REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DE MATHEMATIQUES ET D'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'études

Présenté pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Spécialité : Informatique Décisionnel et Optimisation

Par : DAHMANE Lamia et HADROUG Selma

SUJET

Algorithme génétique parallèle pour un problème d'ordonnancement d'atelier Job Shop

Soutenu publiquement le : / /2020devant le jury composé de :

Dr. AKROUF Samir	**UMB M'sila**	**Président**
Dr. HEMMAK Allaoua	**UMB M'sila**	**Rapporteur**
Dr. MOUHOUB Nassereddine	**UMB M'sila**	**Examinateur**

Promotion : 2019 /2020

Dédicaces

A nos très chers parents.

Remerciements

On remercie Dieu le tout puissant, qui nous a donné la force et la patience pour l'accomplissement de ce travail.

*Nous remercions en particulierMr.**AllaouaHemmk**, pour l'honneur qu'il nous a fait de bien vouloir nous encadrer, et pour les conseils donnés lors de la réalisation de ce travail.*

*Je remercie beaucoup mon collègue **Dahman Lamia**pour ses efforts et son merveilleux comportement avec moi*

On adresse nos remerciements au membre de jury pour avoir accepté de nous prêté de leur attention et évaluer notre travail.

TABLE DES MATIERES :

Liste des Figures

Liste des Tables

Liste des Algorithmes

Introduction générale

La recherche opérationnelle (aide à la décision)est l'ensemble des méthodes et techniques rationnelles traite un problème pratique ,elle a un objectif limité en effet elle nécessite une boîte à outils (algorithmes et structures des données ,optimisation combinatoire, graphes, complexité, programmations linéaire et mathématique, processus stochastiques, probabilités et statistiques, méthodes multicritères....),aussi la recherche opérationnelle est pluridisciplinaire (Mathématique, Informatique, Economie); avec elle est banalisée (Programmation linéaire, PERT, ...)

Le problème d'Ordonnancement est une branche parmi les branches de recherche opérationnelle qui consiste à affecter des tâches à des ressources à des instants donnés, tel que Ces tâches sont soumises à certaines restrictions. Ordonnancement possède plusieurs modèle parmi les : modèle de job shop, tell que il contient n jobs vont traiter en m machines. Chaque machine traiter un seul job pour un moment de temps. Le temps qui on besoin pour finir le traitement de tous les Jobes en tous les machines est Makespane C_max.

Maintenant nous cherchons comment optimiser le Makespane ; il y a plusieurs des méthodes d'optimisation nous choisissons les algorithmes génétiques qui ils sont dérivé a la nature et ils sont le plus utilises dans de multiples domaines.en effet Les algorithmes génétiques sont capables de trouver des solutions qui ne sont peut-être pas des solutions précises, mais qui sont rapides et efficaces Nous allons également essayer d'utiliser des algorithmes génétiques parallèles afin d'étudier les différences et l'étendue de l'effet du parallélisme dans l'accélération de la recherche de solutions.

Ainsi, notre mémoire sera articulé sur 5 chapitres :

Dans le premier chapitre présent d'une façon générale l'ordonnancement, nous présentons le problème d'ordonnancement généralement, puis nous expliquons la formulation de ce problème, ensuite nous allons représenter dans la quatrième et cinquième section classification et représentation de problème d'ordonnancement.Enfin dans la dernière section on va clôture se chapitre en expliquant les différentes méthodes de résolution du problème d'ordonnancement.

Dans le deuxième chapitre nous avons continué en la présentation générale mais maintenait les algorithmes génétiques, et voila nous leurs principes (Le principe de variation**,** Le principe d'adaptation**,** Le principe d'hérédité), après nous sommes passés par les opérateurs des algorithmes génétiques (sélection, croisement, mutation), Ensuite, nous avons examiné les

algorithmes génétiques parallèles et certains de leurs modèles, et encore leurs mécanismes de fonctionnement.

Dans le troisième chapitre est l'état de l'art nous présentons quoi les autres faire dans notre thème et présenter leur travaille par un petite résume.

Dans le quatrième chapitre nous présentons comment ressouder le problème de Job Shop avec l'utilisation des algorithmes génétiques parallèle.

Enfin dans le dernier chapitre nous présentons notre application et expliquons comment nous le faire.

CHAPITRE 1 ORDONNANCEMEN

1. Introduction

Dans ce chapitre, nous présentons le problème d'ordonnancement généralement, puis nous expliquons la formulation de ce problème, ensuite nous allons représenter dans la quatrième et cinquièmesection classification et représentation de problème d'ordonnancement.
Enfin dans la dernière section on va clôture se chapitre en expliquant les différentes méthodes de résolution du problème d'ordonnancement.

2. Généralités sur l'ordonnancement

Ordonnancer, c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution. Il est considéré comme une branche de la recherche opérationnelle et de la gestion de la production qui vise à améliorer l'efficacité des entreprises en termes de coûts de productionDélais de livraison.

Les problèmes d'ordonnancement, apparaissent dans tous les domaines de l'économie: l'informatique, la construction (suivi de projet), l'industrie (problèmes d'ateliers, gestion de production), l'administration (emploi du temps). Les tâches sont le dénominateur commun des problèmes d'ordonnancement, leur définition n'est ni toujours immédiate, ni toujours triviale. Pour ça, il faut programmer les tâches de façon à optimiser un certain objectif qui sera, suivant le cas ,la minimisation de la durée totale (c'est le critère le plus fréquemment employé) ou le respect des dates de commande ou de lissage des courbes de main d'œuvre ou encore la minimisation d'un coût. D'une manière générale, trois types d'objectifs sont essentiels dans la résolution des problèmes d'ordonnancement : l'utilisation efficace des ressources, un délai d'exécution des tâches aussi faible que possible et le respect des dates d'achèvement prescrites à l'avance [10].

Les différentes données d'un problème d'ordonnancement sont les tâches, les contraintes potentielles, les ressources et la fonction économique.

Un problème d'ordonnancement consiste à affecter des tâches à des ressources à des instants donnés, tel que Ces tâches sont soumises à certaines restrictions.

3. Formulation d'un problème d'ordonnancement

3.1 Les tâcher

Une tâche est un travail (ou job) dont la réalisation nécessite un nombre d'opérations élémentaires. Chaque opération élémentaire nécessite un certain nombre d'unités de temps (sa durée) et d'unités de ressources [12].

On distingue deux types de tâches [9]:

- **Les tâches morcelables (préemptives)** qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.
- **Les tâches non morcelables (indivisibles)** qui doivent être exécutées en une seule fois et ne sont interrompues qu'une fois terminées.

3.2 Les ressource

Une ressource est un moyen technique ou humain utilisé pour réaliser une tâche. On trouve plusieurs types de ressources [9]:

- **Les ressources renouvelables**, qui, après avoir été allouées à une tâche, redeviennent disponibles (machines, personnel, etc.).
- **Les ressources consommables,** qui, après avoir été allouées à une tâche, ne sont plus disponibles (argent, matières premières, etc.).

Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours du temps. Par ailleurs, dans le cas des ressources renouvelables, on distingue principalement, les ressources disjonctives qui ne peuvent exécuter qu'une tâche à la fois et les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité[9].

3.3 Les contraintes

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement les variables représentant les relations reliant les tâches et les ressources. On distingue les contraintes temporelles et les contraintes de ressources [12].

Les contraintes temporelles intègrent:

- **les contraintes de temps alloué**, issues généralement d'impératifs de gestion et relatives aux dates limites des tâches (délais de livraison, disponibilité des approvisionnements) ou à la durée totale d'un projet.

- **les contraintes d'antériorité** et plus généralement les contraintes de cohérence technologique, qui décrivent le positionnement relatif de certaines tâches par rapport à d'autres.
- **les contraintes de calendrier liées** au respect d'horaires de travail, etc.
- **Les contraintes de ressources** traduisent le fait que les ressources sont disponibles en quantité limitée. On distingue deux types de contraintes de ressources, liées à la nature disjonctive ou cumulative des ressources. Une ressource disjonctive ne peut être utilisée que par une tâche à la fois. Par contre dans une ressource cumulative les ensembles de tâches non réalisables simultanément sont de cardinalité quelconque [12].

3.4 Les critères

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi.

Les critères dépendant d'une application donnée sont très nombreux; plusieurs critères peuvent être retenus pour une même application. Le choix de la solution la plus satisfaisante dépend du ou des critères préalablement définis, pouvant être classés suivant deux types,
réguliers et irréguliers.

Les différents critères ne sont pas indépendants; certains même sont équivalents. Deux critères sont équivalents si une solution optimale pour l'un est aussi optimale pour l'autre et inversement [10] :

- **Les critères réguliers** constituent des fonctions décroissantes des dates d'achèvement des opérations. Quelques exemples sont cités ci-dessous:
    - la minimisation des dates d'achèvement des actions.
    - la minimisation du maximum des dates d'achèvement des actions.
    - la minimisation de la moyenne des dates d'achèvement des actions.
    - la minimisation des retards sur les dates d'achèvement des actions.
    - la minimisation du maximum des retards sur les dates d'achèvement des actions.
- **Les critères irréguliers** sont des critères non réguliers, c'est-à-dire qui ne sont pas des fonctions monotones des dates de fin d'exécution des opérations, tels que:
    - la minimisation des encours.
    - la minimisation du coût de stockage des matières premières.
    - l'équilibrage des charges des machines.
    - l'optimisation des changements d'outils.

La satisfaction de tous les critères à la fois est souvent délicate, car elle conduit souvent à des situations contradictoires [4] et à la recherche de solutions à des problèmes complexes d'optimisation [9].

4. Classification des problèmes d'ordonnancement

4.1 Modèles à une opération

Il est représenté dans un Modèle à machine unique et un Modèle à machines parallèle.

4.1.1 Modèle à machine unique

Pour un modèle à machine unique, une seule machine exécute toutes les tâches à effectuer, Ce modèle est illustré dans la figure 1.1 :

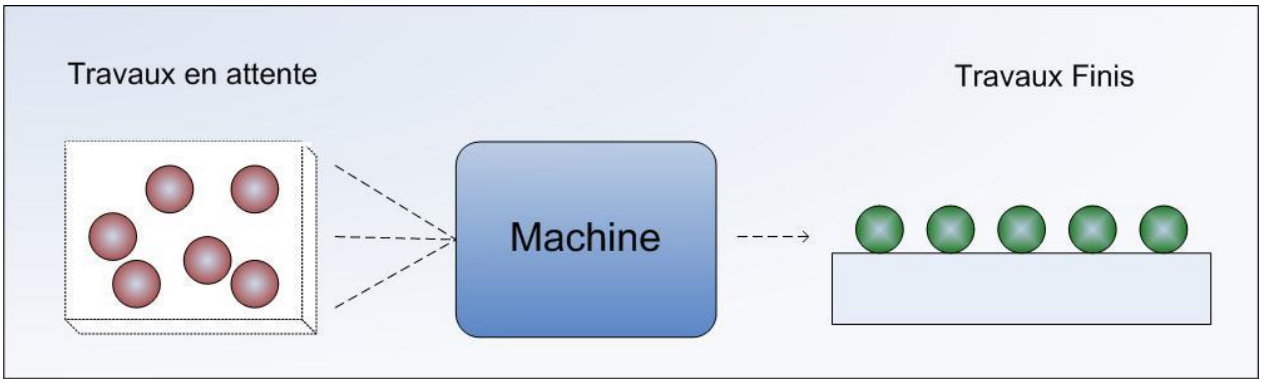

Figure 1.1 Ordonnancement sur machine unique [a].

4.1.2 Modèle à machines parallèle

Quant au modèle à machine parallèle, Lorsque la machine i est libérée, le job lui est attribué, comme illustré à la Figure 1.2, Pour que ce modèle soit essentiel pour le secteur industriel, en particulier l'industrie textile. Alors que les machines parallèles sont classées suivant leur rapidité [12] :

- machines identiques (P) : la vitesse d'exécution est la même pour toutes les machines M_j et pour tous les travaux J_i [12].

- machines uniformes (Q) : chaque machine M_j a une vitesse d'exécution propre et constante. La vitesse d'exécution est la même pour tous les travaux J_i d'une même machine M_j [12].
- machines indépendantes (R) : la vitesse d'exécution est différente pour chaque machine $M_j$et pour chaque travail J_i.

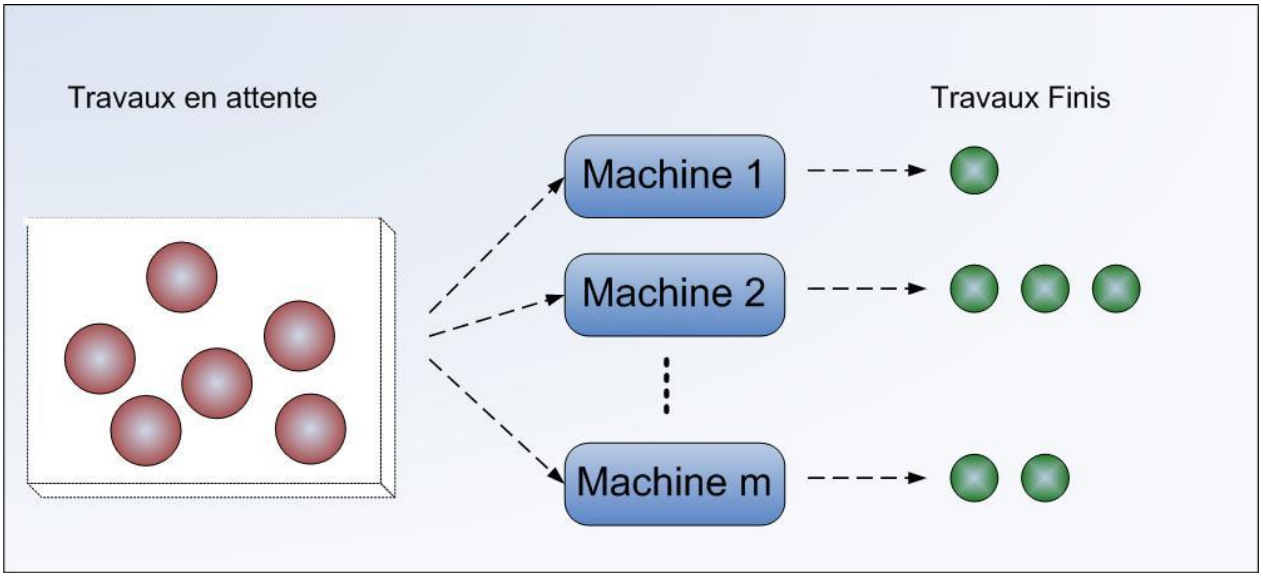

Figure 2.2 Modèle à machine parallèle [a].

4.2 Modèles à plusieurs opérations

Le modèle à plusieurs opérations est constitué des cas où un job, pour se réaliser, doit passer par plusieurs machines, chacune de ces machines ayant ses spécificités. On distingue trois modèles selon l'ordre de passage des jobs sur les machines, à savoir les modèles de flow-shop, job-shop et open-shop.

4.2.1 Modèle flow-shop

Le modèle de flow-shop, Appelés également modèle linéaire et aussi appelé ateliers à cheminement unique, Parce que toutes les jobs passent par les machines dans le même ordre, Comme le montre la figure, nous avons quatre machines et quatre jobs. Où les jobs suivent le même ordre de traitement sur les machines.

