétiques par rapport aux autres métaheuristiques pour un système saturé.

Nous constatons aussi que l'accroissement des capacités de files d'attente fait accroître le temps de calcul pour la plupart des métaheuristiques étudiées.

A.1.2. Étude comparative avec présence de pannes

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	0.022	0.027	0.018	0.024	0.217	0.234	0.255	0.237
Le recuit simulé	0.017	0.012	0.017	0.024	0.121	0.125	0.110	0.266
Les essaims particulaires	0.031	0.030	0.027	0.028	0.573	0.588	0.597	0.601
Les algorithmes génétiques	0.028	0.031	0.027	0.029	3.45	3.26	3.36	3.275
Recherche avec tabous	0.029	0.017	0.029	0.029	0.725	0.740	0.728	0.736
Electromagnétisme métaheuristique	0.028	0.020	0.029	0.028	0.627	0.604	0.597	0.600

Tableau A.3: Temps de calcul pour une capacité de file d'attente = 2.

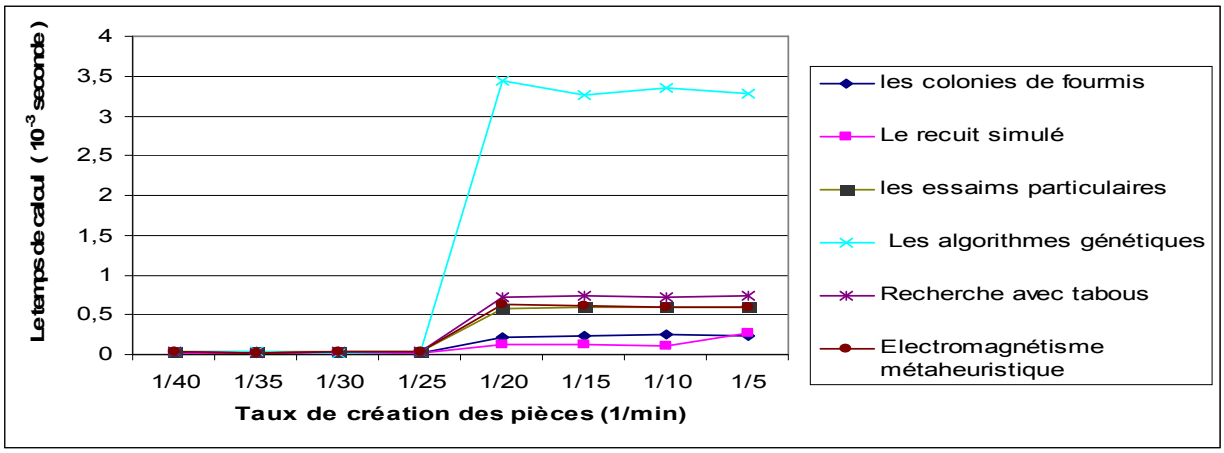

Figure A.3: Temps de calcul pour une capacité de file d'attente = 2

La courbe de figure A.3 et le tableau A.3 montrent que pour une taille de file d'attente égale à 2, les temps de calcul du recuit simulé sont meilleurs que ceux des autres métaheuristiques étudiées pour un taux de création de pièces supérieur à 1/25 où les algorithmes génétiques ont le plus mauvais temps de calcul.

Pour un taux de création de pièces inférieur ou égale à 1/25, les temps de calculs des métaheuristiques sont pratiquement les mêmes.

Capacité de file d'attente	2	4	6	8
Les colonies de fourmis	0.237	2.824	4.89	6.93
Le recuit simulé	0.266	0.14	0.128	0.222
Les essaims particulaires	0.601	0.73	0.912	1.013
Les algorithmes génétiques	3.275	3.85	4.15	4.51
Recherche avec tabous	0.736	0.995	1.415	1.789
Electromagnétisme métaheuristique	0.600	0.691	0.8411	0.922

Tableau A.4: Temps de calcul pour un taux de création de pièces =1/5 (1/min)

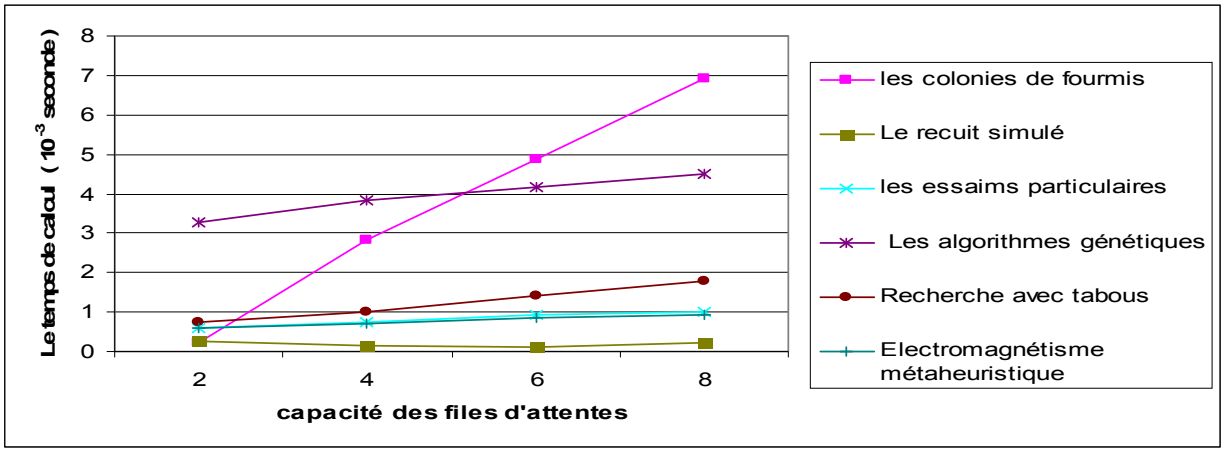

Figure A.4: Temps de calcul pour un taux de création de pièces =1/5 (1/min)

La figure et le tableau A.4 montrent que le temps de calcul pour le recuit simulé reste meilleur par rapport aux autres métaheuristiques, pour un système saturé, même avec présence de pannes.

Nous pouvons aussi remarquer que les temps de calcul les plus mauvais sont ceux des algorithmes génétiques et des colonies de fourmis et que pour la plupart des métaheuristiques étudiées l'augmentation des capacités de files d'attente augmente le temps de calcul.

A.2. Cas avec ré-ordonnancement de la station de chargement

A.2.1. Étude comparative sans présence de pannes

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	0.017	0.017	0.021	0.025	0.022	1.059	1.04	1.06
Le recuit simulé	0.014	0.017	0.015	0.017	0.020	0.112	0.104	0.177
Les essaims particulaires	0.029	0.027	0.043	0.027	0.048	0.613	0.621	0.638
Les algorithmes génétiques	0.028	0.026	0.023	0.024	0.029	3.338	3.295	3.368
Recherche avec tabous	0.032	0.027	0.029	0.027	0.035	0.692	0.690	0.744
Electromagnétisme métaheuristique	0.028	0.028	0.029	0.026	0.028	0.631	0.615	0.632

Tableau A.5: Temps de calcul pour une capacité de file d'attente = 2.

