الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

Une approche intelligente pour un problème d'optimisation multicritère

Encadré Par :

Dr BENHAMZA KARIMA

Présenté par :

MAHBOUBI THEMER

Juillet 2019

Dédicaces

Je dédie ce travail :

***À ma très chère mère wahida** : Aucune dédicace ne saurait être assez éloquente pour exprimer ce que tu mérites pour tous les sacrifices que tu n'as cessé de me donner depuis ma naissance, durant mon enfance et même à l'âge adulte.*

À *mon père Abdelghani :* *Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous, rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation.*

À tous les membres de ma famille : *Veuillez trouver dans ce modeste travail l'expression de mon affection, et surtout ma sœurs : SELMA et mes frères :mokim et raid , les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous.*

Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

À *mes amis :* *Surtout mes frères: Chouaib Jalel, Midou,Wassim et les autres, Sans ton aide, tes conseils et tes encouragements ce travail n'aurait vu le jour.*

À tous mes chères ami (e)s et mes chers collègues.

THEMER

Remerciements

Nous remercions en premier lieu ALLAH qui nous a éclairé notre chemin pour achever ce modeste travail.

Je tiens à remercier sincèrement mon encadreur Dr Benhamza karima pour son encadrement, sa gentillesse et ses orientations durant tout le semestre.

Je remercie également tous ceux qui ont, de près ou de loin, aidé à rendre ce travail possible, que ce soit par des informations, des idées ou par des encouragements.

Et bien sûr nous remercions ma famille : mes parents, mes frères et sœur, pour leurs soutiens et aides.

À tous mes collègues et mes amis

Résumé

Les problèmes décisionnels introduisent le plus souvent plusieurs critères de nature complexe. La résolution de tels problèmes relève du domaine de l'optimisation combinatoire. Plusieurs méthodes existent et peuvent être classées dans deux grandes familles : les méthodes exactes et les méthodes approchées.

Dans ce travail, on s'intéresse à l'application d'une méthode métaheuristique au problème d'ordonnancement. Ce problème décisionnel est de nature complexe. Notre choix s'est orienté vers l'application de l'algorithme d'optimisation par essaim particulaire. L'intérêt est de montrer l'efficacité de cette métaheuristique dans la résolution de ce problème.

Mots clés : Décision multicritère, Optimisation, Algorithme d'optimisation par essaim, Problème d'ordonnancement.

Table des matières

❖TABLE DES FIGURES

❖LISTE DES TABLEAUX

❖LISTE DES ABREVIATION

PSO	Particule Swarm Optimization
TS	Tabu Search
JVM	JAVA Virtual Machine
API	Application Programming interface
GUI	Graphical User Interface
JFC	Java Foundation Classe

❖INTRODUCTION GENERALE

Aujourd'hui les problèmes d'optimisation occupent une place prépondérante dans les études des chercheurs et des ingénieurs. Pour bien comprendre un problème d'optimisation il faut d'abord identifier les différentes fonctions objectives à optimiser, l'espace de recherche et les variables.

Pour résoudre ces problèmes les chercheurs ont développés plusieurs approches de résolution pour donne des solutions quasi-optimale. Parmi Ces approches d'optimisation on cite les méthodes exactes, les heuristiques et les métaheuristiques telle que l'optimisation par essaim particulaire, la recherche Tabu, les algorithmes génétiques... etc.

Néanmoins, pour appliquer une méthode de résolution a un problème d'optimisation donné se base sur deux facteurs principaux la qualité de solution et le temps de calcul.

Dans ce travail, on s'intéresse aux problèmes d'optimisation multicritère qui sont de complexité grandissante tel que le problème d'ordonnancement qui est classés parmi les problèmes difficiles.

On décrit un problème d'ordonnancement comme suit : l'affectation de n job a des ressource physique (machines). Chaque job consiste en une séquence des taches, selon un ordre spécifié. Chaque tache doit être exécutée sur une machine donnée pendant un temps bien déterminé.

La résolution de problème d'ordonnancement en utilisant les méthodes exactes révèlent des faiblesses en temps et espace mémoire parfois inapplicable à cause de sa nature complexe.
Pour traiter ce problème, il est nécessaire d'utilisé une méthode de résolution approché.

Ce travail s'inscrit dans le cadre d'utilisation des métaheuristiques pour la résolution de problèmes d'ordonnancement avec minimisation de temps d'achèvement d'exécution (Makespan). Notre contribution consiste à l'adaptation, la programmation de l'algorithme d'optimisation par essaim particulaire en anglais particule swarm optimisation (PSO).

Combiné avec une recherche locale. L'intérêt est d'étudier et de montrer l'efficacité de cette méthode a la résolution de tel problème.

Ce mémoire et devisé en deux parties :

La partie 1 : Etat de l'art cette partie présente les problèmes d'optimisation et les méthodes résolution. Exposé en deux chapitres.

La partie 2 : Conception et implémentation : Cette partie expose la modélisation et a discussion des résultats obtenus.

Une conclusion générale est donnée à la fin de ce mémoire pour souligner les résultats obtenus et présenter des perspectives de recherche.

Chapitre1 : Les problèmes d’optimisation

1. Introduction :

Vu le rythme croissant de la recherche technologique, les chercheurs dans tous les domaines sont confrontés à des problèmes de grande taille .Généralement, ils sont mis sous la forme d'un problème à résoudre à travers des méthodes classiques .Cependant il est nécessaire de déterminer une méthode de résolution compatible.

Plusieurs méthodes d'optimisation qui visent à maximiser ou minimiser une seule ou plusieurs fonctions objectives existent. Il y'a deux classes des méthodes, les méthodes exactes et les méthodes approchées

Dans ce chapitre, on va présentera les concepts de base lie à ce domaine, et on discutera les limites de chaque approche.

2. Problèmes d'optimisation :

Quand on confronte un problème a un ou plusieurs objectifs a optimisées la prise de décision sera un processus difficile. On parle alors de problèmes d'optimisation

La définition de 'Th. Paschos ' d'un problème d'optimisation est la suivante :

> *« L'optimisation c'est l'art de comprendre un problème réel, de pouvoir le transformer en un modèle mathématique que l'on peut étudier afin d'en extraire les propriétés structurelles et de caractériser les solutions du problème »* [1].

Un problème d'optimisation est défini par un espace d'état, une ou plusieurs fonctions objective et un ensemble de contraintes définit par [2] comme suit :

Espace d'état : il représente les domaines de définition des variables du problème.

Variables du problème : elles peuvent être des données qualitatives ou quantitatives et de nature diverse (réelle, entière, booléenne, etc.)

Une fonction objective : ou la fonction cout représente le but à atteindre pour le décideur (minimisation, maximisation).