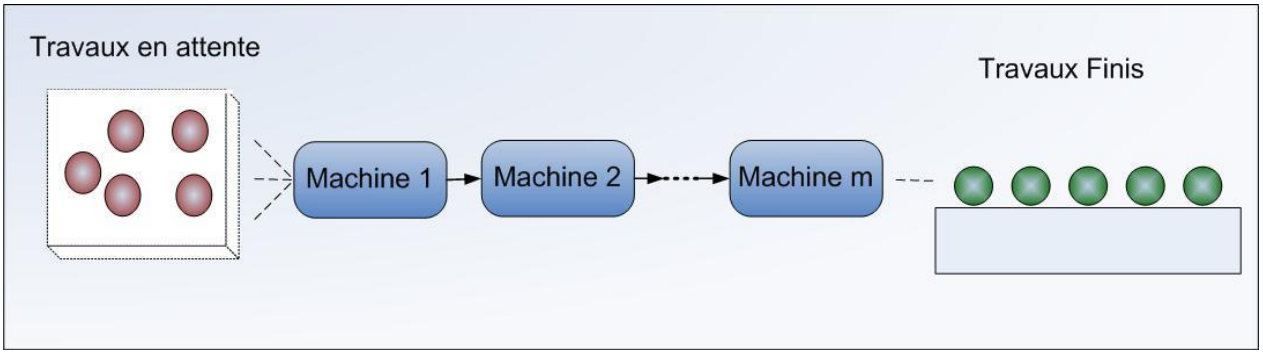

Figure 3.3 Ateliers à cheminement unique (Flow-shop) [a].

4.2.2 Modèle job-shop

Concernant le modèle de job-shop Appelés également ateliers à cheminement multiple. Ce modèle consiste à affecter un J de r jobsJ_1,J_2 ,J_3,....J_r et M un ensemble de m machines M_1 , M_2, ... , M_m. Un ensemble O d'opérations doit être ordonnancé. Chaque job $J_i$est composé d'un ensemble de k opérations, notéesO_{i1},O_{i2},O_{i3},...O_{ik} tel que cette opérations sont réalisées selon un ordre bien déterminé, et Une opération ne peut appartenir qu'à un seul job. Chaque opération O est affectée à une machine M_j. Le temps de traitement d'une opération O_{ij} est notéP_{ij}. Il ya un exemple de modèle de Job-shop dans le Tableau 1 ci-dessous, composé de deux jobs. Chaque job comporte trois opérations. Chaque opération est à réaliser sur une machine donnée. Il y a trois machines dans le problème considéré. Les temps de traitement varient d'une opération à une autre [5].

Job 1	$O_1 = (M_3 ; 4)$	$O_2=(M_1 ; 3)$	$O_3 = (M_2 ; 2)$
Job 2	$O_4 = (M_2 ; 7)$	$O_5 = (M_1 ; 6)$	$O_6 = (M_3 ; 5)$

Table 1.1 La table de le modèle de job-shop.

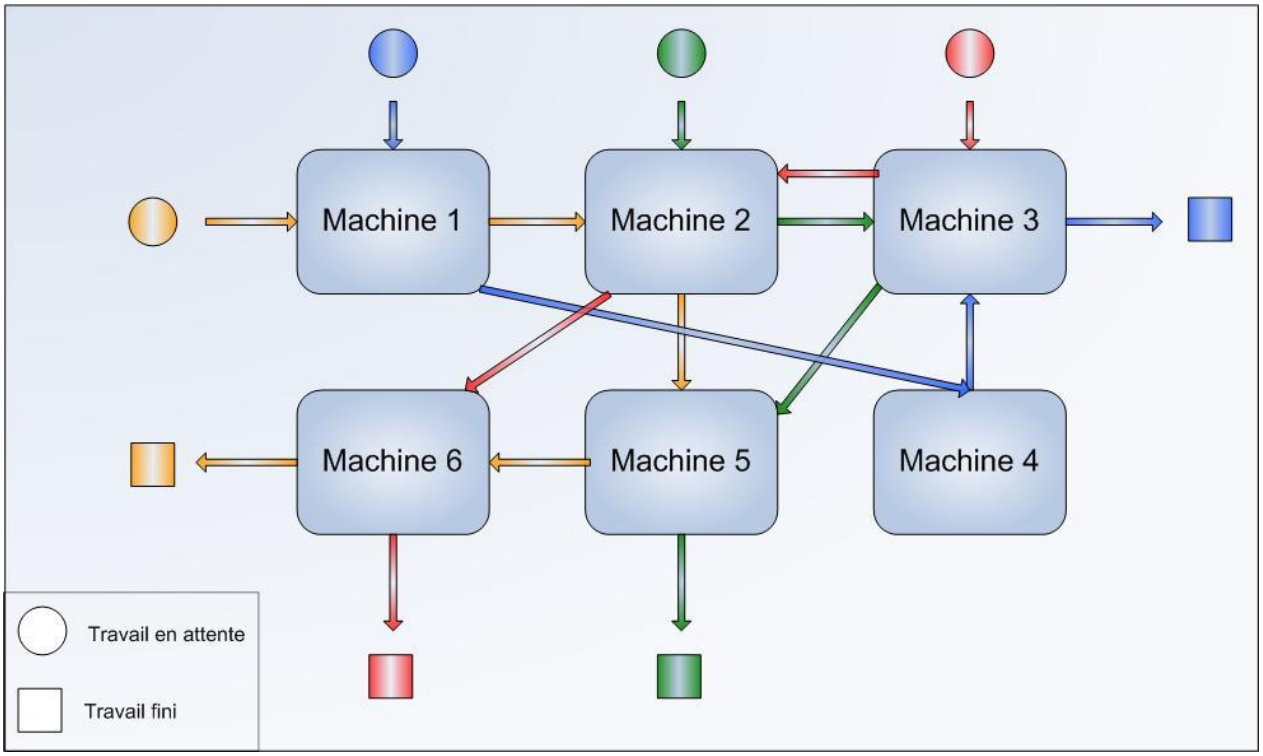

Figure 4.4 Ateliers à cheminement multiple (Job-hop) [a].

4.2.2.1 Formalisation linéaire

Plusieurs formulations linéaires existent pour le job-shop et certaines d'entre elle se basent sur la formulation de Manne. Dans Pham (2008), une évaluation des formulations linéaires du job-shop est proposée.

Pour un problème de Job-Shop de n jobs et m machines, on considère les notations suivantes :

$\tilde{J}$ l'ensemble de tous les jobs;$\tilde{J}$ ={1; 2; … ; n} ;

I l'ensemble de toutes les opérations ;

M l'ensemble de machines M={1; 2; … ; m} ;

A_jl'ensemble de tous les couples d'opérations consécutives pour le job j ∈ $\tilde{J}$;

B l'ensemble de tous les couples d'opérations $(i, j) \in I, i \neq j$ exécutées sur la même machine ;

I_kl'ensemble des opérations exécutées sur la machine $k \in M$;

P_i Le temps opératoire de l'opération $i \in I$;

H un nombre entier positif suffisamment grand ;

x_i La date de début de l'opération $i \in I$;

x_τ La date de début de l'opération * ;

Pour chaque couple d'opérations $\forall(i, j) \in I$ s'exécutant sur la même machine $k \in M$

$$y_{ij}\begin{cases}1 \text{ si l'opération } i \text{ éxécutée avant l'opération j.}\\ 0 \text{ sinon}\end{cases}$$

4.2.2.2 Formalisation mathématique

$x_j - x_i \geq P_i \forall (i,j) \in A_k, \forall k \in \widetilde{J}\ (1)$

$x_j + H(1 - y_{ij}) - x_i \geq P_i \forall (i,j) \in B (2)$

$x_i + H_{y_{ij}} - x_j \geq P_i \forall (i,j) \in B (3)$

$x_\tau - x_i - P_i \geq 0 \qquad \forall i \in I (4)$

$y_{ij} \in \{i,j\} \forall (i,j) \in B (5)$

$x_i \geq 0 \qquad \forall i \in I (6)$

$x_\tau \geq 0 \qquad (7)$

- **les contraintes (1)** assurent : aucune opération ne peut commencer avant la fin d'exécution de l'opération qui la précède.
- **les contraintes (2) et (3)** assurent: des contraintes de disjonction machine et assurent que deux opérations s'exécutant sur la même machine doivent être ordonnées grâce à l'utilisation de la variable binaire $y_{ij} \in \{0,1\}$de **la contrainte (5).**
- **Les contraintes (4)** : fixent la date de début de l'opération *,ceci est assuré par le fait que $x_\tau$est supérieur à toutes les dates de début plus le processing time x_i+P_i de chaque opération par $i \in I$.
- **Les contraintes (6) et (7)** : imposent que toutes les dates de débuts des opérations sont positives ou nulles [13].

4.2.3 Modèle open-shop

Modèle open-shop est un modèle d'atelier moins contraint que le flow shop et le job shop , tel que Chaque job j peut avoir son propre ordre de passage sur toutes les machines, Cela signifie que l'arrangement n'est pas connu à l'avance , Comme le montre la figure--. Bien qu'il y ait quelques difficultés à résoudre le problème d'ordonnancement en raison de l'absence d'arrangement préalable Cependant, ce modèle nous a permis simultanément de résoudre deux problèmes d'ordonnancement(déterminer le cheminement de chaque travail et ordonnancer les travaux)en tenant compte des gammes trouvées .

5. Représentation des problèmes d'ordonnancement

Il existe trois sortes de représentations possibles d'un problème d'ordonnancement: le diagramme de Gantt, le graphe Potentiel-Tâches et la méthode PERT.

5.1 Le diagramme de Gantt

Le diagramme de Gantt, C'est une excellente méthode car il est très facile de représenter la solution d'ordonnancement.

Il est aussi appelé le diagramme à barres, inventé cette méthode Henry Gantt (1861 – 1919) , Cette méthode est utilisée par de nombreux chefs de projet. représente La Figure 1.5 un ordonnancement de cinq jobs (J_i, J_2, J_3) sur 2 machines parallèles identiques,tel que l'axe des abscisses représente le temps etsur l'axe des ordonnées apparaissent les machines(M1,M2,M3),Sur chaque ligne horizontale, on met l'ordonnancement des tâches sur cette machine. Chaque tâche est représentée par une barre. La longueur de cette barre est proportionnelle à sa durée. A la fin, on obtient sur le digrammel'enchaînement de opérations sur chacune des machines, avec les dates de début et de fin de chaque tâche.

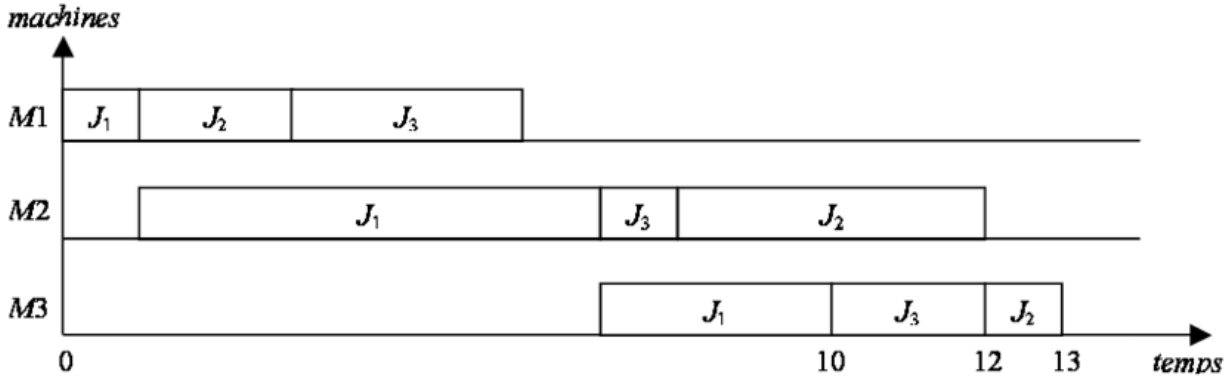

Figure 5.5 Exemple de Diagramme de Gant [c].

5 .2 Graphe Potentiel-Tâches

Cette outil graphique a été développé grâce à la théorie des réseaux de Pétri qui ont surtout servi à modéliser les systèmes dynamiques à évènements discrets [11].

Dans ce genre de modélisation, les tâches sont représentées par des nœuds et les contraintes par des arcs [3]. Ainsi, les arcs peuvent être de deux types :

- **Les arcs conjonctifs** illustrant les contraintes de précédence et indiquant les durées des tâches,
- **Les arcs disjonctifs** indiquant les contraintes de ressources [8], [1].

5.3 Méthode PERT (Program Evaluation and Research Task)

La méthode PERT est une technique permettant de gérer l'ordonnancement dans un projet. Il est représenté sous la forme d'un graphe d'un réseau de tâches plusieurs tâches qui grâce à leur dépendance et à leur chronologie concourent toutes à l'obtention d'un produit fini.

Tel que, la méthode PERT implique au préalable:

- Un découpage précis du projet en tâches.
- L'estimation du la durée de chaque tâche.
- La nomination d'un chef de projet chargé d'assurer le suivi du projet, de rendre comptesinécessaireet de prendre des décisions en cas d'écart par rapport aux prévisions.

6. Les méthodes de résolution du problème d'ordonnancement

6.1 Les méthodes exactes

On dit qu'une méthode exacte est une méthode utile, Lorsqu'il est utilisé pour résoudre des problèmes de petite taille. Autrement dit, quand il est le temps de calcul nécessaire pour atteindre la solution optimale n'est pas excessif [14]. ces méthodes examinent d'une manière implicite, la totalité de l'espace de recherche pour produire la solution optimale :

- La programmation dynamique
- la programmation linéaire
- Les méthodes arborescentes de type séparation/évaluation (Branch-and-Bound).

6.2 Les méthodes approximatives

Ces méthodes sont considérées pour les problèmes d'ordonnancement dans lesquels nous ne trouvons pas de solution optimale en un temps raisonnable [17] .Parmi ces méthodes, les heuristiques et les méta-heuristiques.

6.2.1 Les heuristiques

Dépend les heuristiques sur des méthodes empiriques, tel que elles se Construit sur des règles simplifiées pour optimiser un ou plusieurs critères. Le principe général de cette catégorie de méthodes est d'intégrer des stratégies de décision pour construire une solution proche de celle optimale tout en cherchant à avoir un temps de calcul raisonnable [7].

Exemple des heuristiques :

- **RANDOM :** l'opération est choisie aléatoirement parmi les opérations non encore ordonnancées.
- **SPRT** (ShortestRemainingProcessing Time): C'est l'opération dans lequel la durée opératoire est inférieure à celles des autres opérations.
- **LPT** (LongestProcessing Time) : C'est l'opération dans lequel la durée opératoire est supérieure à celles des autres opérations.

6.2.1 Méta-heuristiques

Une méta-heuristique est souvent définie comme une procédure exploitant au mieux lastructure du problème considéré,dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible [19].Les principales métaheuristiques sont celles basées sur les *méta-heuristiques à solution unique (* la recherche locale (RL), le recuit simulé (RS), la recherche avec tabous (RT)),et les *méta-heuristiques à population de solutions* (les algorithmes génétiques (AG) et l'optimisation par colonie de fourmis (OCF) ainsi que l'algorithme à évolution différentielle (*DE*)).