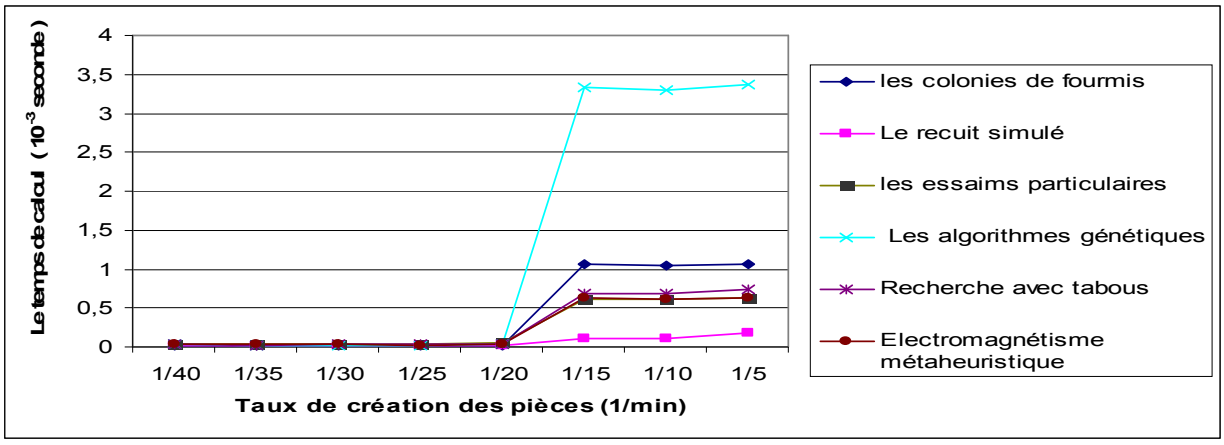

Figure A.5: Temps de calcul pour une capacité de file d'attente = 2

La courbe de figure A.5 et le tableau A.5 montrent que pour une taille de file d'attente égale à 2, les temps de calcul des colonies de fourmis et les algorithmes génétiques sont toujours supérieurs à ceux des autres métaheuristiques étudiées pour un taux de création de pièces supérieur à 1/20 où le recuit simulé a le meilleur temps de calcul, même si réordonnance la station de chargement.

Pour un taux de création de pièces inférieur ou égale à 1/20, toutes les métaheuristiques donnent pratiquement les mêmes temps de calcul.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	0.031	0.029	0.021	0.027	0.030	3.231	3.262	3.325
Avec ré-ordonnancement	0.028	0.026	0.023	0.024	0.029	3.338	3.295	3.368

Tableau A.6: Temps de calcul des algorithmes génétiques pour une capacité de file d'attente = 2

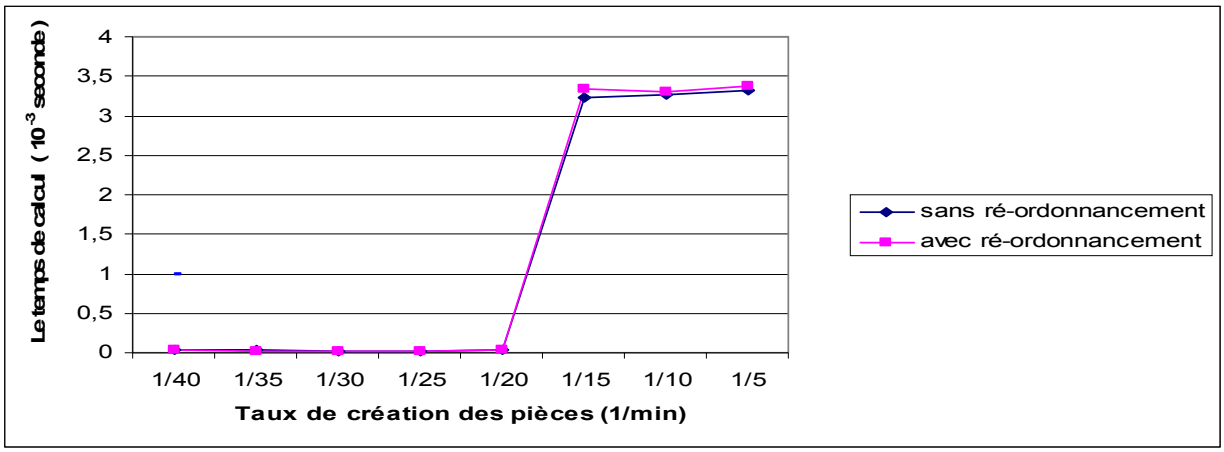

Figure A.6: Temps de calcul des algorithmes génétiques pour une capacité de file d'attente = 2

La figure et le tableau A.6 montrent qu'il y'a une petite augmentation du temps de calcul des algorithmes génétiques dans le cas avec ré-ordonnancement par rapport à celui trouvé sans ré-ordonnancement pour un taux de création de pièces supérieur à 1/20, et une capacité de files d'attente égale à 2. Si non pour les autres cas les résultats sont presque identiques.

A partir des figures et des tableaux A.1, A.5 et A.6, nous pouvons remarquer que si on veut généraliser pour toutes les métaheuristiques, les résultats sont pratiquement identiques.

A.2.2. Étude comparative avec présence de pannes

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Les colonies de fourmis	0.014	0.024	0.022	0.017	1.078	1.084	1.211	2.41
Le recuit simulé	0.018	0.018	0.019	0.020	0.135	0.117	0.112	0.117
Les essaims particulaires	0.029	0.030	0.026	0.031	0.578	0.613	0.593	0.617
Les algorithmes génétiques	0.033	0.03	0.024	0.028	3.32	3.208	3.23	3.360
Recherche avec tabous	0.026	0.028	0.025	0.032	0.694	0.704	0.725	0.728
Electromagnétisme métaheuristique	0.025	0.031	0.033	0.029	0.596	0.622	0.607	0.625

Tableau A.7: Temps de calcul pour une capacité de file d'attente = 2.

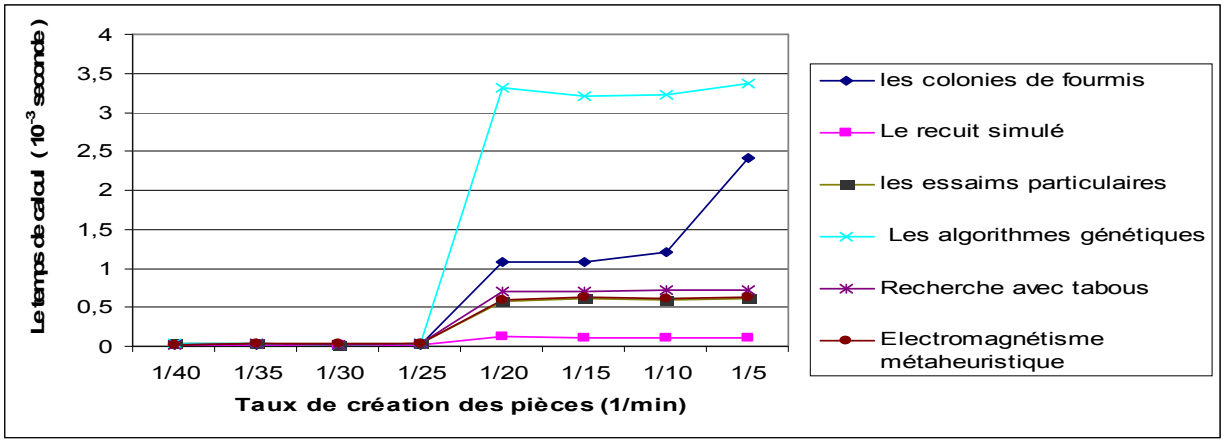

Figure A.7: Temps de calcul pour une capacité de file d'attente = 2

La courbe de figure A.7 et le tableau A.7 montrent que pour une taille de file d'attente égale à 2, les temps de calcul du recuit simulé sont toujours meilleurs que ceux des autres métaheuristiques

étudiées pour un taux de création de pièces supérieur à 1/25 et les algorithmes génétiques ont le plus mauvais temps de calcul, même avec la présence de pannes.