L'ensemble de contraintes : c'est l'ensemble des conditions sur l'espace d'état que les variables doivent satisfaire afin de limiter l'espace de recherche

Une méthode d'optimisation : c'est la recherche d'une valeur optimale ou l'optimum, qui représente le point ou l'ensemble des points de l'espace d'états qui satisfait mieux un ou

plusieurs contraintes.

Dans ce qui se suit nous allons présenter la classification des problèmes d'optimisation**.**

3. Classifications des problèmes d'optimisation :

On trouve plusieurs méthodes de classification devisées selon des facteurs bien précise.

Le schéma ci-dessous présente une représentation globale de classifications problèmes d'optimisation :

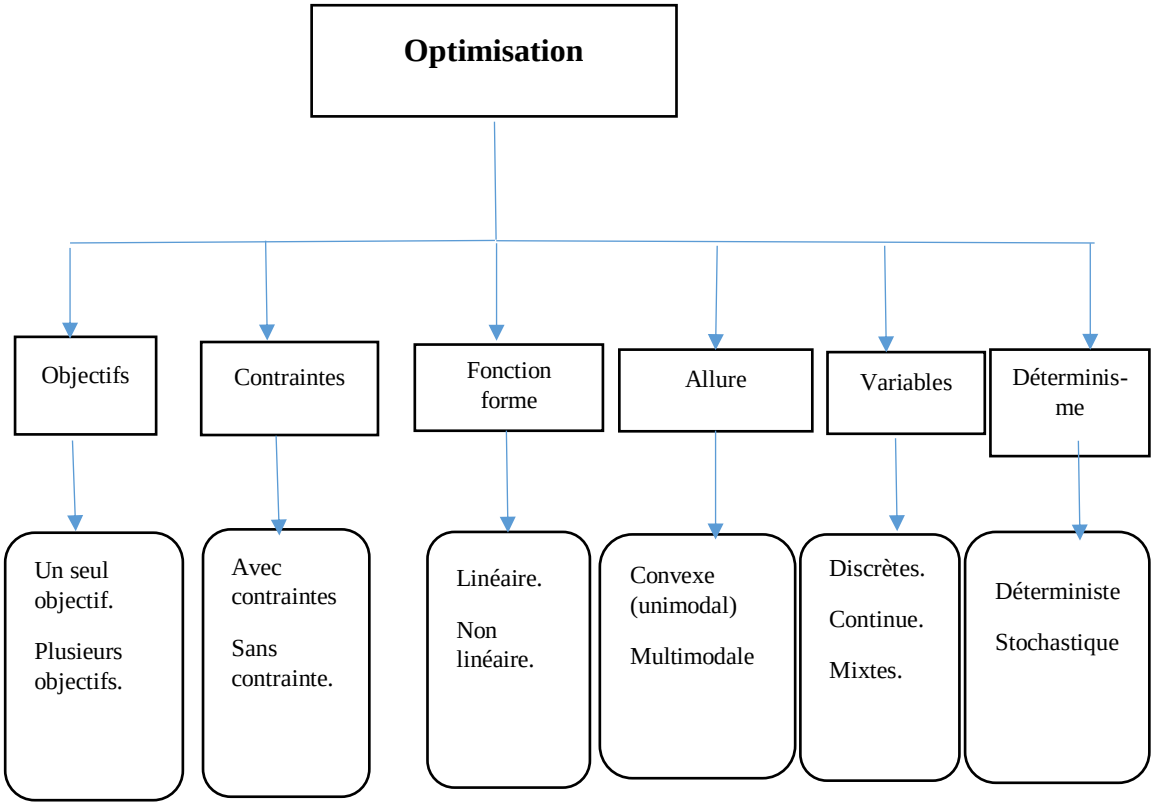


Figure1.1 : Classification des problèmes d'optimisation [3]

D'après cette figure plusieurs facteur sont prise en compte pour classifier les problèmes d'optimisation tel que la fonction forme les contrainte le type des variable et le nombre des objectifs ...etc.

Dans e qui suit on s'intéresse à la classification selon le nombre des objectifs à atteindre.

3.1. Les problèmes d'optimisation mono-objectifs :

Lorsque le processus d'optimisation est effectué sur un problème à objectif unique, une seule fonction est minimisée ou maximisée. Pour résoudre un problème a un

seul objectif il faut optimise en maximisant ou minimisant un critère donné [4]

Dans la plupart des cas, l'optimum découvert n'est pas unique il existe un ensemble de solution qui minimise ou maximise ce critère

Un problème a un seul objectif caractérisé par : [5]

Espace décisionnel : représente l'ensemble des solutions possible du problème. Cet ensemble est déterminé par des contraintes données.

L'espace de recherche *:* regroupe les valeurs que variables de décision peuvent être prises.

Espace réalisable : regroupe les valeurs des variables qui satisfaire les contraintes données.

3.2. Les problèmes d'optimisation Multi-objectifs :

La forme de problème mono-objectif était liée à un problème dans lequel nous cherchons un optimum pour une seule fonction objective[6].

Un problème d'optimisation multi-objectif aura deux objectifs ou plus impliquant Nombreuses variables et contraintes de décision.

Le but que nous voulons atteindre lors de la résolution du problème d'optimisation multi objectif est de minimiser ou maximiser fonctions de l'objectif.

Note : dans un problème d'optimisation multicritère, nous avons souvent objectif contradictoires. [7].

Un problème a un plusieurs objectifs est caractérisés par :[8]

3.2.1. La fonction objective :

Définit la caractéristique d'un système qui devrait être améliorée. Cette caractéristique est représentée par une équation mathématique qui dépend d'un certain nombre de quantités organisées dans un vecteur dénommé vecteur de variables de conception.

3.2.2. Les contraintes :

Les contraintes représentent des limitations quant à la solution du problème d'optimisation. Ces contraintes peuvent représenter des informations impliquant des variables de conception, des limitations physiques, des aspects de sécurité, environnementaux et économiques…etc.

4. Domaine d’application de l’optimisation multi-objectif :

L'application de l’optimisation multi-objectives dans différents domaines économiques tels optimisation le modèle bioéconomique de la pêche. [9]. L’auteur étudie le cas d'un modèle pour les pêcheries de la mer du Nord avec quatre objectifs à prendre en compte : maximiser les profits, maintenir des parts de quota relativement historiques entre les pays, maintenir les emplois dans l'industrie et minimiser les déchets.

Dans le domaine de la finance, identifier les tendances significatives de l'analyse technique dans la série chronologique financière, comme dans les travaux de.[10] niché de Pareto. Deux objectifs sont pris en compte, à savoir la qualité des correspondances.