CHAPITRE 2LES ALGORITHMES GENETIQUES

1. Introduction

Dans ce chapitre on explique c'est quoi les algorithmes génétiques, et leur principes en plus comment ca marche, ensuit n'oublier pas les opérateurs des algorithmes génétiques puis en exploite les métaheuristique, en fin en représente le parallélisme et leurs modèles.

2. Définition

Dans les années 1970, des algorithmes génétiques ont été créés par John Holland, ce sont des algorithmes dérivées de la nature, leur paradigme est lié avec la génétique qui est la population, elle signifie un ensemble de solutions, et l'individu représente une solution, en plus le chromosome est un composent de la solution, enfin le gène qui est une caractéristique, il y a trois opérateurs des algorithmes génétiques : sélection, croisement et mutation.

3. Principes des algorithmes génétiques

Il y a 3 principes pour les algorithmes génétiques pour l'évolution des espèces a partir l'utilisation de la théorie de Darwin sont :

3.1 Principe de variation

Lorsque nous avons une population, chaque individu ou la population est unique, ce sont donc des déférences très importantes, ce qui aide à la sélection

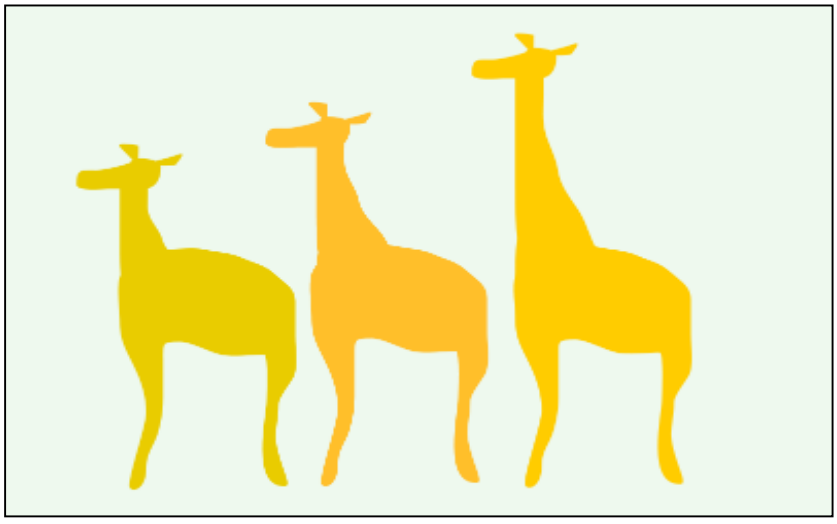

Figure 2.1 représentation du principe de variation [b].

3.2 Principe d'adaptation

Ce principe est basé sur la recherche d'individus capables d'atteignent plus facilement l'âge adulte en raison de leur capacité à s'adapter a leur environnement et cela signifie qu'ils ont la capacité de survivre et de se reproduire.

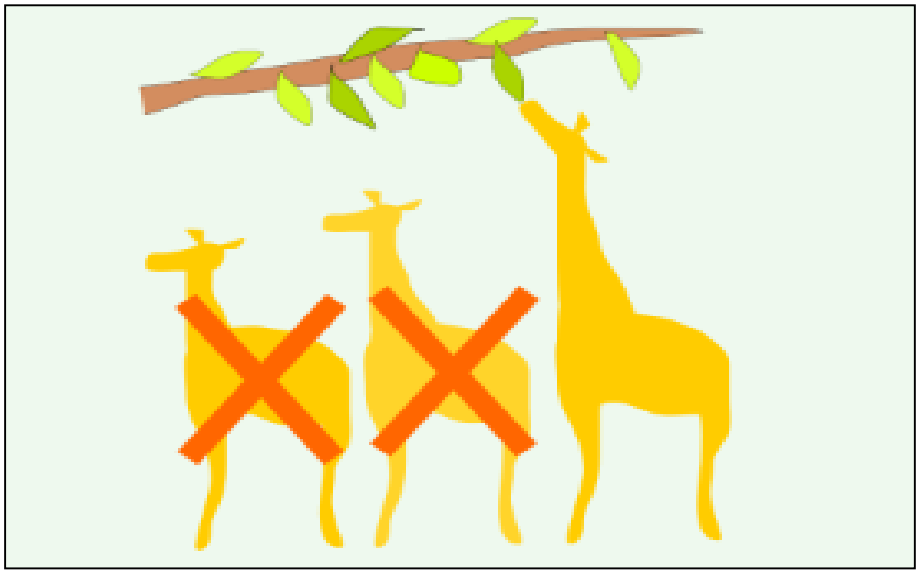

Figure 2.2représentation du principe d'adaptation [b].

3.3 Principe d'hérédité

Les individus doivent avoir des caractéristiques héréditaires qui sont transmises à leur des cendance, car cela assure l'évolution de l'espèce tout en restant des propriétés bénéfiques pour les individus.

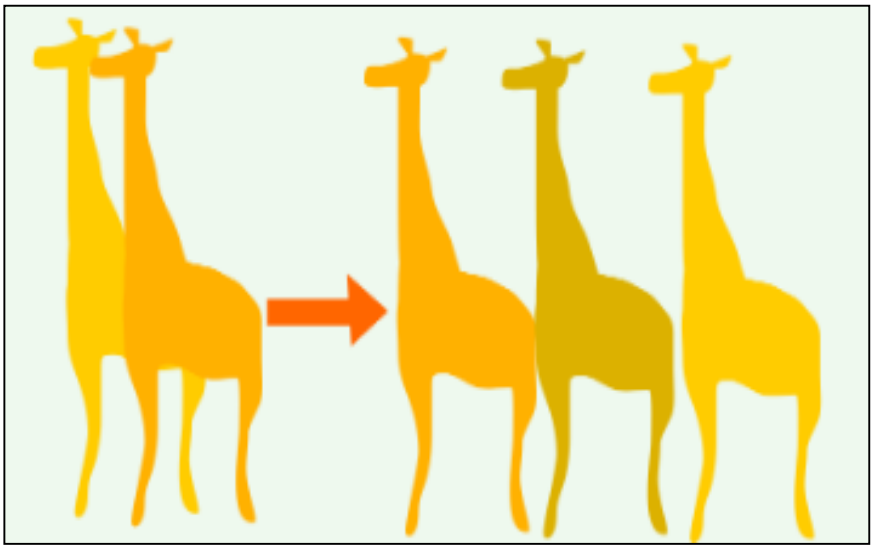

Figure 2.3représentation Principe d'hérédité [b].

4. Les opérateurs d'algorithmes génétiques

4.1 L'opérateur de sélection

La sélection c'est l'application du principe d'adaptation de la théorie de Darwin tell que il signifie la manière de choisir les individus à partir de la connaissance quel sont les individus les mieux adaptés afin d'avoir une population de solution la plus proche de converger vers l'optimum global, il y a des techniques de sélection nous citons :

- **Sélection par rang: on va ranger les individus a partirde leur scores et on va choisir les individus qui possèdent les** meilleurs scores d'adaptation.

- **Probabilité de sélection proportionnelle à l'adaptation: c'est la technique de la roulette ou roue de la fortune, en effet la probabilité de choisir chaque individu est liéà son adaptation au problème.**

individu	Fonction objectif	pourcentage
1	30	39%
2	5	7%
3	16	21%
4	10	13%
5	15	15%

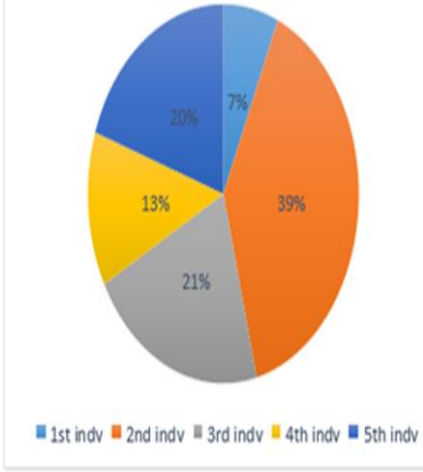

Figure 2.4 Probabilité de sélection proportionnelle à l'adaptation

- **Sélection par tournoi :** on va sélectionner proportionnellement sur des paires d'individus, après on choisit parmi ces paires l'individu qui a la meilleur score d'adaptation.

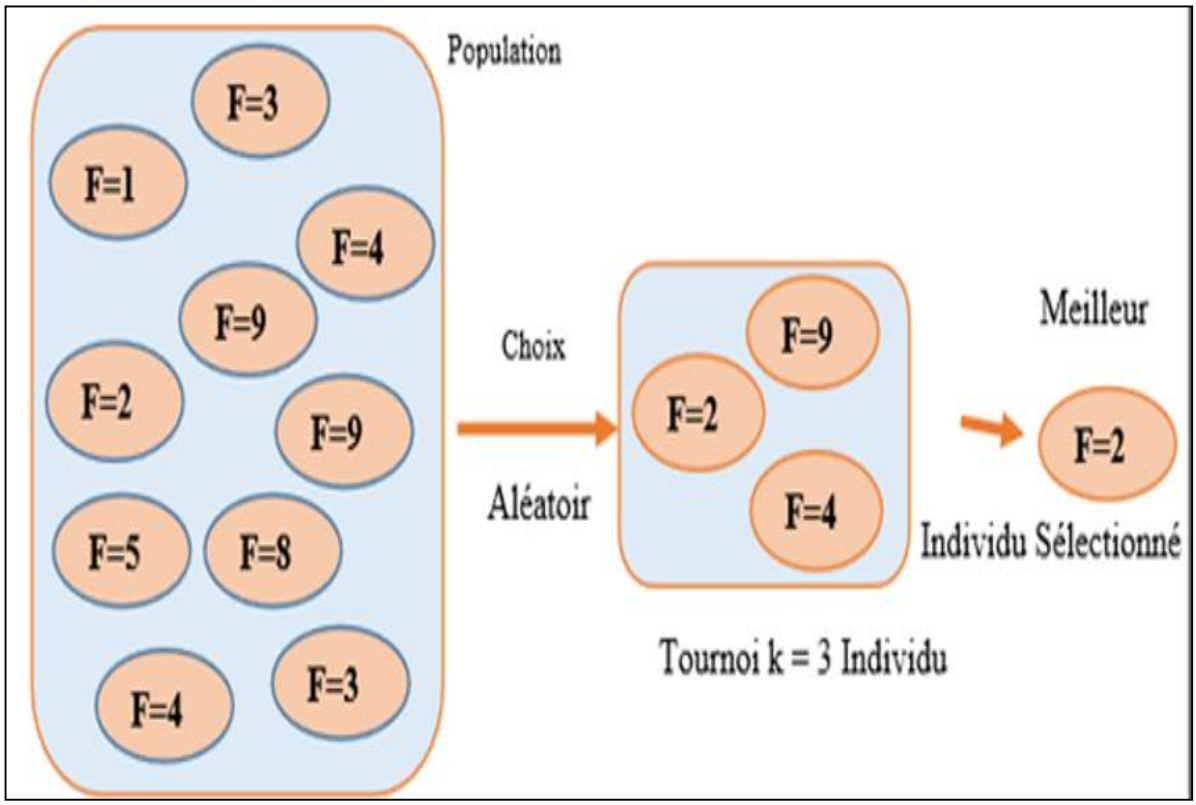

Figure 2.5 Sélection par tournoi.

- **Sélection uniforme :** on va sélectionner aléatoirement, uniformément sans tenir compte de la valeur d'adaptation.

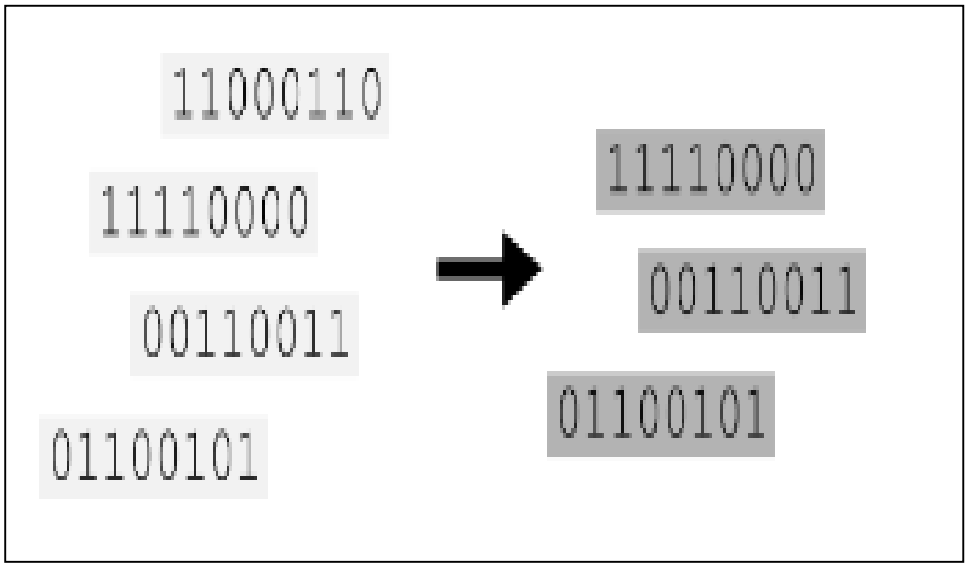

Figure 2.6 Individus en représentation binaire une fois la sélection effectuée [16].

4.2 L'opérateur de croisement

Le croisement c'est les chromosomes qui on va obtiendra lorsque deux chromosomes partagent leurs particularités. Celui-ci permet le brassage génétique de la population et l'application du principe d'hérédité de la théorie de Darwin.

Il y a deux méthodes pour le croisement :

- **Simple croisement (une seul point):**

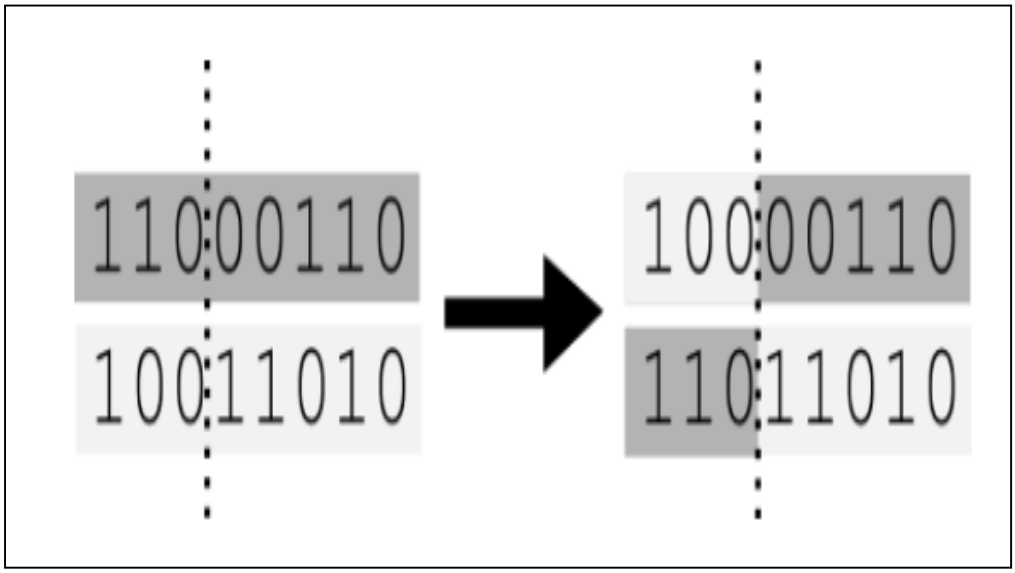

Figure 2.7 croisement sur une seul point [16].