Pour un taux de création de pièces inférieur ou égale à 1/25, les temps de calcul des métaheuristiques sont presque identiques.

taux de création des pièces (1/min)	1/40	1/35	1/30	1/25	1/20	1/15	1/10	1/5
Sans ré-ordonnancement	0.028	0.031	0.027	0.029	3.45	3.26	3.36	3.275
Avec ré-ordonnancement	0.033	0.03	0.024	0.028	3.32	3.208	3.23	3.360

Tableau A.8: Temps de calcul des algorithmes génétiques pour une capacité de file d'attente = 2

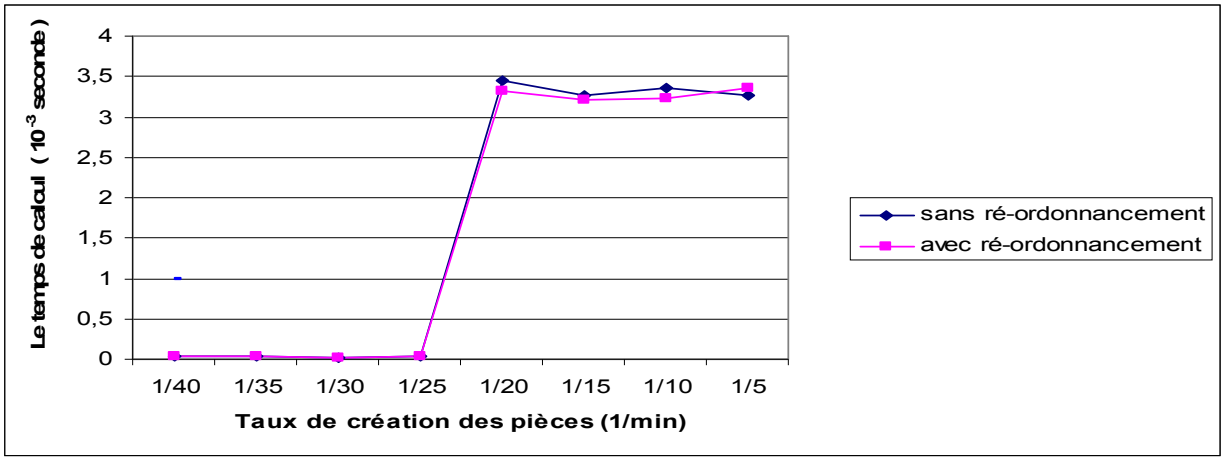

Figure A.8: Temps de calcul des algorithmes génétiques pour une capacité de file d'attente = 2

La figure et le tableau A.8 montrent que si on compare le temps de calcul des algorithmes génétiques dans le cas avec ré-ordonnancement et celui trouvé sans ré-ordonnancement pour un taux de création de pièces supérieur à 1/25, et une capacité de files d'attente égale à 2, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

A partir des figures et des tableaux A.2, A.7 et A.8, nous pouvons remarquer que si on veut généraliser pour toutes les métaheuristiques, nous ne pouvons pas dire qu'un cas est meilleur que l'autre.

Annexe B

Abréviations:

FMS	Flexible manufacturing system
EPL	Equal probability loading
FA	First available
SQ	Shortest queue
LWQ	Least work in queue
DMM	Dissimilarity maximisation method
NP	Non polynomial
CPU	Central processing unit
MOCN	Machines outils à commande numérique
CN	Commande numérique
MF	Module flexible
CF	Cellule flexible
GF	Groupe flexible
SF	Système flexible
LF	Ligne flexible
SFM	Single flexible machine
FMC	Flexible manufacturing cell
MMFMS	Multi-machine flexible manufacturing systems
MCFMS	Multi-cell flexible manufacturing system
OF	Ordres de fabrication
SPT	Short processing time
EDD	Earliest due date
CR	Critical ratio
FCFS	First come first served
FIFO	First in First out

SA	Simulated annealing
TS	Taboo search
GA	Genetic algorithms
ACO	Ant colony optimisation
PSO	Particle swarm optimisation
EM	Electromagnetism like method
AS	Ant system
ACS	Ant colony system
MMAS	Max min ant system
OEP	Optimisation par éssaims particulaires
MST	Minimum Slack Time
SLACK/OPN	Slack per Operation
COVERT	Cost Over Time
FV	Fraiseuse verticale.
FH	Fraiseuse horizontale.
T	Tour.
TP	Toupie.
SC	Station de chargement.
SD	Station de déchargement.

Annexe C

Variables et symboles :

Chapitre 1 :

r_i	Date d'arrivée du job *i* dans le système.
d_i	Date échue du job *i* .
Ci	Date de fin effective du job *i.*
D_{ij}	Coefficient de dissimilitude entre les routages *i* et *j*.
X_j	L'état du routage (sélectionné ou non).

Chapitre 2 :

T	La température.
E	L'énergie.
ΔE	La variation de l'énergie.
$V(s)$	L'ensemble des voisins de s.
$Liste\_tab$	La liste taboue.
$F(x)$	La fonction objectif.
j_i^k	La liste des successeurs de la ville *i* visitées par la fourmi *k.*
$\tau_{ij}^k(t)$	L'intensité de la piste *i,j.*
η_{ij}	La visibilité de la piste *i,j.*
$T^k(t)$	Le trajet effectué par la fourmi *k* à l'itération *t.*
$L^k(t)$	La longueur de la tournée effectuée par la fourmi *k* à l'itération *t.*
ρ	Le taux d'évaporation.
$\tau_{\min}$	L'intensité maximale de la piste.
$\tau_{\max}$	L'intensité minimale de la piste.
$v_i(t)$	La vélocité de la particule *i* à l'itération *t.*
$X_i(t)$	La position de la particule *i* à l'itération *t.*

p_i	La mémoire de la particule *i* à la génération courante.
p_g	La meilleure solution trouvée par l'essaim.
c_1	Le coefficient cognitif.
c_2	Le coefficient social.
r_1 *et* r_2	Les coefficients d'accélération.
W	Probabilité de modification.
F_1	Opérateur de modification.
F_2 *et* F_3	Des opérateurs de croisements.
q_i	La charge de particule *i*.
F_i	La force de la particule *i*.
l_k	La limite minimale de déplacement.
u_k	La limite maximale de déplacement.
Avg	Le moyen des fonctions objectifs.

Références bibliographiques

[Adamou 97] Adamou, M., (1997). Contribution à la modélisation en vue de la conduite des systèmes flexibles d'assemblage à l'aide des réseaux de Petri orientés objet, *Thèse de doctorat, Université de Franche-Comté.*

[Akyol 07] Akyol, D.E. and Araz, O.U. (2007). A Neural Network Based Decision Support System for Real-Time Scheduling of Flexible Manufacturing Systems, *book chapter in: Operations Research Proceedings 2007 , book series: Operations Research Proceedings Volume 2007, Part IV, 83-88.*

[AMMONS 88] Ammons, J. C., Govindaraj, T. and Mitchell, C. M. (1988). Decision models for aiding FMS scheduling and control, *IEEE Transactions on Systems, Man, and Cybernetics, 18(5), 744-756.*

[Artiba 97] Artiba, A. et Aghezzaf, E. H. (1997). An architecture of a multi-model system for planning and scheduling, *International Journal of Computer Integrated Manufacturing, 10, 5, 380-393.*

[Askin 93] Askin, R.G et Standridge, C.R. (1993). Modeling and Analysis of Manufacturing System, *John Wiely & Sons, Ed.*

[Aydin 00] Aydin, M. E. and Öztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents, *Robotics and Autonomous Systems 33 (2000) 169-178.*

[Aydin 04] Aydin, M. E. and Fogarty, T. C. (2004). A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application, *Journal of Intelligent Manufacturing, 15(6),805-814.*