Dans le domaine de la mécanique l’utilisation des échangeurs de chaleur. Pour minimiser le coût total de l'équipement, y compris l'investissement en capital et le montant des dépenses énergétiques annuelles. En même temps, la réduction de la longueur de l'échangeur de chaleur est également visée. Les algorithmes à objectifs multiples recherchent les valeurs optimales des variables de conception, telles que le diamètre extérieur du tube, le diamètre extérieur et l'espacement des déflecteurs.[11]

Dans le domaine d’ordonnancement l’utilisation de l’optimisation multicritère pour attribuer les taches aux ressources physiques en exploitant de façon optimale l’espace dans les ressources physiques et en minimisant le temps (makespan).

5. Conclusion :

Nous nous sommes concentrés dans ce chapitre sur les problèmes d’optimisation en exposant les concepts a mono-objectif et multi-objectifs d’optimisation. Dans le prochain chapitre nous présentons les méthodes de résolution d’un problème d’optimisation à savoir les méthodes exactes et les méthodes approchée.

Chapitre 2 : Les Méthodes de résolutions

1. Introduction :

Les problèmes technologiques à complexité grandissante sont souvent difficiles à résoudre (les problèmes à plusieurs objective).

Les méthodes de résolution d'un problème multi-objectif sont repartis en deux classes les méthodes exactes qui sont dédiées aux problèmes d'optimisation relativement petits. La deuxième classe regroupent les méthodes de résolution approchées qui donnent des solutions approximatives. Parmi ces méthodes on trouve les algorithmes bio-inspiré de plusieurs domaines (inspiré par des systèmes naturels).

Dans ce chapitre nous allons exposer les méthodes de résolution en soulignant les méthodes de résolutions approchées.

2. Les méthodes exactes :

Cette classe de méthodes de résolution est dédiée aux problèmes de taille raisonnable.

Il existe de nombreuses méthodes exactes telles que la famille de Branch et X (algorithme de Branch and Bound [12], algorithme de Branch and Cut [13], algorithme de Branch and Price [14]), la programmation linéaire, la programmation dynamique.

a. **Un algorithme de branche X** : utilise une stratégie de division et de conquête pour partitionner l'espace de la solution en sous-problèmes, puis optimise individuellement chaque sous-problème
b. **Programmation dynamique :**

C'est le résultat des travaux de Richard Bellman [15] dans les années cinquante. le méthode est efficace pour la résolution exactes des problèmes d'optimisation.

Cette approche est basée la récursivité pour décrire la valeur optimale du critère à une étape en fonction de sa valeur de l'étape précédente.

Les méthodes exactes sont souvent coûteuses en temps, donc on ne peut pas l'appliquées sur des problèmes de grande taille ou difficiles à résoudre [16].

Lorsque les instances deviennent trop grandes pour des méthodes exactes, on utilise souvent des méthodes heuristiques, notamment métaheuristiques que nous détaillerons dans la suite.

3. Les méthodes approchées :

3.1. Heuristiques :

Les méthodes heuristiques ont été introduites par G. Polya en 1945 sous la définition suivante :

Une heuristique est une technique (consistant en une règle ou un ensemble de règles) qui cherche (et trouve, espérons-le) de bonnes solutions à un coût de calcul raisonnable. Une heuristique est approximative dans le sens où elle fournit (espérons-le) une bonne solution pour relativement peu d'effort, mais elle ne garantit pas l'optimalité.[17].

Les méthodes heuristiques ont été proposées, pour déterminer une précision non parfaite mais une qualité satisfaisante d'approximations de solutions exactes. Ces méthodes étaient initialement basées essentiellement sur les connaissances et l'expérience des experts et visaient à explorer l'espace de recherche de manière particulièrement pratique.[18]

3.2. Métaheuristique :

Le terme métaheuristique a été proposé par Glover en 1990 pour définir une heuristique de haut niveau utilisée pour guider d'autres heuristiques en vue d'une meilleure évolution de l'espace de recherche [19].

D'après [19]"*Les métaheuristiques font référence à une stratégie maîtresse qui guide et modifie d'autres heuristiques afin de produire des solutions allant au-delà de celles qui sont normalement générées dans une quête d'optimalité locale*." [19].

Donc on peut dire qu'une métaheuristique est une stratégie de niveau supérieur qui guide une heuristique sous-jacente pour résoudre un problème donné [20].

3.2.1. Propriétés des métaheuristiques :

Les métaheuristiques et les heuristiques sont des techniques approximatives, non déterministes et stratégies qui guident le processus de recherche pour le but de l'exploration et l'exploitation de l'espace de recherche afin de trouver des solutions proches aux solutions optimales.

Généralement les métaheuristiques ne sont pas dédiées à un problème spécifique [21].

3.2.2. Classification des métaheuristiques :

Pour classifier les métaheuristiques il faut différencier entre les métaheuristiques avec une seule solution et avec une population des solutions.

3.2.2.1. Métaheuristiques à base de population des solutions :

Les métaheuristiques a population de solution appelé aussi les méthodes évolutionnaires, consiste à travaillent sur une population de solutions et non pas sur une solution unique. Le principe de ces méthodes c'est de combiner ensemble des solutions entre elles pour en former de nouvelles en essayant d'hériter des bonnes caractéristiques des solutions parents. Ce processus est répété jusqu'à arriver à un critère d'arrêt. On peut citer comme exemple les algorithmes génétiques, les algorithmes de fourmis, les algorithmes de d'optimisation par essaim [22].

a. Optimisations par colonie de fourmis :

- **Source d'inspiration :** Le comportement des insectes (fourmis, abeilles et termites) en 1989 par les expériences de [23].
- **Principe de fonctionnement :**

Les fourmis artificielles évoluent d'un état à un autre dans monde discret. Une mémoire est associée à chaque fourmis stock l'historique de ses actions .la quantité de phéromones déposée par les fourmis réelles représente une valeur stockée dans une variable pour les fourmis artificielles et dépend du problème à traiter.

Les valeurs stockées dans les variables de l'ensemble des fourmis artificielle regroupe la quantité de phénomène déposé par les fourmis réel [24].

- **Principale caractéristique :**

La réalisation de recherche efficace même avec l'élimination de certain individu.

Applicable pour défèrent problème tel que le problème de voyageur de commerce et l'affectation dynamique [3].

b. Les algorithmes génétiques :

- **Source d'inspiration :**

Sont des méthodes inspirées des mécanismes de l'évolution naturelle. Développé par les expériences de **[**25] [26].

- **Principe de fonctionnement :**

La génération d'une population initiale ce fait de façon aléatoire et les populations seront générées en appliquant les diffèrent technique jusqu'à arriver à la condition d'arrêt [25] [26].