- **Double croisement (deux points) :**

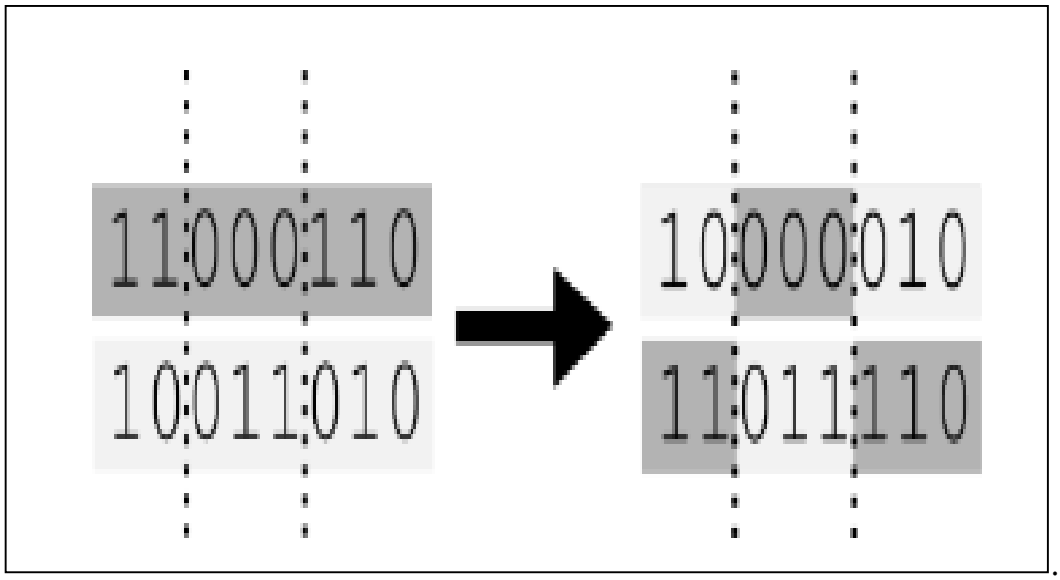

Figure 2.8 croisement sur deux points[16].

4.3 L'opérateur de mutation

La mutation est l'application du principe de variation de la théorie de Darwin, en effet la mutation est le changement d'un gène dans un chromosome selon un facteur de mutation qui est la probabilité, une mutation soit effectuée sur un individu ,pour permettre d'éviter une convergence prématurée de l'algorithme vers un extremum local.

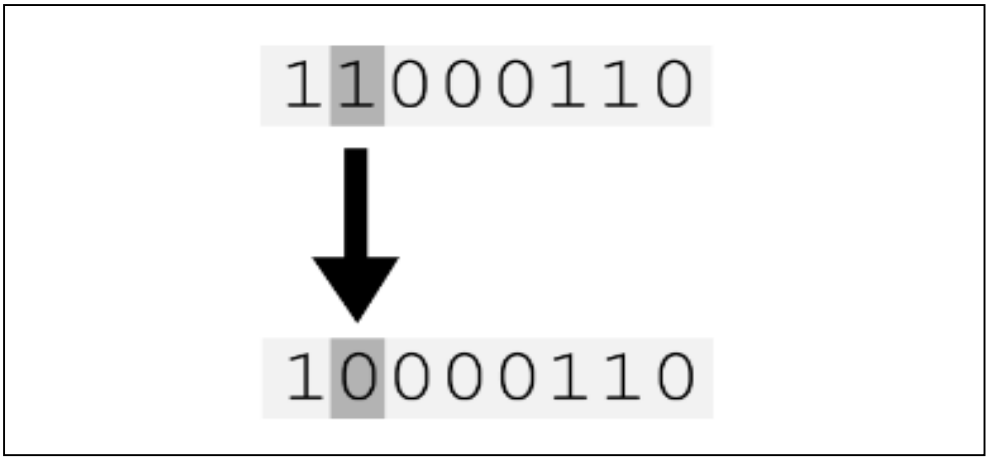

Figure 2.9représentation d'opérateurdu mutation[16].

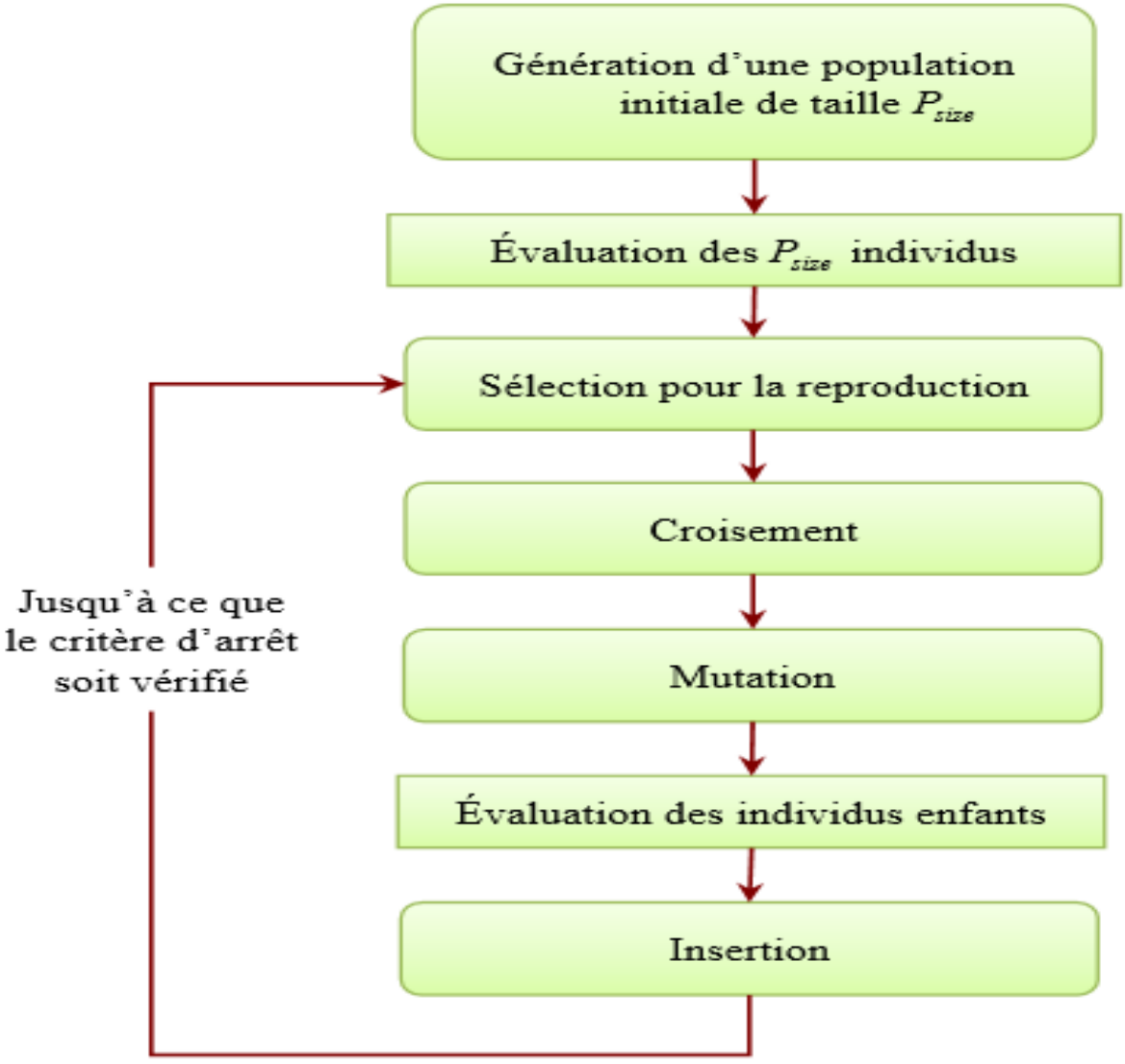

Figure 2 .10 le fonctionnement général des algorithmes génétiques.

Explications :

- Le début de l'algorithme à partir de création du population est aléatoire.
- Après ça on va faire l'évaluation, c.-à-d. si une solution est disponible, pour ceci nous utilisons « fitness », afin de définir le score d'adaptation des individus lors du processus de sélection.
- Tous les opérateurs appliquent dans boucle à la fin on choisit la solution qui possède la meilleur fitness.

5. Les méta-heuristiques

5.1 Recuit simulé

Le recuit simulé (RS) est une méta-heuristique connue pour être la plus ancienne. Elle a été proposée par Kirkpatrick en 1983 d'après le travail de Metropolis d'où elle puise ses origines statistiques. Le terme recuit est inspiré d'un processus utilisé en métallurgie dans lequel on fait alterner les cycles de chauffage et de refroidissement des métaux pour minimiser l'énergie des matériaux. Cette méta-heuristique utilise une approche de Monte Carlo pour simuler le comportement d'un système qui tend à atteindre un équilibre thermal ainsi devenir stable, cette analogie est utilisée pour résoudre les problèmes d'optimisation combinatoire. Il a été prouvé qu'en surveillant de manière précise le taux de refroidissement, l'algorithme est capable de trouver l'optimum global mais paradoxalement, ceci prendrait un temps infini. C'est pour cela que d'autres versions du recuit simulé, le « Fastannealing » et le « VeryFastSimulatedReannealing » (VFSR) qui sont exponentiellement plus rapides que la version de base, sont utilisées pour surmonter le problème des délais. Le recuit simulé possède un sérieux avantage par rapport aux autres méthodes par le fait qu'il ne se fait par piéger dans les minima locaux [15].

```
s= Generer_Solution_Initiale()
T=T0
Tant que (condition d'arrêt non satisfaite)
    s'=Selection_Aleatoire(N(s))
    si f(s')<f(s)
        s=s'
    sinon
        Accepter s' en tant que nouvelle solution avec une probabilité p(T,s',s)
    Mise_A_Jour(T)
```

Algorithme 2.1 L'algorithme de RS.

5.2 GRASP

Le GRASP (GreeclyRandomized Adaptive SearchProcedure) est une méta-heuristique qui consiste en une suite de solutions construites par une approche Vorace et à leur optimisation par l'exploration de leurs voisinages respectifs [18]. Chaque itération de cette méta-heuristique consiste en une phase de construction de solution et en une exploration de voisinage. La construction se fait élément par élément, le choix du prochain élément se faisant par une approche Vorace. GRASP

essaie de tirer à la fois avantage de l'approche Vorace et de l'approche aléatoire. GRASP garde une trace de la meilleure solution et la retourne à la fin de l'algorithme [15].

```
s* = ∞
Tant que (Condition d'arrêt non satisfaite)
        Construire_Solution_Vorace(s)
        Trouver le minimum local s' dans N(s)
        si f(s')<f(s*)
                s*=s'
Retourner meilleure solution de s*
```

Algorithme 2.2 L'algorithme de GRASP .

5.3 Recherche avec Tabous

Il y a une autre méta-heuristique c'est la recherche avec tabous (RT) qui été proposée en 1986 par Fred Glover [6]. En général le principe de RT est : on possèdent les solutions, on cherche les solutions qui sont proches, donc celles qu'on peut atteindre par de simples modifications de la solution initiale [15]. On obtient un ensemble de solutions appelées voisinage. Une particularité est toutefois de garder en mémoire, sous forme d'une liste taboue, les espaces déjà explorés pour éviter d'y retourner.

```
Générer une solution initiale s
s*=s
k=0
Initialiser la liste taboue
Tant que (Condition d'arrêt non satisfaite)
        Déterminer la meilleure solution (s_{k+1}) dans N(s_k)
            en tenant compte de la liste taboue
        Si f(s_{k+1})<f(s*)
                s*=s_{k+1}
        k=k+1
        Mise à jour de la liste taboue
```

Algorithme 2.3 L'algorithme de RT.

6. Le parallélisme

Le parallélisme est une architecture pour traiter l'information de manière simultané, c'est-à-dire on peut faire plusieurs operateurs au même temps, pour gagner le temps .

6.1 Les modèles des algorithmes génétiques parallèles(AGPs)

6.1.1 AGP maître-esclave

On suppose qu'il n'y a qu'une seule population panmictique, c'est-à-dire une AG canonique. Cependant, contrairement à l'AG canonique, l'évaluations des individus sont distribuées en ordonnant des fractions de la population parmi les nœuds esclaves de traitement. Un tel modèle présente l'avantage d'être facile à mettre en œuvre et ne modifie pas le comportement de recherche d'un AG canonique.

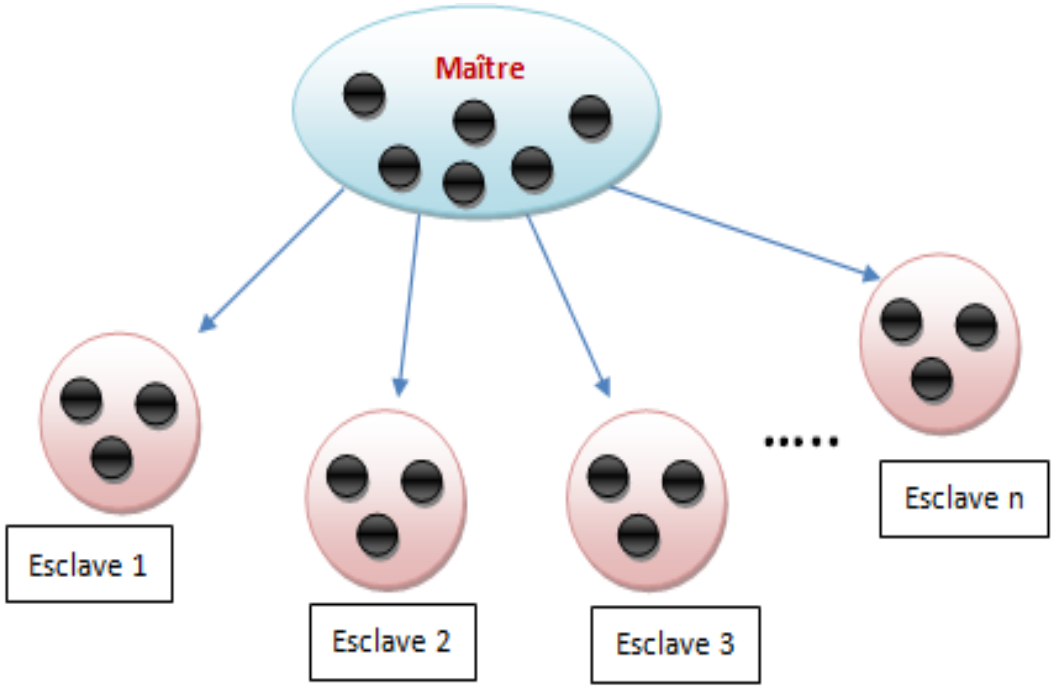

Figure 2 .11 Le modèle AGP maître-esclave

6.1.2 AGP à grains fins

L'AGP à grains fins ne se compose d'une seule population, qui est structurée spatialement. Il est conçu pour fonctionner sur un système de traitement massivement parallèle étroitement lié, c'est-à-dire un système informatique composé d'un grand nombre d'éléments de traitement et connecté dans une topologie à grande vitesse spécifique. Par exemple, la population d'individus dans un AGP grain fin peut être organisée comme une grille bidimensionnelle. Par conséquent, la sélection et l'accouplement dans un AG parallèle à grains fins sont limités à de petits groupes. Néanmoins, les groupes se chevauchent pour permettre certaines interactions entre tous les individus afin que les bonnes solutions puissent se diffuser à travers les populations entières. Parfois, AG parallèle à grain fin est également appelée modèle AGP cellulaire.