[Baek 99] Baek, D.C., Oh, S.Y. et Yoon, W.C. (1999). A visualized human-computer interactive approach to job shop scheduling. *International Journal of Computer Integrated Manufacturing, 12(1), 75-83.*

[Beckers 92] Beckers, R., Deneubourg, J. L., and Goss, S. (1992). Trails and U-Turns in the Selection of a Path by the Ant Lasius Niger, *J. Theor. Biol., 159:397-415.*

[Bel 88] Bel, G., Bensana, E. et Dubois, D. (1988). Construction d'ordonnancements prévisionnels : un compromis entre approches classiques et systèmes experts, *Revue d'Automatique, Productique, Informatique Industrielle, 22, 5, 509-536.*

[Bensana 88] Bensana, E., Bel, G. et Dubois, D. (1988). OPAL: a multi-knowledge based system for industrial job-shop scheduling, *International Journal of Production Research, 26, 3, 795-815.*

[Bérard 98] Bérard, C., Grabot, B. et Nguyen, P. (1998). Coopérations Recherche/Entreprises pour la conception de logiciels d'ordonnancement : SipaPlus, Io, TAPAS, *Revue Française de Gestion Industrielle, vol.17, n°4, 1998, pp. 21-38.*

[Bilkay 04] Bilkay , O., Anlagan , O. et Kilic, S. E. (2004). Job shop scheduling using fuzzy logic, *International journal of advanced Manufacturing Technology, 23, 606-619.*

[Birbil 03] Birbil, S.I., and FANG, S., (2003). An Electromagnetism-like Mechanism for Global Optimization, *Journal of Global Optimization vol 25: 263–282.*

[Bistline 98] Bistline, W.G.Sr, Banerjee, S. and Banerjee, A. (1998). An interactive decision support system for solving real time scheduling problems considering customer and job priorities with scheduling interruptions , *Computers Ops Res,25(11),981-995.*

[Blackstone 82] Blackstone, J.H., Phillips, D.T. et Hogg, G.L. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations, *International Journal of Production Research*, Vol. 20, No. 1, 27-45.

[Bonabeau 99] Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). Swarm Intelligence, From Natural to Artificial Systems, *Oxford University Press.*

[Boryczka 04] Boryczka, U., (2004). Ant Colony System for JSP, *Book chapter in: Cellular Automata, Book series: Lecture Notes in Computer Science, 3305, 296-305.*

[Boucon 91] Boucon, D. (1991). Ordonnancement d'atelier : Aide au choix des règles de priorité, *Thèse de Doctorat en Automatique, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Avril 1991.*

[Boukachour 97] Boukachour, J. Galinho, T. et Pécuchet, J.-P. (1997). Un réseau de neurones pour bien sélectionner les heuristiques d'ordonnancement, *MOSIM 97, 1ère Conférence Francophone sur la Modélisation et la Simulation, Rouen, France, 5-6 Juin, 1997.*

[Breskvar 03] Breskvar, U., Kljajic, M. (2003). Interactive scheduling with genetic algorithms and visual event simulation model, Computer as a Tool. *The IEEE Region 8 Volume 1, 429 - 432.*

[Buyurgan 04] Buyurgan, N., Saygin, C. and Kilic, S.E, (2004). Tool allocation in flexible manufacturing systems with tool alternatives, *Robotics and Computer-Integrated Manufacturing 20, 341–349.*

[Buyurgan 06a] Buyurgan, N. and Mendoza, A. (2006). A Multi-objective scheduling framework for flexible manufacturing systems. *International Journal of Production Research, 44(7),1273-95.*

[Buyurgan 06b] Buyurgan, N. and Saygin, C. (2006). An integrated control framework for flexible manufacturing systems. *International Journal of Advanced Manufacturing Technology, 27, 1248-1259.*

[Buyurgan 08] Buyurgan, N. and Saygin, C. (2008). Application of the analytical hierarchy process for real-time scheduling and part routing in advanced manufacturing systems. *Article in press in: Journal of Manufacturing Systems (2008), doi:10.1016/j.jmsy.2008.08.002.*

[Byrne 97] Byrne, M. D. and Chutima, P. (1997). Real-time operational control of an FMS with full routing flexibility, *International journal of Production Economics 51, 109-113.*

[Cakar 05] Cakar, T. and Yildirim, M.B. (2005). A neuro-genetic approach to design and planning of a manufacturing cell, *Journal of Intelligent Manufacturing, 16, 453-462.*

[Caprihan 04] Caprihan, R., Wadhwa, S. and Kumar, S. (2004). On the consequences of information delays in the scheduling of semi-automated flexible machines. *International Journal of Flexible Manufacturing Systems, 16(3), 251–275.*

[Caprihan 05] Caprihan, R. and Wadhwa, S. (2005). Scheduling of FMS with information delays-A simulation study. *International Journal of Flexible Manufacturing Systems 17(1), 39-65.*

[Caprihan 06] Caprihan, R., Kumar, A., Stecke, K.E. (2006). A fuzzy dispatching strategy for due-date scheduling of FMSs with information delays, *International Journal of Flexible Manufacturing Systems, 18(1), 29-53.*

[Caprihan 97] Caprihan, R. and Wadhwa, S. (1997). Impact of routing flexibility on the performance of an FMS-A simulation study. *International Journal of Flexible Manufacturing Systems, 9(3),273-298*

[Carlier 88] Carlier J., et Chretienne, P. (1988). Problèmes d'ordonnancement - modélisation - complexité - algorithmes, *ERI Masson*.

[Chan 04] Chan, F.T. S. and Chan, H. K. (2004). A comprehensive survey and future trend of simulation study on FMS scheduling, *Journal of Intelligent Manufacturing, 15(1), 87- 102.*

[Chen 07] Chen, S. Chang, P. Chan, C. and Mani., V. (2007). A Hybrid Electromagnetism-Like Algorithm for Single Machine Scheduling Problem. *Book chapter in Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Volume 4682.*

[Cho 95] Cho, H. and Wysk, R. A. (1995). Intelligent workstation controller for computer-integrated manufacturing : problems and models, *Journal of Manufcturing Systems, 14(4), 252-263.*

[Christian 98] Christian, A. (1998). Ordonnancement en temps réel d'atelier avec prise en compte des temps de préparation des ressources, *thèse PhD, Laboratoire d'Analyse et d'Architecture des Systèmes du C.N.R.S, INSA.*

[Clerc 04] Clerc, M. et Siarry, P. (2004), Une nouvelle métaheuristique pour l'optimisation difficile: la méthode des essaims particulaires, *J3eA - Vol. 3 - 7 (2004).*

[Colorni 92] Colorni, A., Dorigo, M., and Maniezzo, V. (1992). Distributed Optimization by Ant Colonies. *In Varela, F. and Bourgine, P., editors, Proceedings of ECAL'91 - First European Conference on Arti_cial Life, pages 134-142, Paris, France. Elsevier Publishing.*

[Darwin 59] Darwin, C. (1859). On The Origin of Species by Means of Natural Selection or the Preservation of Favored Races in the Struggle for Life, *Murray, London. (in* Dréo, J., Pétrowski, A., Siarry, P. et Thaillard, E. (2006). Metaheuristics for Hard Optimization, *Springer 2006).*

[Das 97] Das, S. K. and Nagendra, P. (1997). Selection of routes in a flexible manufacturing facility, *International Journal of Production Economics, 48.237-247.*

[Demmou 77] Demmou, R. (1977). Etude de familles remarquables d'ordonnancement en vue d'une aide à la décision, *Thèse de docteur-ingénieur présentée à l'Université Paul Sabatier de Toulouse.*

[Dominic 04] Dominic, P. D. D., Kaliyamoorthy, S. and Kumar, M. S. (2004). Efficient dispatching rules for dynamic job shop scheduling, *International Journal of Advanced Manufacturing Technology (2004) 24, 70-75.*

[Dorigo 96] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System : Optimization by a Colony of Cooperating Agents, *IEEE Trans. Syst. Man Cybern, B(26) :29-41.*

[Dorigo 97a] Dorigo, M. and Gambardella, L. M. (1997). Ant Colonies for the Traveling Salesman Problem, *BioSystems, 43 :73-81.*

[Dorigo 97b] Dorigo, M. and Gambardella, L. M. (1997b). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Trans. Evol. Comp., 1:53-66.*

[Doumeingts 83] Doumeingts, G., Breuil, D., et Pun, L. (1983). La gestion de production assistée par ordinateur, *Hermés*.