- **Principale caractéristique :**

Il est simple et facile à implémenter Il a un degré de flexibilité pour gérer des différents problèmes multi-objectifs [3].

c. Algorithme d'optimisation par essaim particulaire :

- **Source d'inspiration :**

Comportement social des animaux évoluant en essaim (les poissons, les oiseaux...etc.). Introduite par [27].

- **Principe de fonctionnement :**

Une population initiale (essaim) est répartie de façon aléatoire sur l'espace de recherche, chaque particule est caractérisée par une position x et une vitesse v.

Chaque particule est évaluée et sauvegardée en mémoire sa meilleure position passé et informée ces voisins de sa meilleur position connue et à partir de la chaque particule choisit la meilleure des meilleures positions dont elle a connaissance et modifie sa vitesse et son déplacement en fonction de cette information [27].

- **Principale caractéristique :**

Algorithme a grande performance, robuste et Simple à programmer, la recherche des solutions est basée sur la vitesse de particule. La communication entre les particules afin de trouver la meilleure position des règles de déplacement très simples [3].

La méthode PSO se base sur la technique de mécanisme de déplacement des particules

3.2.2.2. Métaheuristiques à une seule solution solutions :

Appelé aussi méthodes de recherche locale et comme leur nom indique sont des méthodes qui cherchent de façon itérative d'améliorer une solution unique. Dans cette catégorie, on peut citer la méthode de recherche tabou (Tabu Search TS) et le Recuit Simulé [3].

a. **Recuit Simulé :**

- **Source d'inspiration :**

Inspiré du principe de la thermodynamique et a été proposé par Kirkpatrick, Gelatt et Vecchiet en 1983[28].

- **Principe de fonctionnement :**

Le recuit simulé est une méthode de recherche local reposent sur un paramètre qui représente la température T qui dégrade de façon lente avec le nombre d'itérations et détermine la probabilité qu'une dégradation de la solution courante est acceptée [28].

- **Principale caractéristique :**

Cette méthode est simple a programmé et donne des bonnes solutions [3].

b. La méthode tabu :

- **Source d'inspiration :**

Phénomène d'aspiration proposé par [29].

- **Principe de fonctionnement :**

La recherche Tabou est une méthode de recherche locale. Pour chaque itération, l'algorithme test le voisinage de point x et choisir le meilleur voisin x' tel que $\forall\, x'' \in V(x)$, $f(x')$ est meilleur que $f(x'')$ et $x' \notin$ à la liste taboue, sauf si $f(x')$ est meilleur que $f(x)$. On note que x' est retenu même si $f(x')$ n'est pas meilleur que $f(x)$[29].

- **Principale caractéristique :**

Sauvegarder l'information pertinente de la recherche précède dans une mémoire dynamique dite liste tabu.

L'utilisation de technique de d'intensification pour l'exploitation de situations de bonne qualité et la technique de La diversification pour diriger l'algorithme vers des espaces de recherche non explorées [2].

Le schéma ci-dessous résume les méthodes d'optimisation et ces classes :

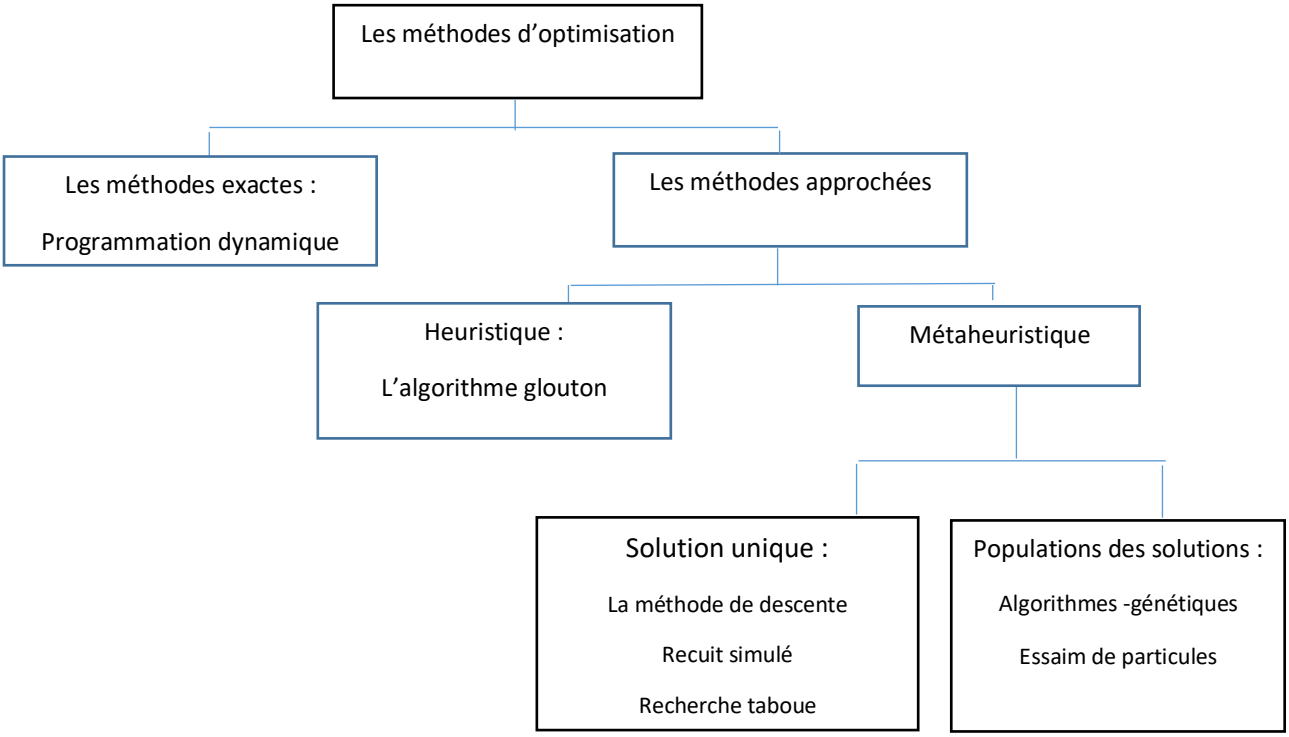


Figure2.1 : Schéma de classification des méthodes d'optimisation [3].

4. Conclusion :

Nous avons souligné dans ce chapitre les déférent approches de résolution appliquée aux problèmes d'optimisation. Nous avons présenté les méthodes exactes avec leurs limites pour les problèmes de grandes dimensions.

Nous avons aussi souligné l'importance et l'efficacité des méthodes approchées.

Dans la prochaine partie, nous allons présenter la conception et l'implémentation de notre méthode.