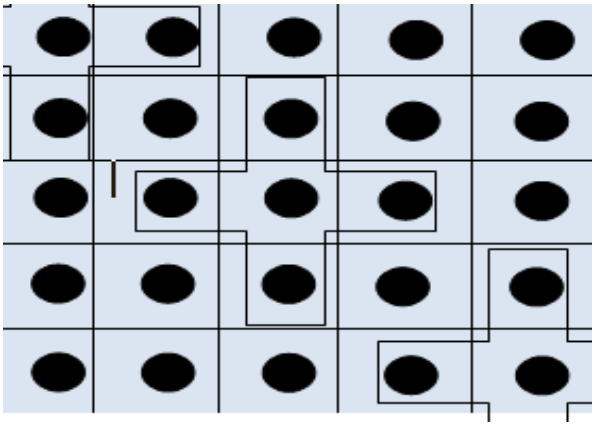

Figure 2 .12 Le modèle AGP à grains fins

6.1.3 La AGP à populations multiples

La AGP à populations multiples (ou dèmes) peut être plus sophistiquée, car elle se compose de plusieurs sous-populations qui échangent des individus à l'occasion. Cet échange d'individus est appelé migration et il est contrôlé par plusieurs paramètres. Les AGP à population multiple sont également connus sous divers noms. Puisqu'ils ressemblent au «modèle insulaire» en génétique des populations qui considère les dèmes relativement isolés, il est aussi souvent connu sous le nom de modèle insulaire AGP.

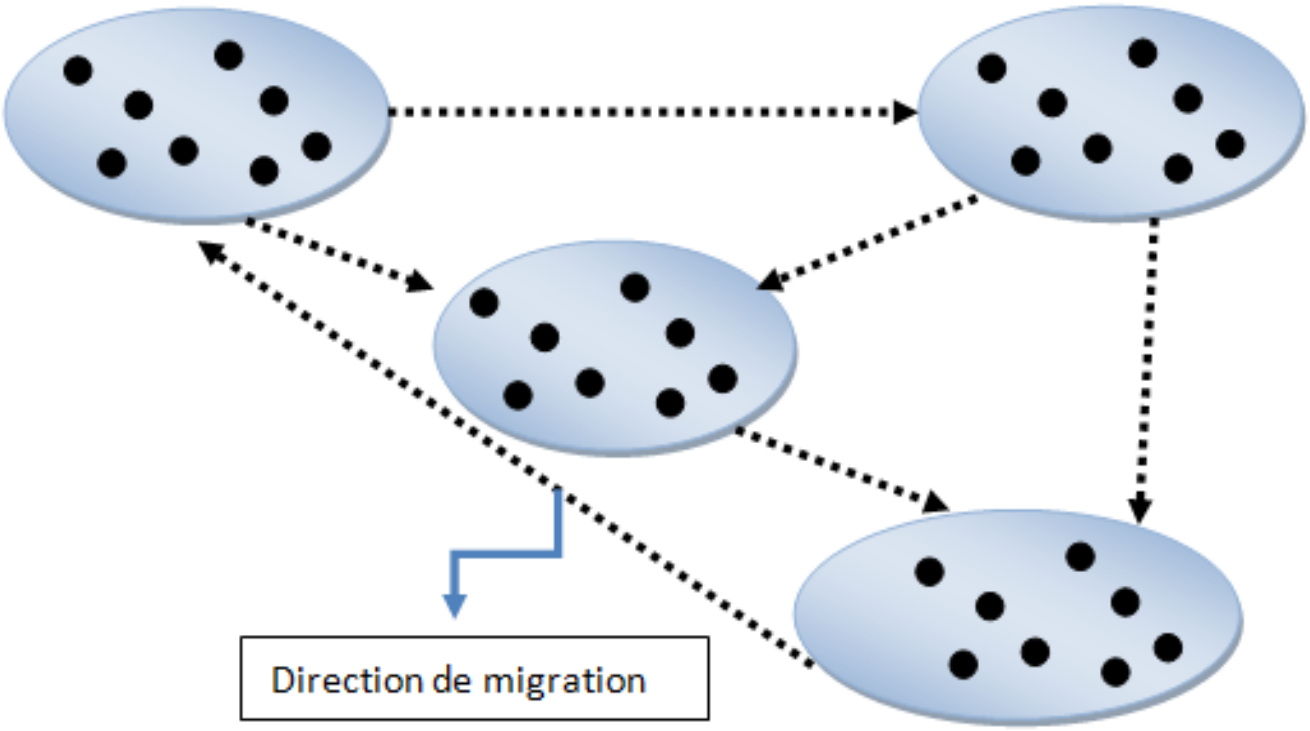

Figure 2 .13 Le modèle AGP à populations multiples .

7. Conclusion

Dans ce chapitre, nous introduisons les concepts des algorithmes génétiques et leurs principes (**Le principe de variation, Le principe d'adaptation, Le principe d'hérédité**), après nous sommes passés par les opérateurs des l'algorithmes génétique(sélection ,croisement, mutation) ,Ensuite, nous avons examiné les algorithmes génétiques parallèles et certains de leurs modèles .

CHAPITRE 3 L'ETAT DE L'ART

1. Introduction

Dans ce chapitre, on représente des articles des chercheures fissent des recherches en notre thème du mémoire, alors en résume chaque article comme suite :

(2004)Nourah Al-Angari, AbdullatifALAbdullatif appliquent les algorithmes génétiques parallèles avec succès pour ressouder les problèmes des planifications du taches. L'évaluation de « fitness » est l'opération qui consomme le plus long temps de CPU, ce qui affecte les performances AG. L'algorithme proposé de maître-esclave synchrone surpasse l'algorithme séquentiel en cas de problème complexe et élevé de générations.

(2019) JiaLuoa, ShigeruFujimurab et Didier El proposent un modèle de planification Flow Shop, utilisant la valeur de puissance de crête en tenant compte de nouvelles fonctions. Comme le problème est fortement NP-difficile, à cause de la nouvelle législation gouvernementale, des préoccupations environnementales des clients et de l'augmentation constante du coût de l'énergie, l'efficacité énergétique est devenue un paramètre essentiel des processus de fabrication industrielle ces dernières années. La plupart des efforts, compte tenu des problèmes énergétiques dans les problèmes de programmation, se sont concentrés sur la programmation statique. Mais en fait, les problèmes d'ordonnancement sont dynamiques dans le monde réel avec de nouveaux emplois incertains après l'heure d'exécution. En effet ils développent un algorithme génétique hybride parallèle basé sur les priorités avec une approche de rééchelonnement complet réactif prédictif. Afin d'obtenir une accélération pour répondre à la réponse courte dans l'environnement dynamique, la méthode proposée est conçue pour être hautement cohérente avec le modèle logiciel NVIDIA CUDA. Enfin, des expériences numériques sont menées et montrent que leur approche peut non seulement atteindre de meilleures performances que l'approche statique traditionnelle, mais aussi obtenir des résultats compétitifs en réduisant considérablement les exigences de temps**.**

(2018) J.Adan,A.Akcay, J.Stokkermansb et R.VandenDobbelsteentraitent Un algorithme génétique hybride pour améliorer le processus de planification, dont les principales caractéristiques sont un mécanisme de croisement amélioré pour la recherche locale, deux procédures de recherche locale rapide supplémentaires et une fonction d'ajustement multi-objectif contrôlée par l'utilisateur. Les tests avec des données de production réelles montrent que cette approche multi-objectifs peut atteindre l'équilibre souhaité entre le temps de

production, le temps de configuration et les retards, produisant des calendriers de production de haute qualité pratiquement réalisables.

(2006) R.Nedunchelian, K.Koushik, N.Meiyappan, V.RaghudéveloppentUn algorithme génétique pour planifier dynamiquement des tâches hétérogènes à des processeurs hétérogènes dans un environnement distribué. Le problème de planification est connu pour être NP-complet. Les algorithmes génétiques, une technique de recherche méta-heuristique, ont été utilisés avec succès dans ce domaine. L'algorithme proposé utilise plusieurs processeurs avec un contrôle centralisé pour la planification. Les tâches sont prises en lots et sont planifiées pour minimiser le temps d'exécution et équilibrer les charges des processeurs. Selon leurs résultats expérimentaux, l'algorithme génétique parallèle proposé (PPGA) diminue considérablement le temps de programmation sans affecter négativement Makespan des programmes résultants

(2013) FrankWerner donne un aperçu de certains algorithmes génétiques pour les problèmes de Shop Scheduling. Dans un problème de shop Scheduling, un ensemble de Jobs doit être traité sur un ensemble de machines de telle sorte qu'un critère d'optimisation spécifique soit satisfait. En fonction des restrictions sur les itinéraires technologiques des emplois, on distingue Flow Shop (chaque Job est caractérisé par le même itinéraire technologique), Job Shop (chaque Job a un itinéraire spécifique) et Open Shop (aucun itinéraire technologique n'est imposé sur les Jobs). Il considère également certaines extensions des problèmes de Shop Scheduling tels que Shop hybrides ou flexibles (à chaque étape de traitement, nous pouvons avoir un ensemble de machines parallèles) ou l'inclusion de contraintes de traitement supplémentaires telles que les temps de traitement contrôlables, les temps de publication, les temps de configuration ou le condition sans attente. Après avoir donne une introduction aux algorithmes génétiques de base discutant des représentations de solutions brèves, de la génération de la population initiale, des principes de sélection, de l'application d'opérateurs génétiques tels que le croisement et la mutation, et des critères de terminaison, il discute de plusieurs algorithmes génétiques pour les types de problèmes particuliers en mettant l'accent sur leur caractéristiques et différences communes. Ici, il concentre principalement sur les problèmes à critère unique (minimisation de la durée de validité ou d'un critère de somme particulier tel que le temps total d'achèvement ou le retard total), mais mentionne brièvement quelques travaux sur les problèmes multicritères. il discute de certains résultats de calcul et les comparons avec ceux obtenue par d'autres heuristiques.

En outre, il résume également la génération d'instances de référence pour les problèmes de durée de fabrication et donne une brève introduction à l'utilisation du package de programmes « LiSA-ALibrary of SchedulingAlgorithms » développe à « Otto-von-Guericke-University Magdeburg » pour résoudre les problèmes de Shop Scheduling , qui comprend également un algorithme génétique.

(2007) Kheireddine MERHOUM et Messaoud DJEGHABA développent une application qui permet de minimiser le makespan pour un problème d'ordonnancement job-shop flexible ils utilisèrent les Algorithme génétique. Cependant le problème d'ordonnancement job-shop flexible dans la littérature est considéré comme une problématique difficile à résoudre dans le domaine de l'optimisation combinatoire, sa complexité est de type NP-complet au sens fort. L'objectif est montré les performances des algorithmes génétiques (métaheuristique) dans la résolution de ce genre de problème.

(1997) Shyh-Chang Lin, Erik D. Goodman et William F. Punch décriassent une AG pour les problèmes de Job Shop Scheduling. En utilisant l'algorithme de Giffler et Thompson, ils avent créés deux nouveaux opérateurs, le croisement THX et la mutation, qui transmettent mieux les relations temporelles dans le planning. L'approche a produit d'excellents résultats sur les problèmes de job shop Scheduling. Ils avent testé de nombreux modèles et échelles d'AG parallèles dans le contexte de problèmes de job shop scheduling. Le modèle hybride composé d'Algorithme génétique Coarse-grain à connectés dans une topologie de style AG fine-grain a donné les meilleurs résultats, semblant intégrer avec succès les avantages des AG coarse-grain et fine-grain .

(1998) Erick Cantú-Paz organise et présente la manière unifiée certaines des publications les plus représentatives sur les algorithmes génétiques parallèles. Pour organiser la littérature, l'article présente une catégorisation des techniques utilisées pour paralléliser les AG, et montre des exemples de toutes. Cependant, comme la majorité des recherches dans ce domaine se sont concentrées sur des AG parallèles à populations multiples, l'enquête se concentre sur ce type d'algorithmes. En outre, l'article décrit certains des problèmes les plus importants dans la modélisation et la conception d'AG parallèles à plusieurs populations et présente quelques avancées récentes.

(1993) Harpal Mainiprésente des algorithmes génétiques - des algorithmes évolutionnaires basés sur une analogie avec sélection naturelle et survie du plus apte - appliqué à une combinatoire NP-Complete problème d'optimisation: minimisation de la durée de vie d'un Flow Shop No Wait(FSNW). Ceci est un critère d'optimisation important dans des situations réelles et le problème elle-même à une signification pratique. Nous limitons nos applications à

Flow Shop de trois machines aucun problème d'attente connu pour être NP-complet. L'hypothèse stochastique est que le les temps de traitement des Jobs sont décrits par des variables aléatoires normalement distribuées. Il discute comment ce problème peut être traduit en un problème TSP, il utilise le concept d'intervalle de démarrage. Des algorithmes génétiques, séquentiels et parallèles sont ensuite appliqués pour rechercher l'espace de la solution et il présente les algorithmes et les résultats empiriques.

(2017) ArtanBerisha, EliotBytyc et ArdeshirTershnjakuont essayé avec de nombreuses techniques de trouver le moyen le plus approprié et le plus rapide pour résoudre le problème. Avec l'émergence de systèmes multi-cœurs, la mise en œuvre parallèle a été envisagée pour trouver la solution,leurs approches tentent de combiner plusieurs techniques dans deux algorithmes: l'algorithme à grain grossier et l'algorithme de tournoi multi-thread. Les résultats obtenus à partir de deux algorithmes sont comparés à l'aide d'une fonction d'évaluation d'algorithme. Compte tenu du temps d'exécution, l'algorithme à grain grossier a fait deux fois mieux que l'algorithme multi-thread.

(2018) JiaLuo et Didier EL BAZ leurs travaux ont été consacrés aux algorithmes génétiques (AG) pour rechercher des solutions optimales aux problèmes d'ordonnancement des magasins. En raison de la dureté NP, le coût en temps est toujours lourd. Avec le développement du calcul haute performance (HPC), l'intérêt s'est concentré sur les AG parallèles pour les problèmes de Shop Scheduling. Ils présentent les travaux récents sur la résolution des problèmes de Shop Scheduling avec l'utilisation des AG parallèles. Il présente les publications les plus représentatives dans ce domaine par la catégorisation des AG parallèles et analyse leurs conceptions en fonction des cadres.