[Dréo 03] Dréo, J., Pétrowski, A., Siarry, P. et Thaillard, E. (2003). Métaheuristiques pour l'optimisation difficile, *Eyrolles 2003.*

[Dréo 04] Dréo, J. (2004), Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical, *thèse de doctorat 2004, université Paris12.*

[Dunkler 88] Dunkler, O., Mitchell, C.M., Govindaraj, T. et Ammons, J.C. (1988). The effectiveness of supervisory control strategies in scheduling flexible manufacturing systems, *IEEE Trans Syst Man Cybern, March–April, 18, 223-237.*

[Eberhart 95] Eberhart, R. C. and Kennedy, J. (1995). New optimizer using particle swarm theory, *Proceedings of the 6th International Symposium on Micro Machine and Human Science, pp. 39–43, Nagoya, Japan.*

[ElMekkawy 03] ElMekkawy, T. Y. and ElMaraghy, H.A. (2003). Real-time scheduling with deadlock avoidance in flexible manufacturing systems, *International Journal of Advanced Manufacturing Technology, 22, 259-270.*

[Erkmen 97] Erkmen, A.M., Erbudak, M., Anlagan, O. and Unver, O. (1997). Genetically Tuned Fuzzy Scheduling for Flexible Manufacturing System, *Proceedings of the 1997 IEEE International Conference on Robotics and Automation Albuquerque, New Mexico - April 1997.*

[Erschler 76] Erschler, P., (1976). Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnancement, *Thèse de doctorat d'Etat présentée à l'Université Paul Sabatier de Toulouse.*

[Esquirol 87] Esquirol, P. (1978). Règles et processus d'inférence pour l'aide à l'ordonnancement de tâches en présence de contraintes, *Thèse de doctorat présentée à l'Université Paul Sabatier de Toulouse.*

[Esquirol 97] Esquirol, P., Lopez, P., Haudot, L. and Sicard, M. (1997). Constraint-oriented cooperative scheduling for aircraft manufacturing, *IEEE Expert 12(1), 32- 39.*

[Esquirol 99] Esquirol, P. et Lopez, P. (1999). L'ordonnancement, *Economica, 1999.*

[Farhoodi 90] Farhoodi, F. (1990). A knowledge-based approach to dynamic job-shop scheduling. *International Journal of Computer Integrated Manufacturing, 3(2), 84-95.*

[Fraser 57] Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers, *Australian Journal of Biological Sciences, 10:484-491*

[Froment 88] Froment, B. (1988). Gestion en temps réel d'atelier flexible : analyse et contribution à l'optimisation, application au pilotage des services logistiques, *Thèse de doctorat présentée à l'Ecole Centrale de Paris.*

[Gao 05] Gao, Q., Luo,X., and Yang, S. (2005). Stigmergic cooperation mechanism for shop floor control system, *International Journal of Advanced Manufacturing Technology (2005) 25, 743-753.*

[Ge 06] Ge, H.W., Lu, Y.H., Zhou, Y., Guo, X.C. and Liang, Y.C. (2006). A novel particle swarm optimization-based approach for job-shop scheduling, *Book chapter in: Computational Methods, 1093-1098.*

[Ge 07] Ge, H., Du, W., Qian, F., and Wang, L. (2007). An Intelligent Hybrid Algorithm for Job-Shop Scheduling Based on Particle Swarm Optimization and Artificial Immune System, *Book chapter in: Analysis and Design of Intelligent Systems using Soft Computing Techniques, Book series: Advances in Soft Computing, 41, 628-637.*

[Ghomri 07] Ghomri, L. et Sari, Z. (2007). Influence des Contraintes et des Perturbations sur les Performances des Règles de Routage dans un FMS, *Rabat, la conférence internationale Conception et Production Intégrées.*

[Glover 97] Glover, F. and Laguana, M. (1997). Tabu Search, *Article was principally adapted from the book « Tabu Search » of Glover, Kluwer Academics publishers*

[Goldberg 89] Goldberg, E.E. (1989), Genetic Algorithms in Search, Optimization, and Machine Learning, *Addison Wesley, Reading, MA.*

[Goss 89] Goss, S., Aron, S., Deneubourg, J. L., and Pasteels, J. M. (1989). Self- Organized Shortcuts in the Argentine Ant. *Naturwissenchaften, 76:579-581.*

[Grabot 99] Grabot, B., Bérard, C. et Nguyen, P. (1999). An implementation of man-software cooperative scheduling: the IO software, *Production Planning and Control, vol.10, n°3, 1999, pp. 238-250.*

[Grave 81] Graves, S.C. (1981). A review of production scheduling, *Operations Research, Vol. 29, No. 4, 646-675.*

[Guo 08] Guo, Z. X., Wong, W. K., Leung, S. Y. S., Fan, J. T. and Chan, S. F. (2008). A genetic-algorithm-based optimization model for scheduling flexible assembly lines, *International Journal of Advanced Manufacturing Technology (2008) 36,156-168.*

[Hassam 06] Hassam, A. (2006). Manipulation des routages alternatifs en temps réel dans les systèmes flexibles de production, *mémoire de magister, université Abou Bekr Belkaid, Tlemcen.*

[Hassam 07] Hassam, A. and Sari, Z. (2007). Real-Time Selection of Process Plan in Flexible Manufacturing Systems: Simulation Study, *CHINA, International Conference on Industrial engineering and Systems Management.*

[Hatchuel 92] Hatchuel, A., Sardas, J.C., (1992). Les grandes transitions contemporaines des systèmes de production, une démarche typologique, *In : De Tressac, G, Dubois, P., Les nouvelles rationalisations de la production - Toulouse : Cépaduès-Editions, 1992, p. 1-24.*

[Holland 75] Holland, J.H. (1975), Adaptation in Natural and Artificial Systems, *University of Michigan Press, Ann Arbor.*

[Horst 95] Horst, R. and Pardolos, P. M. (1995). Handbook of Global Optimization, *Kluwer Academic Publishers.*

[Huang 08] Huang, R. and Yang, C. (2008). Ant colony system for job shop scheduling with time windows, *International Journal of Advanced Manufacturing Technology 39,151-157.*

[Ishii 96] Ishii, N. and Muraki, M. (1996). A process-variability-based on-line scheduling system in multi product batch process, *Computing in Chemical Engineering, 20, 217-234.*

[Jerald 06] Jerald, J., Asokan, P., Saravanan, R., Rani, A.D.C. (2006). Simultaneous scheduling of parts and automated guided vehicles in an FMS environment using adaptive genetic algorithm, *International Journal of Advanced Manufacturing Technology 29, 584–589.*

[Kang 07] Kang, K., Zhang, R. and 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, 823-835.*