Chapitre3 : Conception

1. Introduction

Dans ce chapitre, on va réaliser une étude conceptuelle sur l'application de la métaheuristique d'optimisation par essaim particulaire (PSO) pour la résolution du problème d'ordonnancement. L'algorithme PSO proposé par Kennedy et Eberhart en 1995 [27] est basé sur l'observation du comportement social des animaux tels que les oiseaux, les poissons. Ce chapitre présente le choix de l'algorithme PSO, le problème d'ordonnancement et les travaux connexes. On expose aussi le modèle PSO combiné à une recherche locale appliqué à ce problème.

2. Algorithme PSO :

L'algorithme PSO est indépendant du problème, ce qui signifie que peu de connaissances spécifiques relatives à un problème donné sont nécessaires. Tout ce que nous devons savoir, c'est l'évaluation de la condition physique de chaque solution [30].

Le PSO utilise une stratégie de recherche basée sur une population. Le plus important dans cet algorithme sont les formules mathématiques à mettre à jour avec les particules de l'ensemble de l'essaim. Par conséquent, PSO est facile à manipuler et à utiliser [31].

2.1.Principe de l'algorithme PSO :

- Supposons que l'espace de recherche soit en D-dimension, puis que la i -ème particule de l'essaim puisse être représentée par un vecteur en D-dimension.
- L'initialisation d'ensemble de particules que nous appelons « essaims » de manière aléatoire.
- Chaque particule peut changer de position dans l'espace de de recherche, exactement comme un oiseau en vol qui cherche de la nourriture dans le ciel.
- Une particule d'un essaim ajuste sa nouvelle vitesse de déplacement en fonction de sa meilleure expérience, de la meilleure expérience de toutes les particules de l'essaim et de la vitesse de déplacement précédente.
- Ensuite, chaque particule se déplace vers une nouvelle position en fonction de la vitesse nouvellement générée et de sa position précédente.

2.2. Pseudo code de l'algorithme PSO

Le Pseudo code de l'algorithme PSO détaillé dans la figure suivante selon.

1. Initialisez un essaim de particules avec des positions et des vitesses aléatoires dans l'espace à problèmes de dimension D.

2. Pour chaque particule, évaluez la fonction d'optimisation souhaitée.

3. Comparez la valeur de remise en forme des particules avec les particules pbest. Si la valeur actuelle est meilleure que

pbest, puis définissez la valeur pbest égale à la valeur actuelle et la position pbest égale à la

position actuelle dans l'espace D-dimensionnel.

4. Comparez la valeur d'évaluation de la condition physique avec la meilleure forme physique obtenue par les essaims jusqu'à présent. Si la valeur actuelle est meilleure que gbest, réinitialisez gbest à la valeur de mise en forme des particules actuelle.

5. Modifiez la vitesse et la position de la particule selon les équations (1) et (2).

respectivement.

6. Revenez à l'étape (2) jusqu'à ce qu'un critère de terminaison soit rempli, généralement un critère suffisamment bon.

fitness ou un nombre spécifié de générations.

Figure3.1 : Pseudo code PSO [32].

2.3.Paramètres de l'algorithme:

Un essaim est composé d'un ensemble de particules qui représentent chacune une solution dans un espace de recherche de dimension D. Chaque particule i de l'essaim est modélisée par son vecteur de position courante xi = (xi1, xi2,…, xiD) et son vecteur vitesse de déplacement vi= (vi1, vi2,…, viD)[3].

La qualité de sa position est déterminée par la valeur de la fonction objective f en ce point. Chaque particule mémorise la meilleure position retrouvée, que l'on note P_{bestid}= (P_{besti1}, P_{besti2},…, P_{bestiD}), ainsi que la meilleure position atteinte par ses particules voisins l'essaim, notée g_{bestid} = (g_{best1}, g_{best2},…, g_{best3},…, g_{bestiD}).

V_{id} est appelée la vitesse pour la particule i

X_{id} représente la position de particule i

Pid ou **Pbest** sont également appelé meilleure solution locale, représente les particules avec la meilleure position.

Pgd ou **Gbest** qui s'appelle également (meilleure solution globale), représente la meilleure position parmi toutes les particules de l'essaim

W=est le poids d'inertie. Il régule le compromis entre l'exploration globale et les capacités d'exploitation locales de l'essaim

C1 et C2 =représentent le poids de l'accélération stochastique pour chaque particule.

Mise à jour de la vitesse et la position :

À chaque itération, chaque particule met à jour sa vitesse et sa position en fonction de sa meilleure position X, de sa vitesse actuelle V et de quelques informations sur ses voisines.

$$V(t+1)=w(t)* V(t) +c1random()(p_{best} x_t)+c2random()(g_{best}-x_t)\ldots\ldots..(1)$$

Inertie | Influence personnelle | Influence sociale

$$X_{(t+1)}= x_t+ Vt\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

Le calcule de vitesse de particule se fait selon équation (1) en calculant trois élément bien détaillés dans le schéma qui se suit :

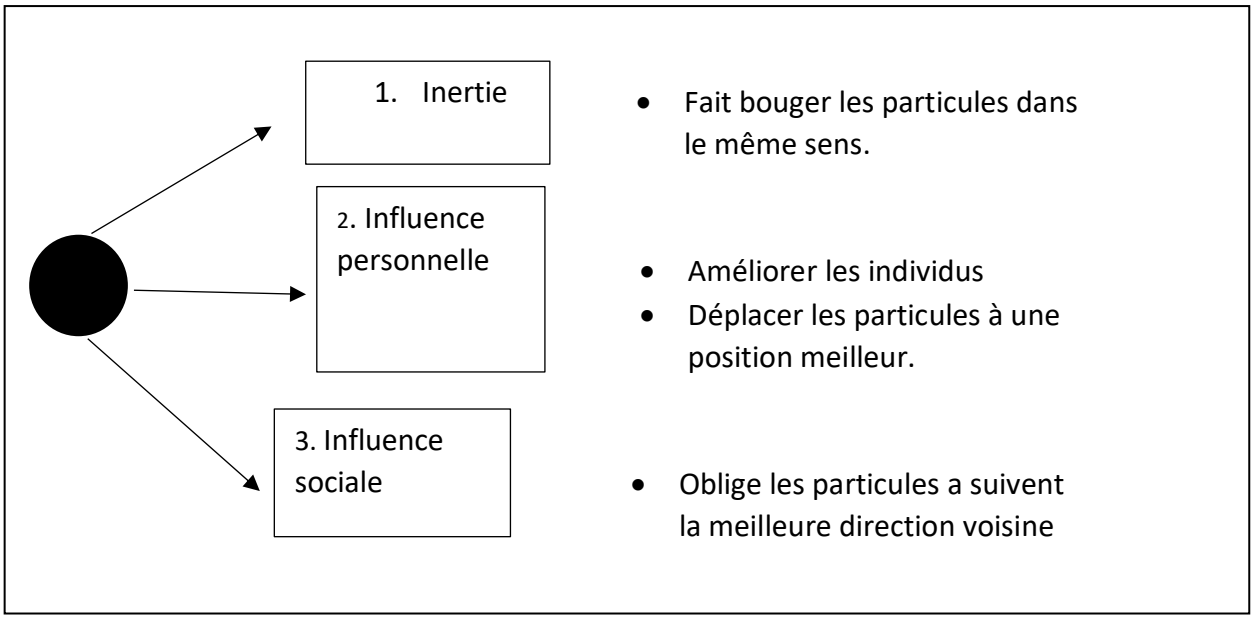


Figure3.2 : Schéma explicative du calcul de la vitesse de particule

2.4. Déplacement des particules :

Le déplacement de chaque particule est en fonction de sa vitesse et des deux meilleures positions trouvée (la sienne et celle de ses voisines) suivant les deux équations précédentes (1) et (2).