(1993) Hsiao-Lan Fang, Peter Ross et Dave Corne décrivent une approche AG qui produit des résultats raisonnablement bons très rapidement sur les problèmes job-shop Scheduling, mieux que les efforts précédents utilisant des algorithmes génétiques pour cette tâche, et comparable aux méthodes de recherche conventionnelles existantes. La représentation utilisée est une variante de celle connue pour fonctionner assez bien pour le problème du voyageur de commerce. Il a le mérite considérable que le croisement produira toujours des horaires légaux. Une nouvelle méthode d'amélioration des performances est examinée sur la base d'un échantillonnage dynamique des taux de convergence dans différentes parties du génome. Leur approche promet également de résoudre efficacement le problème d'Open Shop Scheduling et le problème dejob-shop Scheduling.

(2007)Dudy Lim , Yew-Soon Ong , YaochuJinb ,Bernhard Sendhoff et Bu-Sung Lee présentent Framework d'algorithmegénétiquehiérarchiqueparallèle avec l'utilisation du Grid Computing (GE-HPGA). Framework est développé à l'aide des technologies Grid standard et possède deux caractéristiques distinctives, premièrement une API Grid RPC étendue pour masquer la grande complexité de l'environnement Grid, et deuxièmes pour la découverte et la sélection de ressources en toute transparence. Pour évaluer le caractère pratique du Framework, une analyse théorique de l'accélération possible proposée est présentée. Une étude empirique sur GE-HPGA utilisant un problème de référence et un problème réaliste d'optimisation de la forme des profils aérodynamiques pour divers environnements de grille ayant des protocoles de communication différents, des tailles de cluster, des nœuds de traitement, à des emplacements géographiquement disparates indique également que le GE-HPGA proposé utilisant le calcul de grille propose une Framework crédible permettant d'accélérer considérablement l'optimisation de la conception évolutive en science et en ingénierie.

(2004) Murat Yildizoglu et Thomas Vallée présentent les mécanismes de base de ces algorithmes et un panorama de leurs applications en économie, accompagnés d'une bibliographie représentative.

(2016) Rakesh Kumar PHANDEN représente Job Shop Scheduling qui est un problème important et complexe pour un système de fabrication. C'est un problème bien connu et populaire ayant une caractéristique NP-hard (non polynomiale) de trouver rapidement la solution optimale ou quasi optimale (horaires). Dans Job Shop Scheduling, un ensemble de nombres "N" de Jobs est traité par le biais d'un nombre "M" d'un ensemble donné de machines. Il doit être traité dans l'ordre prescrit en utilisant la séquence d'opérations réalisable pour un travail. Par conséquent, en raison de sa nature complexe, la recherche de solutions approximatives est choisie plutôt que la recherche de la solution exacte qui implique un coût plus élevé. Diverses techniques méta-heuristiques sont utilisées afin de trouver la solution sous-optimale pour le problème de planification de l'atelier de travail. L'algorithme génétique et la méthode de recherche de voisinage variable (VNS) sont les techniques préférées qui sont mieux connues pour la recherche globale et locale de solutions, respectivement. VNS fonctionne comme pour augmenter l'approche GA. Dans le présent travail, les multi-agents sont proposés pour trouver la solution presque optimale pour le problème de planification de l'atelier en utilisant l'approche GA et VNS en parallèle. Le système multi-agents est préféré en raison de sa capacité à fonctionner en parallèle et de sa robustesse ainsi que de l'élucidation de l'intelligence. Dans le système proposé, de nombreux hôtes du réseau sont hébergés par des

agents. JADE est utilisé pour configurer les communications. Chaque agent est conçu pour effectuer la tâche spécifique à savoir l'agent d'initialisation (IA), l'agent de traitement (PA) et l'agent de coordination (CA) pour la génération initiale de la population, pour planifier les opérations sur les machines, pour trouver l'hôte distinctif et pour effectuer les migrations entre différentes populations respectivement. L'objectif est de trouver une valeur optimale de makespan pour le problème de planification de l'atelier de travail. La performance du système est évaluée par une étude de cas et elle révèle que l'approche proposée est suffisamment efficace pour trouver la solution optimale. Les travaux futurs consistent à introduire des agents de perturbation (DA) pour les perturbations internes et externes.

(2012)Mostafa Akhshabi, JavadHaddadnia et Mohammad Akhshabiutilisent un GA parallèle pour résoudre les problèmes d'ordonnancement de Flow Shop afin de minimiser le makespan. Selon leurs résultats expérimentaux, l'algorithme génétique parallèle proposé (PPGA) diminue considérablement le temps CPU sans affecter défavorablement le makespan.

(2004)Fabien PICAROUGNE, Gilles VENTURINI et Christiane GUINOT présentent un algorithme génétique (AG) parallèle qui explore le Web dans le but de trouver des documents pertinents dans le contexte de la veille stratégique. Ils montrent comment le problème de recherche d'information sur Internet peut être modélisé en un problème d'optimisation : Internet est un espace de recherche structuré sous forme de graphe, et une fonction d'évaluation peut être définie à partir de la requête de l'utilisateur. L'AG gère une population de pages Web et décide quelles pages explorer. L'architecture parallèle de Geni Miner II est distribuée sur un réseau local ou sur Internet, chaque client ayant la possibilité de s'enregistrer et de devenir une partie du moteur de recherche. Ils montrent également que l'AG parallèle obtient de meilleurs résultats en comparaison avec un méta moteur de recherche, et qu'il diminue fortement le temps nécessaire à l'obtention des documents. Enfin, les premiers tests réalisés avec des utilisateurs réels montrent le potentiel de ce système et les orientations futures à prendre en compte.

(2003) Michelle Moore présentealgorithmes génétiques parallèles qui sont appliquées au problème NP-complet de planification de plusieurs tâches sur un cluster d'ordinateurs connectés par un bus partagé. Les expériences révèlent que l'algorithme de programmation parallèle développe des programmes très précis lorsque les directives de paramètres sont utilisées.

(2019)Yuri N. Sotskov, Natalja M. Matsveichuk et Vadzim D. Hatsuraétudient les problèmes de Shop Scheduling à deux machines à condition que les limites inférieures et supérieures des durées de ' n' Jobs soient données avant la planification. Une valeur exacte

de la durée du travail reste inconnue jusqu'à la fin du Job. L'objectif est de minimiser le makespan (durée du planning). Ils abordent la question de la meilleure façon d'exécuter un calendrier si la durée du travail peut prendre une valeur réelle du segment donné. Les décisions de programmation peuvent comprendre deux phases: une phase hors ligne et une phase en ligne. En utilisant des informations sur les limites inférieures et supérieures pour chaque durée de Job disponible lors de la phase hors ligne, un planificateur peut déterminer un ensemble dominant minimal de plannings (DS) sur la base de conditions suffisantes pour la domination du programme. Le DS couvre de manière optimale toutes les réalisations (scénarios) possibles des durées de Job incertaines dans le sens où, pour chaque scénario possible, il existe au moins un calendrier dans le DS qui est optimal. Le DS permet à un planificateur de prendre rapidement une décision de planification en ligne chaque fois que des informations supplémentaires sur l'achèvement des travaux sont disponibles. Un planificateur peut choisir un calendrier optimal pour les scénarios les plus possibles. Nous avons développé des algorithmes pour tester un ensemble de conditions pour une domination d'horaire. Ces algorithmes sont polynomiaux dans le nombre de travaux. Leur complexité temporelle ne dépasse pas. Des expériences informatiques ont montré l'efficacité des algorithmes développés. S'il n'y avait pas plus de 600 tâches, les 1000 instances de chaque série testée ont été résolues en une seconde au plus. Un cas avec 10 000 tâches a été résolu en 0,4 s en moyenne. La plupart des instances de neuf classes testées ont été résolues de manière optimale. Si l'erreur relative maximale de la durée du travail n'était pas supérieure à, alors plus que les instances testées ont été résolues de manière optimale. Si l'erreur relative maximale était égale à, alors les instances testées des neuf classes ont été résolues de manière optimale.

(2019)HOUNNOU Amèdédjihundé H, FIFATIN François-Xavier, DUBAS Frédéric, CHAMAGNE Didier et VIANOU Antoine développent un nouveau concept de dimensionnement avec l'utilisation d'optimisation bi-objective avec les algorithmes génétiques NSGA II. Les deux fonctions objectives considérées sont le coût d'investissement et le rendement de la conduite forcée. Le diamètre et la longueur de la conduite forcée sont considérés comme variables d'optimisation. La méthode de dimensionnement est appliquée à trois sites potentiels en aménagement hydroélectrique du Bénin. Pour chaque site, les résultats de simulation présentent des courbes de front de Pareto qui représentent l'ensemble des solutions non dominées. Ils ont noté que le diamètre est un paramètre déterminant dans l'optimisation bi-objectif du coût d'investissement et du rendement de la conduite forcée. La longueur ne constitue pas un paramètre d'optimisation mais est plutôt est un paramètre spécifique à définir en tenant compte des contraintes environnementales du site. Cette étude présente aussi l'avantage qu'elle peut permettre de dimensionner la conduite forcée pour un site quelconque.

CHAPITR 4 RESSOUDER LE PROBLEM

1.Introduction

En ce chapitre on expose l'idée pour ressouder un problème d'ordonnancement de job shop avec l'utilisation des algorithmes génétique parallèles et on essaye de minimiser le Makespane.

2. Présentation de problème

En générale le problème de job shop est comme suit :

- En a n jobs$\{J_i\}_{1\leq j\leq n}$ vont traiter en m machines $\{M_j\}_{1\leq j\leq m}$.
- Chaque machine traiter un seul job pour un moment de temps.
- Il faut traiter tous les jobs en toutes les machines.
- O_{ij} est L'opération de job J_i qui traite en M_i.
- Le temps de début d'opération O_{ij} est S_{ij} et le temps de fin d'operation est C_{ij}.
- Le temps qui on besoin pour finir le traitement de tous les Jobes en tous les machines est MakespaneC_{max}.Tell que C_{max}=$max\{C_{ij}\}_{1\leq n,1\leq j\leq m}$.

On posse un exemple de Job Shop aves 3 jobs et 3 machines comme suite :

Job	Machine(le temps de traitement)		
1	1(4)	2(4)	3(4)
2	1(3)	3(4)	2(5)
3	2(4)	1(3)	3(2)

Table 4.1exemple de job shop.

On représente le problème aves la matrice de séquence des Jobes$\{T_{ik}\}$ et la matrice de temps de traitement $\{P_{ik}\}$ comme suite :

$$\{T_{ik}\} = \begin{vmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{vmatrix} \{P_{ik}\} = \begin{vmatrix} 4 & 4 & 4 \\ 3 & 4 & 5 \\ 4 & 3 & 2 \end{vmatrix}$$

3. Présentation de la solution

On représente algorithme génétique pour ressoude le problème de Job Shop comme suite :

- Prior-rule est proposée par Giffler et Thompson pour créer un Schedule active.
- La mutation utilisée technique de NieghborhoodSearching
- On pose le nombre des opérations des JobesJ_i tell que job[i](job[i]≤m ,pour tous les Jobs).
- La somme des opérations qui traitée est noté par L ,en effet L= $\sum_{i=1}^{n} job[i]$.

On code les opérations du problème précédent comme suite :

Jobs	opération		
j_1	1	2	3
j_2	4	5	6
j_3	7	8	9

Table 4.2 les opérations de l'exemple précédent.

3.1 Générer la solution initiale

a partir la matrice $\{T_{jk}\}$ les opérations {1,4 ,8} sont traiter en M_1, alors M_2 traite les operations {2,6,7} et M_3 traite {3,5,9}

Machine	Opération(job)		
M_1	$1(J_1)$	$4(J_2)$	$8(J_3)$

M_2	$7(J_3)$	$6(J_2)$	$2(J_1)$
M_3	$5(J_2)$	$3(J_1)$	$9(J_3)$

Table 4.3 La solution initiale del'exemple précédent.

On pose la matrice de solution initiale $S_{jk} = i$, en effet K^{th} est l'opération en $M_j$est J_i

$$\{S_{jk}\} = \begin{vmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \end{vmatrix}$$

On va créer plusieurs des matrices S_{jk}a partir le changement des codages des opérations et en choisit la population initiale aléatoirement.

L'algorithme de génération de population initiale comme suite :

```
Début
    Ecrire (entrer la taille de population)
    Lire (n)
    Générer un premier solution
    K←1
     Pour i←1 à n faire
          Si (la solution sans cycle ) alors
               Si (la solution=1)
                    K++ ;
                    Sinom
                         La solution est existe
               Finsi
               Sinom
              Solution nom réalisable
          Finsi
     Finpour
Fin
```

Algorithme 4.1 L'algorithme de génération de population initiale.

3.2 La fonction objective « fitness »

Fitness = M - C_{max} , tell que C_{max}est Makespane et M est un paramètre pour change le problème minimal a maximale.

3.3 Mutation

Les étapes de la mutation est :

- Sélectionne une opération (op_1) qui traitée par la machine (M_{op_1}) et qui possède la position (pos_1).

- Sélectionne une opération (op_2) qui traitée par la machine (M_{op_2}) et qui possède la position (pos_2)
- Si $M_{op_1} = M_{op_2}$ on fit la mutation, en effet on change leur position.

L'algorithme de génération de la mutation comme suite :

```
Début
  sélectionner aléatoire op_1,op_2 // traiter en M(op_1),M(op_2).
  position(op_1)=pos_1
  position(op_2)=pos_2
   K=1
    Pour k=1 à n faire
        Si (M(op_1)=M(op_2) ) alors
              Position(op_1)←Position(op_2)
              K++ ;
            Sinom
              Selectioner autre individu
       Finsi
    Finpour
Fin
```

Algorithme 4.2 L'algorithme de génération de la mutation.

3.4 Croissement

On fit le croisement avec l'utilisation Algorithme GT en trois parents P_1, P_2, P_3 [2]:

Parent 1 :

mahine	Opération(Job)		
M_1	$4(j_2)$	$8(j_3)$	$1(j_1)$
M_2	$7(j_3)$	$2(j_1)$	$6(j_2)$
M_3	$5(j_2)$	$9(j_3)$	$3(j_1)$

Parent 2 :

machine	Opération(Job)		
M_1	$1(j_1)$	$8(j_3)$	$4(j_2)$
M_2	$7(j_3)$	$2(j_1)$	$6(j_2)$
M_3	$9(j_3)$	$3(j_1)$	$5(j_2)$

Parent 3 :

machine	Opération(Job)		
M_1	$1(j_1)$	$4(j_2)$	$8(j_3)$
M_2	$7(j_3)$	$2(j_1)$	$6(j_2)$
M_3	$3(j_1)$	$5(j_2)$	$9(j_3)$

Table 4.4le croisement de l'exemple précédent..

Enfant :

machine	Opération(Job)		
M_1	$1(j_1)$	$4(j_2)$	$8(j_3)$
M_2	$7(j_3)$	$2(j_1)$	$6(j_2)$
M_3	$5(j_2)$	$9(j_3)$	$3(j_1)$

Table 4.5 Le résultat de croisement.