[Karaouzene 07] Karaouzene, Z., Sari, Z. (2007). La gestion des files d'attente par regles de priorités, *4th International Conference on Computer Integrated Manufacturing CIP'2007, Sétif, Algérie.*

[Karaouzene 08] Karaouzene, Z., Sari, Z. (2008). Ordonnancement et Gestion des Files d'Attentes par les Règles de Priorités dans un Système Flexible de Production, *7e Conférence Internationale de modélisation et simulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France.*

[Kazerooni 97] Kazerooni, A. Chan, F. T. S. and Abhary, K. (1997). A fuzzy integrated decision-making support system for scheduling of FMS using simulation, *Computer Integrated Manufacturing Systems, 10 (1), 27-34.*

[Kılıç 06] Kılıç, S., and Kahraman, C., (2006). Metaheuristic Techniques for Job Shop Scheduling Problem and a Fuzzy Ant Colony Optimization Algorithm, *Book chapter in: Fuzzy Applications in Industrial Engineering, Book series: Studies in Fuzziness and Soft Computing, 201, 401-425.*

[Kim 08] Kim, I., Watada, J. and Shigaki, I. (2008). A comparison of dispatching rules and genetic algorithms for job shop schedules of standard hydraulic cylinders, *Soft Comput (2008) 12:121–128.*

[Kirkpatrick 83] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983).Optimization by Simulated Annealing, *Science, volume 220, 4598, p .671-680.*

[Kouider 07] Kouider, A. et Bouzouia, B. (2007). Approche Multi-agents Basée sur le Recuit Simulé pour le Problème d' Ordonnancement Distribué, *4th International Conference on Computer Integrated Manufacturing CIP'2007, Sétif, Algérie.*

[Kouiss 97] Kouiss, K., Pierreval, H. and Mebarki, N. (1997). Using multi-agent architecture in FMS for dynamic scheduling, *Journal of Intelligent Manufacturing (1997) 8, 41-47.*

[Le Gall 89] Le Gall, A. (1989). Un système interactif d'aide à la décision pour l'ordonnancement et le pilotage temps réel d'atelier, *Thèse présentée à l'Université Paul Sabatier de Toulouse.*

[Lei 08] Lei, D., (2008). A Pareto archive particle swarm optimization for multi-objective job shop scheduling, *Computers & Industrial Engineering, 54, 960-971.*

[Lereno 01] Loreno, E., Morello, B. et Baptiste, P. (2001). Système d'aide au paramétrage d'un logiciel d'ordonnancement, *MOSIM 01, 3ème Conférence Francophone sur la Modélisation et la Simulation, Troyes, France, 25-27 Avril, 2001.*

[Letouzey 01] Letouzey, A. (2001), Ordonnancement interactif basé sur des indicateurs : Applications à la gestion de commandes incertaines et à l'affectation des opérateurs, *thèse présentée pour obtenir le titre de Docteur de l'Institut National Polytechnique de toulouse.*

[Li 05] Li, X., and Olafsson, S. (2005). Discovering Dispatching Rules Using Data Mining, *Journal of Scheduling, 8, 515-527, 2005.*

[Lian 06] Lian, Z., Jiao, B., and Gu, X. (2008). A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan, *Applied Mathematics and Computation 183, 1008-1017.*

[Liaw 08] Liaw, C. (2008). An efficient simple metaheuristic for minimizing the makespan in two-machine no-wait job shops, *Computers & Operations Research, 35, 3276 -3283.*

[Liu 01] Liu, P.H., Makis, V. and Jardine, A.K.S. (2001). Scheduling of the optimal tool replacement times in a flexible manufacturing system, *IIE Transactions (2001) 33, 487-495.*

[Lopez 92] Lopez, P., Erschler, J. et Esquirol, P. (1992). Ordonnancement de tâches sous contraintes : une approche énergétique, *Revue d'Automatique, Productique, Informatique Industrielle, vol.26, n°5 6,453-481.*

[Maccarthy 93] Maccarthy, B.L et Lui, J. (1993). A new classification scheme for flexible manufacturing systems, *International Journal of Production Research, Vol. 31, No 2, 299-309.*

[Mahmoodi 99] Mahmoodi, F. and Mosier C.T. (1999). The Effects of Scheduling Rules and Routing Flexibility on the Performance of a Random Flexible Manufacturing System, *The International Journal of Flexible Manufacturing Systems, 11 (1999), 271–289.*

[Mamalis 95] Mamalis, A. G., Malagardis, I. and Pachos, E. (1995). On-line scheduling in metal removel processingusing variable routeing and control strategies, *Computer Integrated Manufacturing Systems, 8, 35-40.*

[Mebarki 95] Mebarki, N. (1995) .Une approche d'ordonnancement temps réel basée sur les règles de priorité des files d'attente, *PhD thèse, Université de Claude Bernard Lyon1.*

[Metropolis 53] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953).Equations of state calculations by fast computing machines, *Journal of Chemical Physics, 21:1087-1091.*

[Montgomery 06] Montgomery, J., Fayad, C. and Petrovic, S. (2006). Solution Representation for Job Shop Scheduling Problems in Ant Colony Optimisation, *Book chapter in: Ant Colony Optimization and Swarm Intelligence, Book series: Lecture Notes in Computer Science, 4150, 484-491.*

[Mori 90] Mori, K., Tsukiyama, M. and Fukuda, T. (1990). Cooperative scheduling environment based on editing and simulation for manufacturing systems , *Man and Cybernetics, Conference Proceedings, IEEE International Conference on Volume, Issue, 4-7, Page(s):557 – 559.*

[Niu 08] Niu, Q., Jiao, B. and Gu, X. (2008). Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time, *Applied Mathematics and Computation 205, 148-158.*

[Nof 79] Nof, S, Barash, M., and Solberg, J., (1979). Operational control of item flow in versatile manufacturing system, *International journal of production research, 17,479-489.*

[O'kane 00] O'kane, J. F. (2000). A knowledge-based system for reactive scheduling decision-making in FMS, *Journal of Intelligent Manufacturing, 11(5), 461-474.*

[Ombuki 04] Ombuki, B. M. and Ventresca, M. (2004). Local Search Genetic Algorithms for the Job Shop Scheduling Problem, *Applied Intelligence 21, 99-109.*

[Ozmutlu 05] Ozmutlu, S. and Harmonosky, C.M. (2005). A real-time methodology for minimizing mean flowtime in FMSs with routing flexibility: Threshold-based alternate routing, *European Journal of Operational Research 166 (2005), 369-384.*

[Pan 05] Pan, Q.K., Tasgetiren, M.F. and Liang Y.C. (2005), A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem with makespan criterion, *Proceedings of the international workshop on UK planning and scheduling special interest group, UK PLANSIG2005. City University, London, pp: 31-41.*

[Park 06] Park, B.J and Choi, H.R (2006). A Genetic Algorithm for Integration of Process Planning and Scheduling in a Job Shop, *Book chapter in: AI 2006: Advances in Artificial Intelligence, Book series: Lecture Notes in Computer Science, 4304, 647-657.*

[Patkai 98] Patkai, B. (1998). A cooperative scheduling system for paper machinery manufacturing.Industrial Electronics Society. *IECON apos, Proceedings of the 24th Annual Conference of the IEEE, 4(31), 2518 - 252.*

[Peng 98] Peng, C. and Chen, F.F. (1998). Real-time control and scheduling of flexible manufacturing systems: a simulation based ordinal optimization approach, *International Journal of Advanced Manufacturing Technology, 14 (10), 775-786.*

[Ponnambalam 00] Ponnambalam1, S. G., Aravindan1, P. and Rajesh, S. V. (2000). A Tabu Search Algorithm for Job Shop Scheduling, *International Journal of Advanced Manufacturing Technology, (2000) 16, 765-771.*