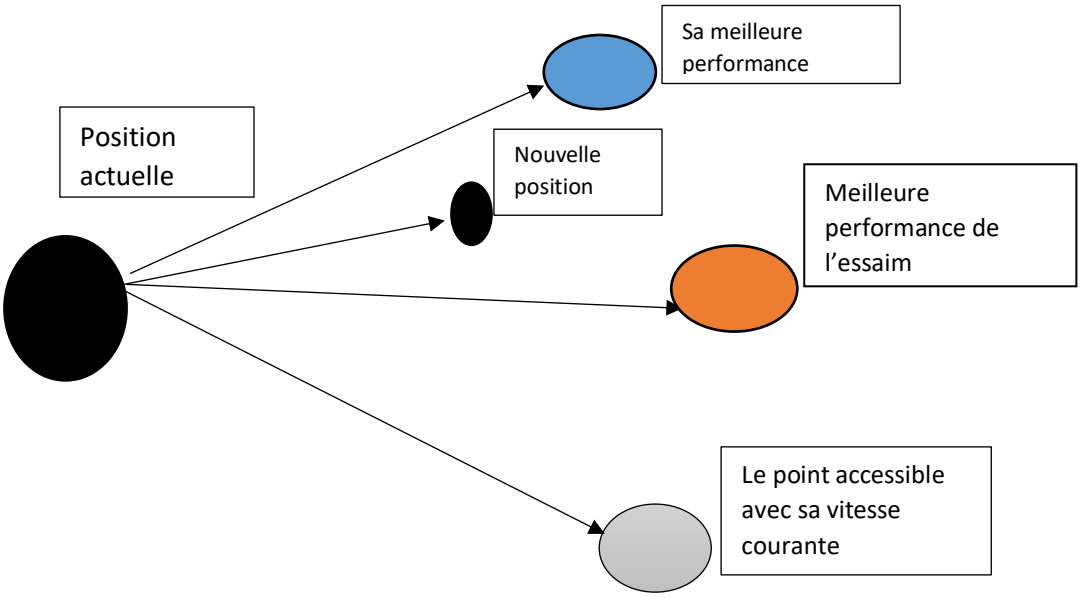


Figure.3.3 : Déplacement des particules

3. Le problème d'ordonnancement :

L'ordonnancement est une branche de la recherche opérationnelle et de la gestion de la production qui vise à améliorer l'efficacité d'une entreprise en termes de coûts de production et de délais de livraison [33].

D'après Carlier, J. et Chrétienne[34] Ordonnancer, c'est programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution.

Un problème d'ordonnancement en anglais job shop schudeling (JSP) se compose d'un ensemble fini des travaux (jobs) J = {1, 2,. . . n} et un ensemble fini de machines M = {1, 2,. . . m}. Plus précisément, chaque travail consiste en un ensemble fini d'opérations. Le traitement d'une opération doit être effectué sur une machine prédéfinie, i. e. la i-ième opération du travail j, notée oij, est traitée sur la machine Mij ∈ M. L'ordre des opérations de chaque travail est fixe, i. e. la séquence de la machine technologique donnée pour chaque travail doit être pris en compte[35].

Avant de la résolution d'un problème d'ordonnancement on fait intervenir plusieurs caractéristiques définies par [36] comme suit :

3.1.Les caractéristiques d'un problème d'ordonnancement :

a. Les tâches :

Une tâche est une entité élémentaire job. Elle fait référence à une opération caractérisée dans le temps, par une date de début et/ou de fin, et dont la réalisation nécessite une durée opératoire préalablement définie.

b. Les ressources :

Une ressource est un moyen technique ou humain destiné à être utilisé pour la réalisation d'une tâche spécifique ou de plusieurs tâches.

On compte deux types des ressources :

Les ressources renouvelables : ce sont les ressources réutilisables pour des autres taches après été avoir utilisé.

Les ressources consommables : ce sont les ressources, qui après avoir été allouées à une tâche, ne sont plus disponibles.

c. Les contraintes :

Elles représentent les limites imposées par l'environnement et elles sont différentes d'un problème à un autre. Il y'a deux types des contraintes suivant la disponibilité des ressources et l'évolution temporelle

- Les contraintes temporelles : représentent des restrictions sur les valeurs que peuvent prendre les variables de décision temporelles d'ordonnancement.
- Les contraintes de ressources : l'occupation des ressources se défèrent entre des ressources disjonctives qui exécutent les taches sur un intervalle disjoint et les ressources cumulative qui limitent le nombre des taches à exécuter en parallèle (disponibilité de la ressource).

d. Les critères d'optimisation :

C'est l'ensemble des objectifs qu'on veut optimiser et chaque critère est représenter par une fonction objective à maximiser ou minimiser. L'évaluation de la qualité de l'ordonnancement revient à satisfaire les différents critères.

On peut citer plusieurs critères liés à une application :

Les critères liés à l'utilisation des ressources :

- Maximiser l'utilisation de la machine.
- Minimiser le temps d'attente de l'ensemble des machines.
- Minimiser l'utilisation des ressources.

Les critères liés aux temps :

- La minimisation de temps d'achèvement terme (Makespan).
- La minimisation des retards de temps d'exécution des tâches.