L'algorithme de génération de croisement comme suite :

```
Début
    T =1
    Sélectionner aléatoire p_1 , p_2 , p_3
    S_1 , S_2 , S_3 //sont des Groupes des opérations de chaque parent .
    2 : trouver b' = min_Ok{b_k}//la machine M' en quelle Ok avec b'.
    Choisir un opération O_j a S_t
    P_t+1 ← O_j + P_t
    Supprimer O_j à S_t
    S_t+1 ← O_j + S_t
    T  ← T+1
        Si ( S_t != {}) Alors
          Répéter 2
           Sinom
               Exite()
           Finsi
    Résultat est la matrice S_t+1 de chaque enfants
    Tester la réalisabilité
        Si ( S_t+1 = 1) Alors
          Acceptés
          Sinom
              Refuser
        Finsi
Fin
```

Algorithme 4.3 L'algorithme de génération de croisement .

3.5 La sélection

Étape 1 : sélectionner l'individu qui possède la meilleur Fitness.

Étape 2 : construire la solution intermédiaire $p'(t)$:

- Faire la mutation à p(t) on obtient $p_1(t)$
- Faire la mutation à p(t) on obtient $p_2(t)$

$p'(t) = p(t) \cup p_1(t) \cup p_2(t)$

Étape 3 : sélectionner (n-1) individus aléatoirement (roulette).

L'algorithme de génération de sélection comme suite :

```
Début
    Pour i=1 à n faire
      Générer un individu in
      Calcule Fitness f( in )
      Calcule Makespan ( in )
      Rangement
      calcule probabilité_s p_i
      Générer un nomber alitoire X //X∈[0,1].
        Si X < p_i alors
          sélectionner l'individu
        Finsi
    Finpour
Fin
```

Algorithme 4.4 L'algorithme de génération de sélection .

3.6 Evolution

L'algorithme de génération d'évolution comme suite :

```
Début
      t=0
      Initialiser p(t)
      Evaluer p(t)
      Calcule l'évolution  Eval( p(t) )
     Tanque    (condition d'arie =1) faire
        Selection()
        T ← T + 1
       Selectionner p(t) de p(t-1)
       Evaluer p(t)
       Calcule l'evolution Eval( p(t) )
          Si  (Eval( p(t-1) ≥ Eval( p(t) )
               T = T - 1
          Finsi
      Fintanque

Fin
```

Algorithme 4.5 L'algorithme de génération d'évolution .

3.7 Parallélisme

On utilisant le modèle Master-Slave comme suite :

```
Début
  Initialiser les threades
  Slaves :auto-intialisation
  t=0
  Initialiser p(t)
  Evaluer p(t)
  Calcule l'évolution  Eval( p(t) )
   Tanque   (condition d'arie =1) faire
      Selection()
      T ← T + 1
     Selectionner p(t) de p(t-1)

    Fintanque
     Slaves : Selectionner la solution optimale et envoyer vers le Mastre
     Master :choisir la solution optimale parmi les solution qui recevoir
Fin
```

Algorithme 4.6L'algorithme Parallélisme de modèle Master-Slave.

CHAPITER 5 : RESULTAT ET SYNTHESE

1. Introduction

Nous fissent la présentation de notre travaille pour bien explication nous commençons par la présentation de martiale de développement apes ca nous expliquant les logiciels qui nous utilisons, enfin nous testons notre programme avec des paramètres déférent.

2. Environnement matériel

Nous avons utilisé comme environnement matériel un ordinateurs TOSHIPA possède les caractéristique suivant :

- Processeur : Intel® Core™ i3-4005U CPU @ 1.70GHz
- RAM : 4.00 Go
- Type De System : Système d'exploitation 64 bits.

3. Environnement logiciel

Plateforme utilisé est Windows 8 avec l'aide du programme suivant :

3.1 VMWork station

Nous utilisons ce programme pour créer plusieurs des ordinateur possèdent des caractéristique déférent pour tester le parallélisme de notre programme.

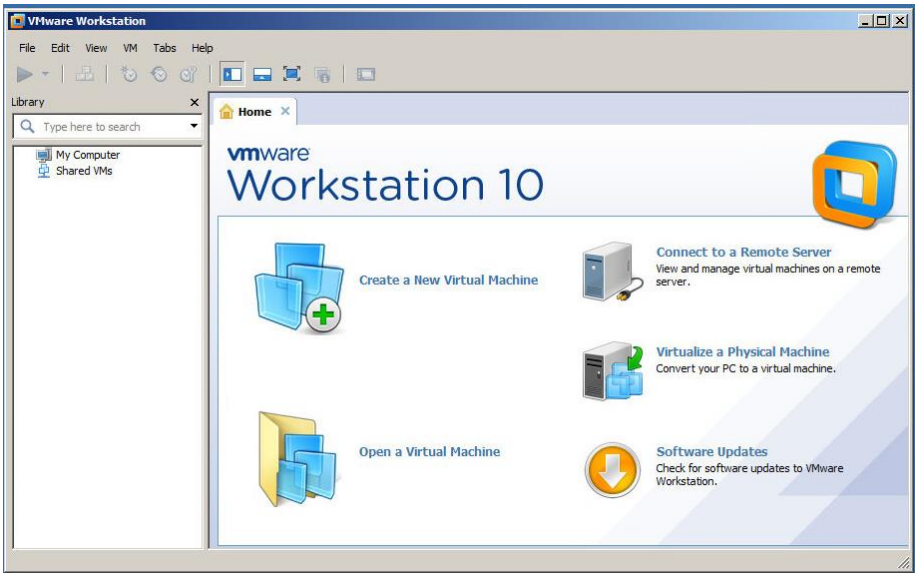

Figure 5.1 L'ogicielVMWork station.

3.2 Visual studio

Ce programme est l'enivrement de dévalement avec langage C#.

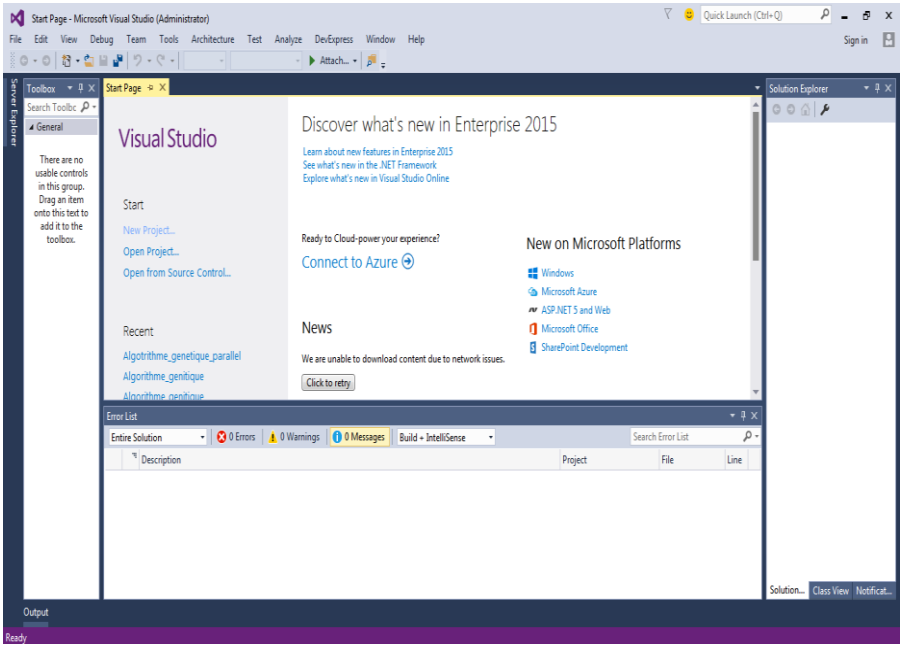

Figure 5.2 L'ogiciel Visual studio.

4. L'interface générale du notre programme

En notre programme on peut créer le nombre des jobs et le nombre des machines et aussi le nombre des processus et alors n'oublier pas les paramètres d'AG comme la population et la génération, enfin on va voir la résulta avec clique exécuter .Pour effectuer le parallélisme on clique sur plusieurs processus on peut voir la déférence entre séquentiel et parallèle a partir le clique sur le graphe de comparaison.

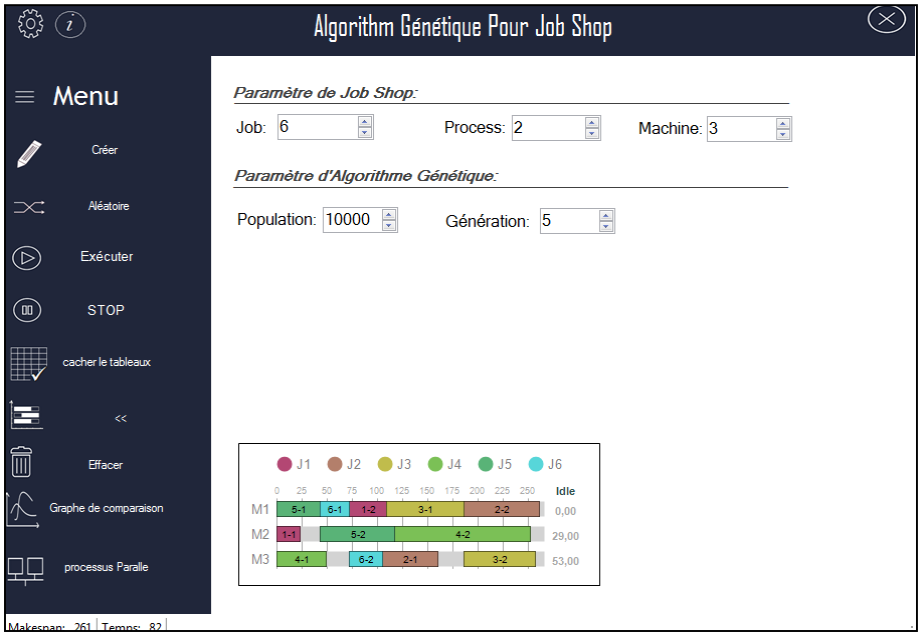

Figure 5.3 L'interface générale du notre programme.

5. Exemples en notre implémentation

- **Exemple 1 :**

On essaye faire l'exemple suivant dans les deux cas (séquentiel et parallèle)

	M1	**M2**	**M3**
Job 1	1	3	2
Job 2	3	1	2
Job 3	1	3	2

Table 5.1 Exemple 1 en notre implémentation.

- **En séquentiel :**

  Le Makespane = 125 et le diagramme de Gant est :

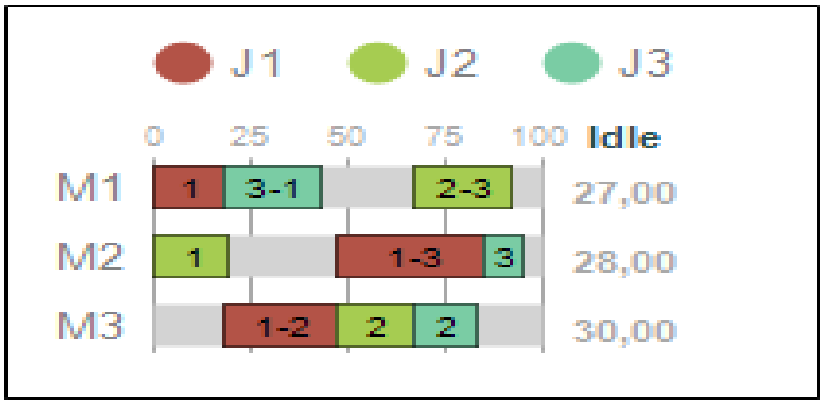

Figure 5.4 le diagramme de Gant .

- **En parallèle:** Le Makespane = 95 et le diagramme de comparaison entre le séquentiel et parallèle est comme suite :

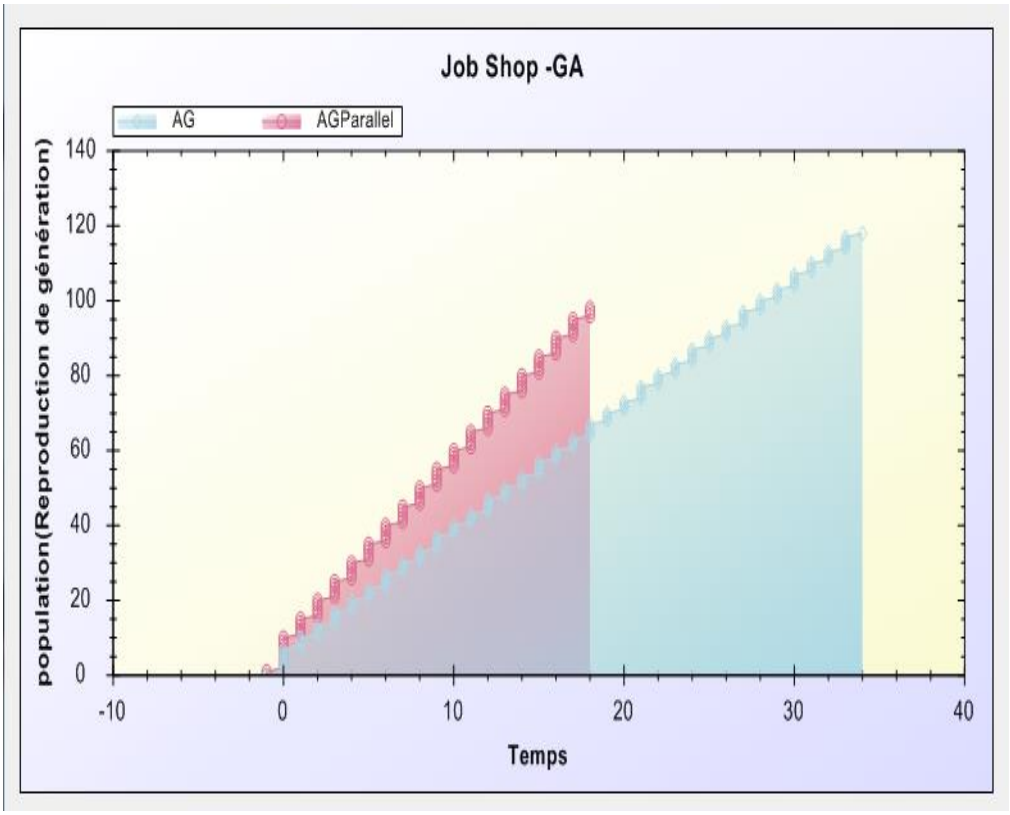

Figure 5.5 le diagramme de Gant (de comparaison entre le séquentiel et parallèle).

➢ **Exemple 2 :**

	M1	**M2**	**M3**	**M4**	**M5**
Job 1	1	3	2	3	4
Job 2	2	3	1	4	2
Job 3	1	4	3	2	1
Job 4	2	4	1	3	4
Job 5	4	3	1	2	2

Table 5.2 Exemple 2 en notre implémentation.