[Portmann 87] Portmann, M.C. (1987). Méthodes de décomposition spatiale et temporelle en ordonnancement de la production, *Thèse de doctorat d'Etat présentée à l'Université de Nancy I.*

[Rachamadugu 94] Rachamadugu, R. and Stecke, K. E. (1994). Classification and review of FMS scheduling procedures, *Production Planning and Control, 5, 2-20.*

[Reddy 06] Reddy, B. S. P. and Rao, C. S. P. (2006). A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS, *International Journal of Advanced Manufacturing Technology, 31, 602-613.*

[Restrepo 08] Restrepo, I.M., and Balakrishnan, S. (2008), Fuzzy-based methodology for multi-objective scheduling in a robot-centered flexible manufacturing cell, *Journal of Intelligent Manufacturing, (2008) 19,421-432.*

[Riane 99] Riane, F., De Brauwer, C. et Artiba, A. (1999). OCEHO : un outil de comparaison et d'évaluation d'heuristiques de résolution de problèmes d'ordonnancement, *Journal Européen de Systèmes Automatisés,32, 7 8, 853-874.*

[Ripon 07] Ripon, K. S. N., Tsang, C., and Kwong, S., (2007). An Evolutionary Approach for Solving the Multi-Objective Job-Shop Scheduling Problem, *Book chapter in: Evolutionary Scheduling, Book series: Studies in Computational Intelligence, 49, 165-195.*

[Romanowicz 97] Romanowicz, R., Jacot, J. Hertz, A. et Verdebout, E. (1997). An expert system prototype for the selection of scheduling method, *ETFA'97, 6th International Conference on Emerging Technologies and Factory Automation, Los Angeles, USA, 9-12 Septembre, 1997.*

[Rossi 07] Rossi, A. and Dini, G. (2007). Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method, *Robotics and Computer-Integrated Manufacturing, 23 (2007), 503-516.*

[Rossi 09] Rossi, A. and Boschi, E. (2009). A hybrid heuristic to solve the parallel machines job-shop scheduling problem, *Advances in Engineering Software, 40, 118-127.*

[Roubellat 94] Roubellat, F., Billaut, J.C., et Villaumie, M. (1994). Ordonnancement d'atelier en temps réel : d'ORABAID à ORDO, *Actes des Journées d'Etude,Ordonnancement et Entreprise Applications concrètes pour le futur, organisées par le Groupement de Recherche Automatique du C.N.R.S., Toulouse, 16-17 Juin, 213-253.*

[Sabuncuoglu 03] Sabuncuoglu, I. and Lahmar, M. (2003). An Evaluative Study of Operation Grouping Policies in an FMS, *The International Journal of Flexible Manufacturing Systems, 15, 217-239.*

[Sabuncuoglu 98] Sabuncuoglu, I. (1998). Scheduling with neural networks: a review of literature and new research directions, *Production Planning and Control, vol.9, n°1, pp. 2-12.*

[Saidi-Mehrabad 07] Saidi-Mehrabad, M. and Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms, International Journal of Advanced Manufacturing Technology, 32, 563-570.

[Sammarra 07] Sammarra, M., Cordeau, J., Laporte, G., Monaco, M.F., (2007). A tabu search heuristic for the quay crane scheduling problem, *J Sched (2007) 10, 327-336.*

[Sari 03] Sari, Z. (2003). Modélisation, analyse et évaluation des performances d'un AS/RS à convoyeur gravitationnel, *thèse de doctorat d'état, Université Abou Bekr Belkaid Tlemcen.*

[Saygin 01] Saygin.C., Chen, F.F. and Singh, J. (2001). Real-Time Manipulation of alternative Routings in Flexible Manufacturing Systems: A simulation Study, *International journal of advanced Manufacturing Technology, 18, 755-763.*

[Saygin 04] Saygin, C. and Kilic, S.E. (2004). Dissimilarity Maximization Method for Real-time Routing of Parts in Random Flexible Manufacturing Systems, *The International Journal of Flexible Manufacturing Systems, 16, 169-182.*

[Saygin 95] C. Saygin, S. E. Kilick, T. Toth and F.Erdelyi. (1995). *On scheduling approaches of flexible manufacturing systems: gap between theory and practice, Selected paper – Postprint volume of the 3rd IFAC/IFIP/IFORS Workshop – Intelligent Manufacturing Systems 95, Pergamon/Elsevier Science,61-66.*

[Saygin 99] Saygin, C., Kilic, S.E. (1999). Integrating flexible manufacturing systems with scheduling in flexible manufacturing system, *International journal of advanced Manufacturing Technology*, 15(4), 268-280.

[Schuster 06] Schuster, C.J. (2006). No-wait job shop scheduling: tabu search and complexity of subproblems, *Math. Meth. Oper. Res, (2006) 63, 473-491.*

[Schwalb 97] Schwalb, E. et Dechter, R. (1997). Processing disjunctions in temporal constraint networks, *Artificial Intelligence, vol., n°93, 1997, pp. 29-61.*

[Scrich 04] Scrich, C.R., Armentano, V.A., and Laguna, M. (2004). Tardiness minimization in a flexible job shop: A tabu search approach, *Journal of Intelligent Manufacturing, 15 (1), 103-115.*

[Sha 06] Sha, D.Y. and Hsu, C. (2006). A hybrid particle swarm optimization for job shop scheduling problem, *Computers & Industrial Engineering 51, 791-808.*

[Simon 77] H.A. Simon, (1977). The new science of management decision, *Prentice-Hall, 1977.*

[Smed 00] Smed, J., Johtela, T., Johnsson, M., Puranen, M. et Nevalainen, O. (2000). An interactive system for scheduling jobs in electronic assembly, *International journal of advanced Manufacturing Technology, 16,450-459.*

[Smith 92] Smith, S. F. (1992). Knowledge-Based production management: approaches, results and prospects, *Production Planning and Control, vol.3, n°4, 350-380.*

[Souier 08] Souier, M., Hassam, A. and Sari, Z. (2008). Evaluation of Metaheuristics performances in manipulation of alternative routing, *International conference on Electrical Engineering Design and Technologies, Hammamet Tunisia.*

[Spalanzani 99] Spalanzani, A., (1999). Algorithmes évolutionnaires pour l'étude de la robustesse des systèmes de reconnaissance automatique de la parole, *Thèse de doctorat de l'Université Joseph Fourier - Grenoble I.*

[Stecke 92] Stecke, K.E. (1992). Procedures to determine part mix ratios for independent demands in flexible manufacturing systems. *IEEE Transactions on Engineering Management 39(4), 359-369.*

[Stutzle 00] Stutzle, T. and Hoos, H. (2000). MAX-MIN Ant System, *Future Generation Computer System, 16:889-914.*

[Stutzle 97] Stutzle, T. and Hoos, H. (1997). Improvements on the Ant System: Introducing MAX-MIN Ant System, *In Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms, Vienna. Springer-Verlag.*

[Talavage 88] Talavage, J.T., et Hannam, R.G. (1988). Flexible Manufacturing Systems in Practise, *Marcel Dekker, New-York.*

[Tang 93] Tang, L.L., Yih, Y. and Liu, C.Y. (1993). A study on decision rules of a scheduling model in an FMS, *Computers in Industry 22 (1993) 1-13.*

[Tao 07] Tao, Z. and Xiao, T. (2007). Petri Net and GASA Based Approach for Dynamic JSP, *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation August 5 - 8, 2007, Harbin, China.*

[Thomas 80] Thomas, V. (1980). Aide à la décision pour l'ordonnancement en temps réel d'atelier, *Thèse de troisième cycle présentée à l'Université Paul Sabatier de Toulouse.*

[Trappey 07] Trappey, A.J. C., Lin, G.Y. P., Ku, C. C. and Ho, P.-S. (2007). Design and analysis of a rule-based knowledge system supporting intelligent dispatching and its application in the TFT-LCD industry, *International Journal of Advanced Manufacturing Technology, (2007) 35,385-393.*

[Tsai 08] Tsai, J., Liu, T., Ho, W., and Chou, J. (2008). An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover, *International Journal of Advanced Manufacturing Technology (2008), 38,987-994.*

[Tsubone 99] Tsubone,H. and Horikawa, M. (1999). A Comparison Between Machine Flexibility and Routing Flexibility, *The International Journal of Flexible Manufacturing Systems, 11 , 83-101.*

[Tunali 97] Tunali, S. (1997). Evaluation of alternate routing policies in scheduling a job-shop type FMS, *Computers ind. Engng* 32(2), 243-250.