3.2. Représentation de solution :

Plusieurs façons de représenter un problème d'ordonnancement est possible :

3.2.1. Diagramme de Grant :

C'est la représentation la plus connue d'un problème d'ordonnancement. On compte deux types de diagramme utilisé Gantt ressources et Gantt taches :

Le diagramme de Gant ressource : représente le temps d'exécution de chaque opération sur la ressource représentée comme suite :

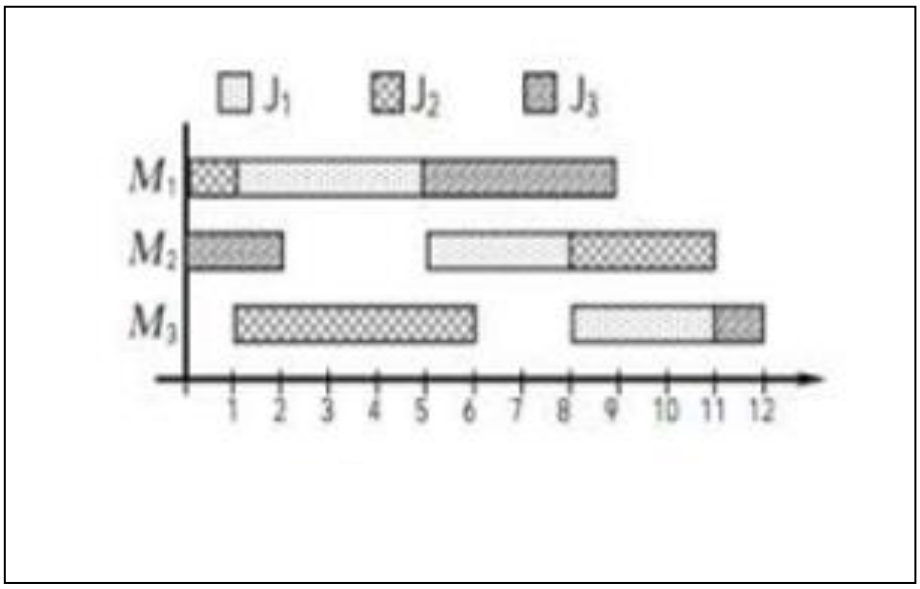


Figure3.4 : Diagramme de Grant de ressource[37]

Le diagramme de Gantt tâche : le temps d'exécution des opérations et le temps ou la tâche est en attente des ressources.

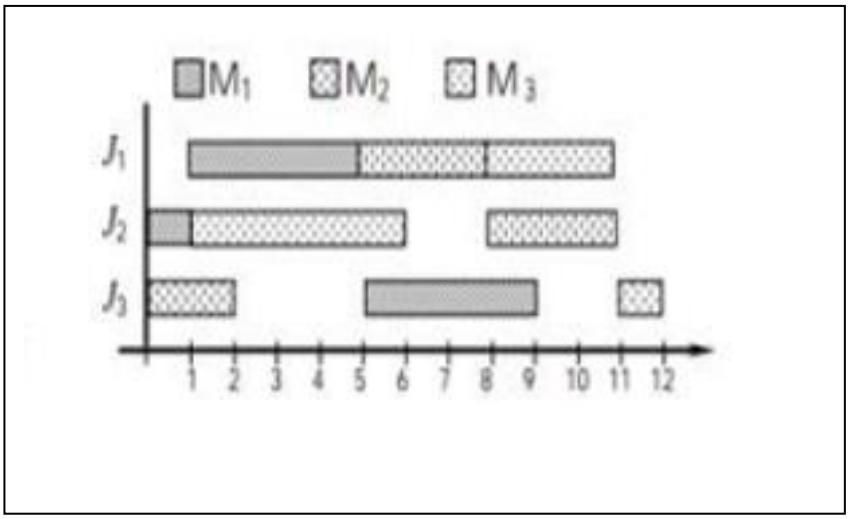


Figure 3.5 : Diagramme de Grant des taches [37]

3.2.2. Graphe Potentiel-Tâches :

Dans ce graphe**,** les tâches sont représentées par des nœuds et les contraintes par des arcs comme le montre la figure 3.6:

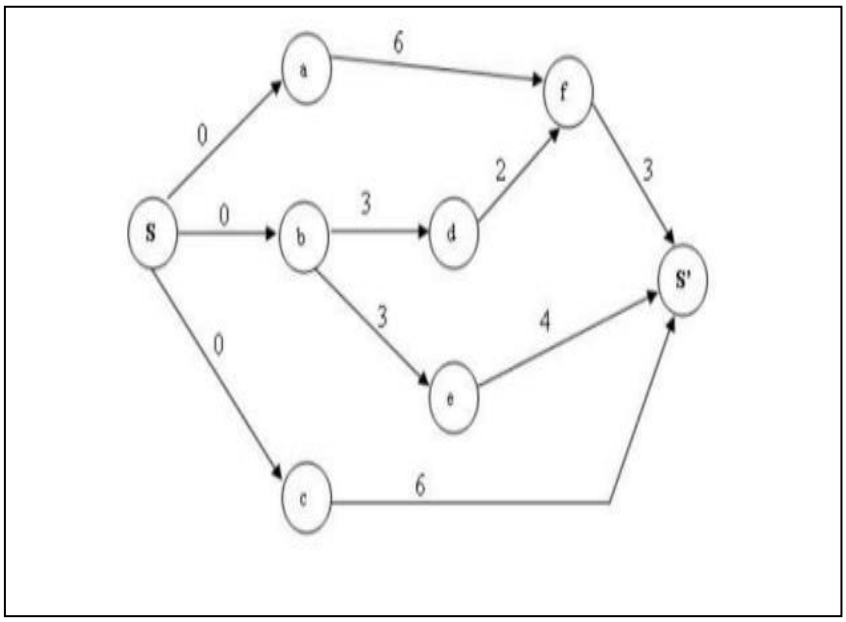


Figure 3.6: Graphe potentielle de taches [38]

La tâche « f » ne peut s'exécuter que si a et d ont été exécuter. Donc, pour exécuter « a » il faut 6 unités de temps et pour exécuter « d » il faut 2+3 unités de temps.

Espace de recherche :

Le Problème d'ordonnancement est un problème d'optimisation discret. Cependant l'approche proposée valide pour les espaces de recherche est continuée.

Pour adapter la version continue en version réelle pour un problème d'ordonnancement, une règle de la plus petite valeur de position SPV est utilisée [39].

La règle SPV permet de trouver une permutation correspondant à la position Xid. Le vecteur de position Xid a des valeurs continues. En utilisant la règle SPV, cette valeur de position continue peut-être convertie en permutation de valeur discrète

L'un des problèmes les plus importants lors de la conception de l'algorithme PSO réside dans sa représentation de la solution. Afin de construire une relation directe entre le domaine du problème et les particules PSO pour le problème d'ordonnancement, nous présentons *n* nombre de dimensions pour n nombre d'emplois (j = l..... n).

Chaque particule a un ensemble continu de valeurs de position. La particule elle-même ne présente pas de séquence. Au lieu de cela, nous utilisons la règle SPV pour déterminer la séquence de traitement impliquée par les valeurs de position x de la particule X.

Exemple d'application de SPV :

Vecteur des jobs [1,2,3,4,5,6].