- **En séquentiel :**Makespane =242 et le diagramme de Gant est suivant :

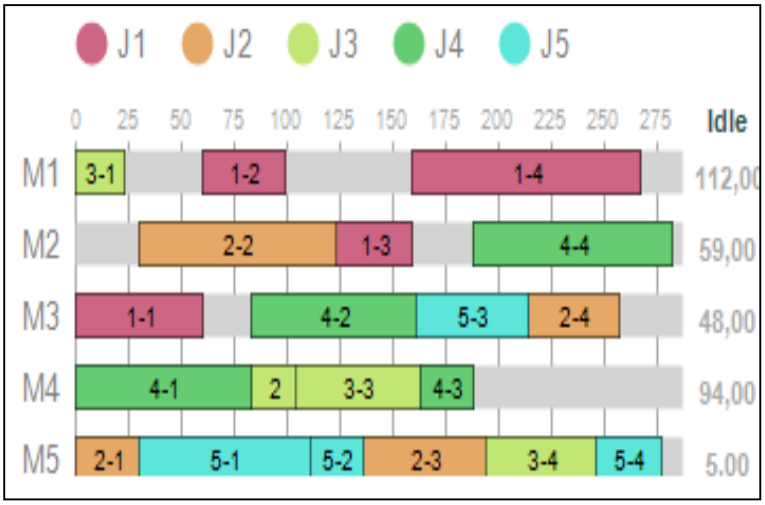

Figure 5.6 le diagramme de Gant (en séquentiel).

- **En parallèle :**Makespane=122 La graphe de comparaison est :

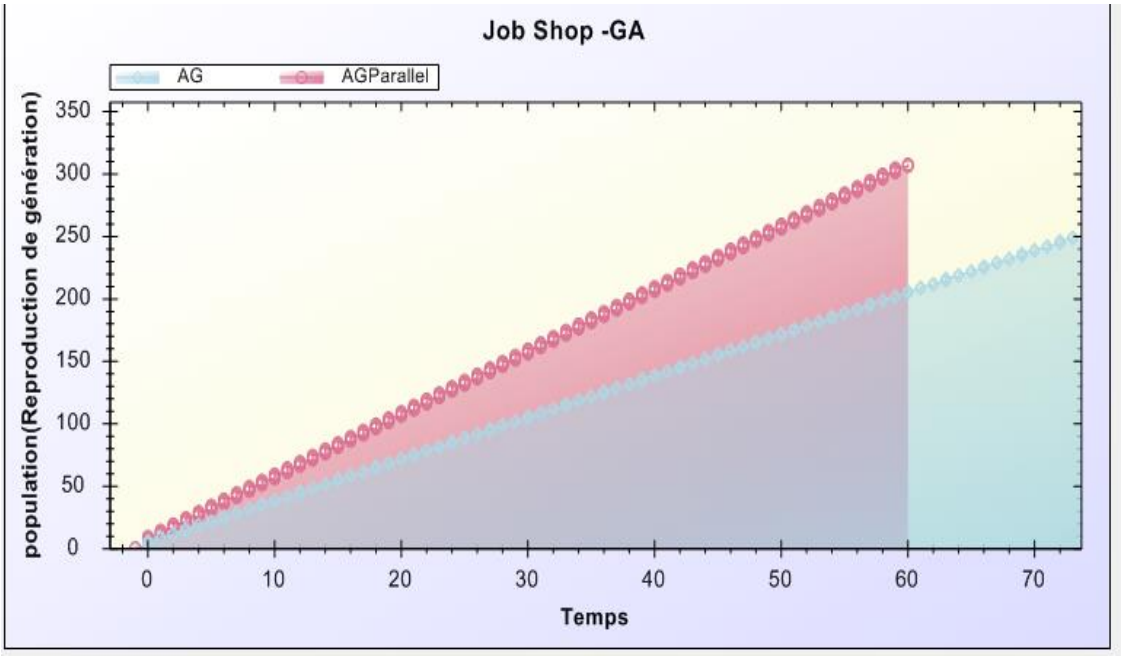

Figure 5.7 le diagramme de Gant (de comparaison entre le séquentiel et parallèle).

Apres la comparaison entre les exemples on a trouves toujours que le parallélisme est mieux que la séquentiel mais maintenant on comparais entre le nombre des processeurs en le parallélisme et c'est qu'est le mieux donc on va utiliser le programme VMWork pour créer des systèmes possèdent des caractéristiques déférent comme le tableaux suivant :

N(job)	N(machine)	Temps(séquentiel)	Temps(parallèle)		
			Processeur(i3)	Processeur(i5)	Processeur(i7)
3	3	62	41	36	28
5	4	112	87	40	34
9	8	590	334	148	97

Table 5.3comparais entre le nombre des processeurs en le parallélisme.

La présentation graphique de chaque cas de tableaux est ce dessous premièrement en le cas de job=3 et machin=3 comme suite :

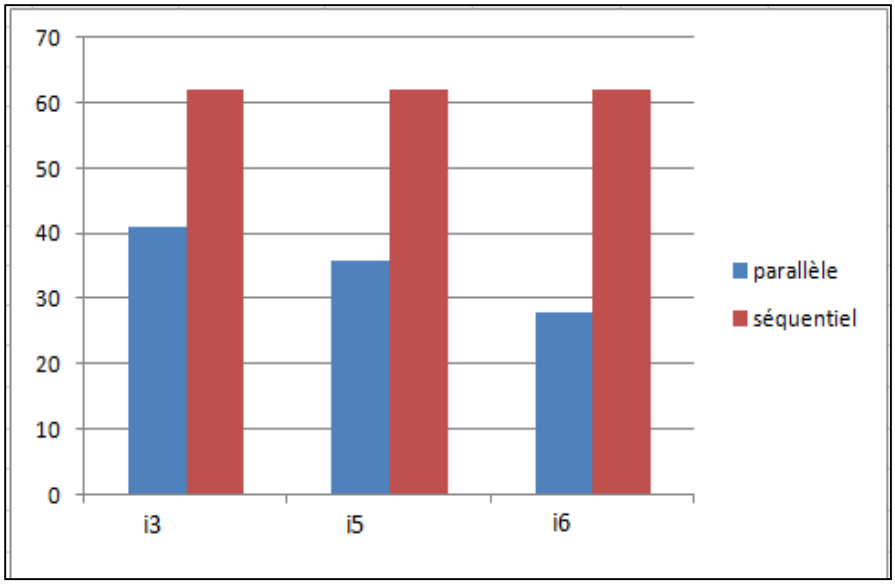

Figure 5.8 graphique de cas J=3 /M=3.

Dexiemment en le cas de job=5 et machin=4 est suivant :

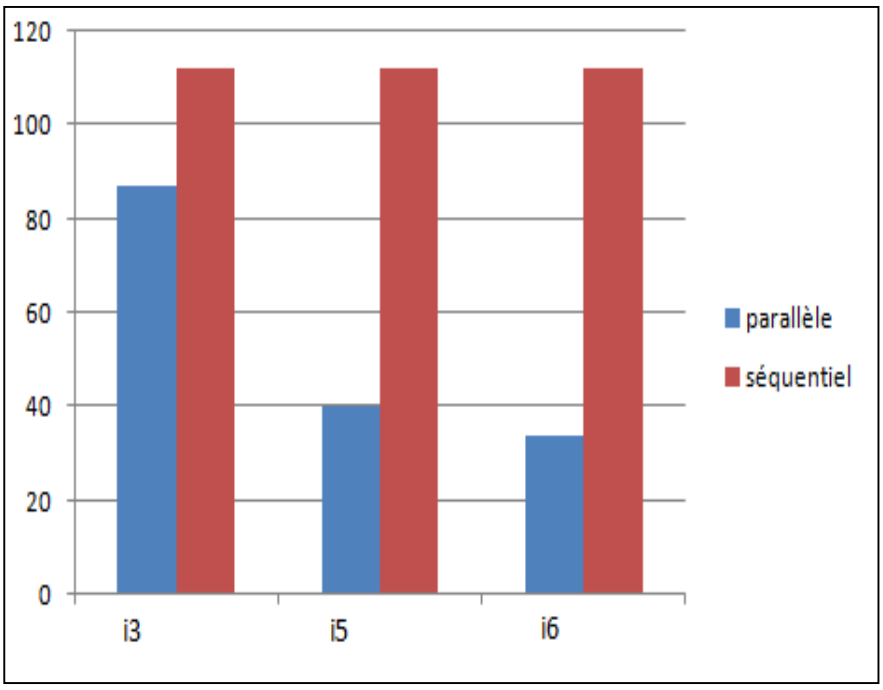

Figure 5.9 graphique de cas J=5 /M=4.

Et enfin en le cas de job=9 et machin=8 est comme suivant :

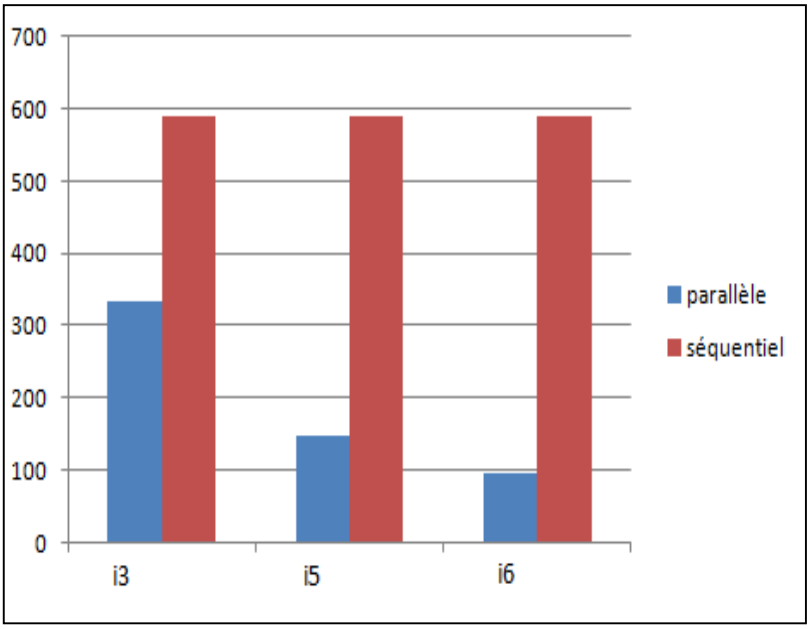

Figure 5.10 graphique de cas J=9 /M=8.

Donc a partir l'application de comparaison, nous constatons que les Algorithme génétique parallèle sont meilleurs que les Algorithme génétique séquentiel, en plus dans les algorithme génétique parallèle nous trouvons quand le nombre des processeurs est grand va nos donne le meilleur résultat.

6. Conclusion

Dans ce chapitre nous avons présenté notre implémentation, ensuite nous avons présenté l'environnement matériel et logiciel de travail réaliser, après ca nous avons posé notre étude comparative entre l'algorithme génétique séquentiel et parallèle après nous avons comparis entre le nombre des processeurs en le parallélisme, enfin nous avons discuté sur les résultats obtenus.

Conclusion générale

Le bute principale de notre travail consistait à appliquer les algorithmes génétique pour résoudre le problème d'ordonnancement de job shop qui est un problème extrêmement complexe. Il est classé parmi les problèmes combinatoires.

Nous avons créer un application pour ressouder ce problème avec l'utilisation des algorithmes génétique , et plus nous ajoutons la parallélisme pour comparer entre les algorithme génétique séquentiel et parallèle ,aussi nous essayons voire c'est qui le mieux pour minimiser le Makespane .enfin nous déduisons que les algorithmes génétiques parallèle sont les mieux pour minimiser le Makespane rapidement.

Le premier chapitre, nous avons étudié l'ordonnancement en cas général, aussi nous présentons et leur modèles.

Dans le deuxième chapitre, nous avons étudie les algorithmes génétiques et leurs principes, aussi leurs mécanisme de fonctionnement.

Dans le troisième chapitre, nous avons présenté quoi les autre fiassent en notre thème.

Dans le quatrième chapitre, nous avons expliqué notre aidée pour ressouder le problème de job shop avec l'utilisation des algorithmes génétiques.

Enfin, nous avons présenté notre application et quel que explication pour le utilisé, aussi les paramètres de notre application.

BIBLIOGRAPHIES

[1]A.S. Jain et S. Meeran, Deterministic job-shop scheduling : past, present and future, EuropeanJournal of OperationalResearch, vol. 113, pp. 390-434, 1999.

[2]B.Giffler et G.L.Thompson, Algorithms for solving production SchedulingProblems,OpérationsResearch, (1960) 8(4), 487-503.

[3] B. Roy, Algèbre moderne et théorie des graphes, volume 2, Editions Dunod, Paris, 1970.

[4] B. Roy et D. Bouysson, Aide multi-critèresà ladécision : Méthodes et cas . Collection Gestion Série : Production et technologie quantitatives appliquées à la gestion. Edition Economica, Paris, 1993.

[5]D.lamy, Méthodes et outils pour l'ordonnancement d'ateliers avec prise en compte des con-traintes additionnelles : énergétiques et environnementales,Thèse de Doctorat, Université Clermont Auvergne,Français, 2017.

[6] F.GLOVER, Future paths for integer programming and artificial intelligence, Computers & Operations Research 13 (1986), 533-549.

[7]G. Bel et J-B. Cavaillé, Ordonnancement de la production . Editions Hermès, Paris, 2001.

[8]Gotha, Les problèmes d'ordonnancement. RAIRO-Recherche Opérationnelle, vol. 27, n° 1, pp. 77-150, 1993.

[9] H.Boukef, Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharma-ceutiques : optimisation par algorithmes génétiques et essaims particulaires, Thèse de Doctorat, Université de Tunis El Manar, Tunis, 2009.

[10] J.Carlier et P. Chrétienne, Problèmes d'ordonnancement, Edition Masson, Paris, 1988.

[11] J. Carlier, P. Chrétienne et C. Girault, ModellingschedulingproblemswithPetri nets. Advanced studies in Petrinets . Lecture notes in Computer Science, Springer Verlag, Paris, 1984.

[12] J.Kaabi-Harrath, Contribution à l'ordonnancement des activités de maintenance dans les systèmes de production, Thèse de Doctorat, Université de Franche-Comté, França, 2004.

[13] M.larabi, Le problème de job-shop avec transport : modélisation et optimisation, Thèse de Doctorat, Université Blaise Pascal - Clermont-Ferrand II, Français, 2010.

[14] M. Meziane, Proposition d'une approche d'ordonnancement pour les ateliers de type Job Shop Flexible, Thèse de Doctorat, Université d'Oran, Algérie, 2018.

[15] M.TALEB, Parallélisassion d'un algorithme génétique pour le problème d'ordonnancement sur machine unique avec temps de réglages dépendants de la séquence,Université du Québec à Chicoutimi ,2008.

[16]M.Yildizoglu, Présentation des algorithmes génétiques et de leurs applications en économie, Article, University of Bordeaux,February 2004.

[17] N.Liouane , Contribution à l'élaboration d'un outil d'aide à la décision pour l'ordonnancement de production en environnement incertain, Thèse de doctorat, Université Lillel Sciences et Technologies, Lille, 1998.

[18]T.FEO et M.RESENDE, Greedy Randomized Adaptive Search Procedures, Journal of Global Optimization 6, 2 (1995), 109-133.

[19]T. Nicholson, Optimization in industry . Editions LongmannPress, Londres, 1971.

Les sites web

[a] https://fr.wikipedia.org/wiki/Ordonnancement_d%27atelier.

[b] http://igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html.

[c] https://www.researchgate.net/figure/Diagramme-de-Gantt-de-la-solution-du-probleme.