[Vilcot 08] Vilcot, G. and Billaut, J. (2008). A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem, *European Journal of Operational Research, 190, 398-411.*

[Wadhwa 06] Wadhwa, S., Rao, K.S., Chan, F.T.S. (2006). Comparative influence of three flexibility types on manufacturing lead-time performance, *International Journal of Advanced Manufacturing Technology 29, 1002-1007.*

[Watson 05] Watson, J., Whitley, L. D. and Howe, A. E. (2005). A dynamic model of tabu search for the job-shop scheduling problem*, Book chapter in: Multidisciplinary Scheduling: Theory and Applications, 2005, 247-266.*

[Widmer 91] Widmer, M. (1991). Modèles mathématiques pour une gestion efficace des ateliers flexibles, *Collection META, Lavoisier TEC DOC.*

[Wu 89] Wu, S. Y. D. and Wysk, R. A. (1989). An application of discrete event simulation to on-line control and scheduling in flexible manufacturing, *International journal of Production Research, 27, pp. 1603-1623.*

[Xia 06] Xia, W. and Wu, Z. (2006). A hybrid particle swarm optimization approach for the job-shop scheduling problem, *International Journal of Advanced Manufacturing Technology (2006) 29, 360-366.*

[Zattar 07] Zattar, I. C., Ferreira, J.C.E., Granado. J.G.G., and Sousa, C.H.B.D. (2007). Integrating Manufacturing Process Planning with Scheduling via Operation-Based Time-Extended Negotiation Protocols, *Book chapter in Complex Systems Concurrent Engineering, (2007) 6,329-336.*

[Zhang 07] Zhang, C., Li, P., Guan, Z. and Rao, Y. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem, *Computers & Operations Research, 34, 3229 - 3242.*

[Zhang 08a] Zhang, C., Rao, Y., and Li, P. (2008). An effective hybrid genetic algorithm for the job shop scheduling problem, *International Journal of Advanced Manufacturing Technology (2008), 39,965-974.*

[Zhang 08b] Zhang., R., and Wu, C., (2008). An Immune Genetic Algorithm Based on Bottleneck Jobs for the Job Shop Scheduling Problem, *Book chapter in: Evolutionary Computation in*

Combinatorial Optimization, Book series: Lecture Notes in Computer Science, Volume 4972/2008, pages: 147-157.

[Zhao 01] Zhao, C. and Wu, Z. (2001). A Genetic Algorithm Approach to the Scheduling of FMSs with Multiple Routes, *The International Journal of Flexible Manufacturing Systems, 13, 71-88.*

[Zouhri 93] Zouhri, M., (1993). Représentation analytique de familles d'ordonnancement : apport des arbres PQR, *Thèse présentée à l'INSA de Lyon*.

Résumé :

Les ateliers flexibles avec des ressources limités, des pannes de machines aléatoires ou des critères de production multiples ont un aspect qui explique que les problèmes d'ordonnancement dans ces systèmes sont généralement de type NP-difficile. C'est pour cela que plusieurs heuristiques et métaheuristiques ont été proposées pour les résoudre.

Dans ce travail, nous avons proposé des algorithmes à base de certaines métaheuristiques (les colonies de fourmis, les algorithmes génétiques, le recuit simulé, la recherche tabou, les essaims particulaires et l'électromagnétisme) pour la sélection de routages alternatifs en temps réel et comparé leurs performances avec les règles DMM (Dissimilarity Maximization Method) et DMM modifiée afin d'avoir une idée sur l'efficacité de ces métaheuristiques et de choisir la plus efficace.

Pour valider les résultats de cette étude, nous avons simulé les règles DMM et DMM modifiée sur un Job shop en utilisant le logiciel de simulation ARENA, et les métaheuristiques ont été simulées par Java sur le même modèle.

Les résultats obtenus ont montré que la plupart des métaheuristiques ont nettement amélioré le taux de production, le taux d'utilisation des différentes machines et le taux d'utilisation du système de transport, pour un système de production saturé et même en présence de pannes.

Abstract:

Flexible manufacturing systems with limited resources, random breakdowns of machines or multiple criteria of production have an aspect which explains that the problems of scheduling in these systems are generally NP-hard. Because of this, several heuristic and metaheuristic were proposed to solve them.

In this work, we have proposed some algorithms based on some metaheuristics (Ant colony, genetic algorithms, simulated annealing, taboo search, particle swarm and electromagnetism) for alternative routing selection in real time and compared theirs performances with DMM (Dissimilarity Maximization Method) and modified DMM rules in order to have an idea on the effectiveness of these metaheuristics and select the most effective.

To validate the results of this study, we simulated DMM and modified DMM rules on a Job shop model using ARENA and metaheuristics were simulated using Java on the same model.

The obtained results showed that all metaheuristics clearly improved the production rate, the utilization of the various machines and the utilization of the material handling system, for overloaded production systems and even in the presence of breakdowns.

ملخص:

حلقات العمل المرنة ذات موارد محدودة، آلات إنتاج تتعطل عشوائيا أو معايير متعددة الجوانب لها مظهر يفسر أن مشاكل الجدول الزمني في هذه النظم عادة من النوع التي أرستها بشدة لهدا العديد من الاستدلالات وفوقية الاستدلالات قد اقترحت لحلها.

في هذا العمل ، اقترحنا خوارزميات استنادا إلى بعض فوقية الاستدلالات (مستعمرات النمل ، الخوارزميات الجينية ، محاكاة الصلب ، تابو البحث ، أسراب من الجسيمات والكهرومغناطيسية) لاختيار مسارات بديلة في الوقت الحقيقي ومقارنة أدائها مع القواعد طريقة أقصى التباين و أقصى التباين المعدلة،لأخد فكرة عن فعالية هذه التقنيات واختيار الأكثر فعالية

للمصادقة على نتائج هذه الدراسة، فإننا حاكينا القواعد في ورشة عمل على مسارات متعددة باستخدام برامج المحاكاة ARENA و الفوقية الاستدلال باستخدام JAVA في نفس الورشة

وأظهرت النتائج التي تم التوصل إلى أن معظم فوقية الاستدلالات حسنت معدل الإنتاج ، معدل استخدام الآلات المختلفة والاستفادة من شبكة النقل لأنظمة إنتاج مشبعة وحتى في وجود حالات العطب

Mots clés : Atelier à cheminement multiples, Routage alternatif, Règle de sélection de routage, Métaheuristiques, Simulation.

Key words: Job shop, Alternative routing, Routing selection rule, Metaheuristics, Simulation.

الكلمات المفتاحية :

ورشة عمل دات العديد من المسارات،تحديد مسارات بديلة، قاعدة تحديد المسارات، فوقية الاستدلالات ، المحاكاة.