Vecteur des positions(X) [1.75, **-0.85,** 3.2, -0.75, **-1.5,** 2.1]

Vecteurs des vitesses (V) [3.12, 2.3, 1.5, -0.8, -0.15, 1.9]

Séquence (SPV) [5,2,4,1,6,3]

Selon cette la règle, la plus petite valeur de position est x = -1.5, la dimension j = 5 est donc considérée comme le premier job la deuxième plus petite valeur de position est x=-0,85 la dimension j = 2 est donc considérée comme deuxième job), etc. En d'autres termes, les dimensions sont triées selon la règle SPV, c'est-à-dire jusqu'à construire une séquence de solution.

Dans ce travail une recherche locale est appliquée a la suite de l'algorithme PSO pour améliorer la solution trouvée.

En effet Toutes les métaheuristiques a population de solution sont suivi par la recherche locale dont laquelle on améliore la solution trouvée par une solution au voisinage si elle existe

4. Travaux connexes :

Plusieurs travaux réalisés dans ce sens. Parmi les travaux on cite des méthodes exactes sur le problème d'ordonnancement :

L'utilisation de méthode de Séparation et évaluation pour des problèmes d'ordonnancement dans les travaux de Jacques Carlier et Ismaïl Rebaï [40]et dans les travaux de Mohammad Mahdavi Mazdeh et Mohammad Rostami[41].
la programmation dynamique a proposé pour le problème d'ordonnancement dans les travaux de Ansis Ozolins[42]. et dans les travaux de Ji Ung Sun [43].
Les méthodes exactes sont très puissant dans la qualité de solution retrouvée parce que elle se basent sur un fondement mathématique mais elles révèlent des faiblesses en temps et espace mémoire dans les problèmes a grande taille.
Plusieurs travaux qui appliquent des méthodes approchées pour la résolution on peut citer : Les travaux de Ibrahim H Osman and CN Pots ont proposé un algorithme de recuit simulé afin de minimiser le temps d'exécution. comparé aux autres heuristiques l'algorithme proposé donne des bonnes résultats.[44]

Les algorithmes génétiques ont été largement utilisés pour résoudre le problème d'ordonnancement :

R. Ruiz, C. Maroto, and J. Alcaraz. [45] ont proposé deux algorithmes génétiques avec des nouveaux opérateurs qui donnent une initialisation efficace de la population. On peut citer aussi les travaux de [46, 47] dans l'application de l'algorithme génétique.

L'application de la recherche tabu pour résoudre ce problème est fournée dans les travaux de de Taillard [48],celui de Nowicki et al. [49] et celui de Chen et al. [50] en 2008.

Ching-Jong Liao, Chao-Tang Tseng, et Pin Luarn. [51] ont proposé l'algorithme PSO pour résoudre ce problème, avec une représentation binaire et un modèle de vitesse stochastique. Ces travaux montré l'efficacité de l'algorithme comparé aux algorithmes génétiques.

Qan-Ke Pan, Ling Wang, M Fatih Tasgetiren, et Bao-Hua Zhao. [52] Ont présenté aussi un algorithme PSO avec une représentation de la solution par permutation combiné a un algorithme de recherche locale avec contrainte de "sans-attente"afn de minimiser le temps d'exécution.

Zhigang Lian, Xingsheng Gu, and Bin Jiao [53] ont propose un modèle PSO par permutation pour de minimiser le temps d'exécution.

Les travaux réalisés ont montré l'importance du problème d'ordonnancement et la complexité de sa nature.

Les contributions dans ce domaine restent ouvertes. Dans notre travail, une étude de l'application PSO au problème d'ordonnancement est présenté dans ce qui suit.

5. Conclusion

Le PSO original a été utilisé pour résoudre des problèmes continus mais En raison des espaces de solutions discrets des problèmes d'optimisation de l'ordonnancement, nous avons modifié la représentation de la position**.**

Dans le chapitre suivant ,on va réaliser une implémentation de cet algorithme selon notre étude conceptuelle en terminant par une expérimentations et discussion de résultats.

Chapitre 4 :
Implémentation

1. Introduction :

Dans ce chapitre nous allons présenter l'implémentation de notre travail en exposant les différents outils. Nous allons aussi illustrer avec explication détaillée les différentes interfaces de l'application et nous terminons par l'expérimentation et la discussion de résultats.

2. Présentation de l'environnement de développement :

2.1.Matériels utilisés pour le développement de l'application :

Nous avons utilisé le matériel suivant :

Fabricant *:*	HP
Processeur *:*	AMD E1-6015 APU 1.4GHZ
Disque dur :	4 Go
RAM :	4 Go
Système d'exploitation :	Windows 8.1 64bit

Tableau 1 : Description de matérielle utilisé

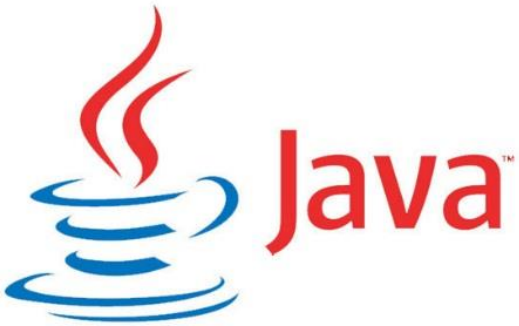
2.2.Logicielle utilisés pour le développement de l'application :

2.2.1. Eclipse IDE :

La plate-forme Eclipse est une fondation générique pour environnement de développement intégré (IDE) sans langage de programmation spécifique. La plate-forme contient la fonctionnalité IDE et est construite avec des composants créant des applications à l'aide de sous-ensembles de composants. Les développeurs créent, partagent et modifient des projets et des fichiers génériques sur la plate-forme, tout en participant à un référentiel d'environnement de développement à plusieurs équipes.

La principale fonction de la plate-forme est de fournir des mécanismes et des règles aux éditeurs de logiciels, permettant une intégration logicielle fluide entre différents éditeurs.

2.2.2. Langage JAVA :



Java est langage programmation informatique orienté objet crée par James Gosling et Patrick Naughton de Sun Microsystems. Mais c'est également un environnement d'exécution. Java pour sépare deux parties. D'une part, votre programme écrit en langage java et d'autre part, une machine virtuelle (JVM) qui se va charger de l'exécution de votre programme Java. . Avec le langage Java, vous pouvez développer, des applications Desktop, développer des applets pour vos sites web, développer des sites en JSP et des applications pour téléphone mobile

2.2.3. Interface graphique d'utilisateur GUI :

Une interface utilisateur graphique (GUI) est une interface par laquelle un utilisateur interagit avec des appareils électroniques tels que des ordinateurs, des appareils portatifs et d'autres appareils. Cette interface utilise des icônes, des menus et d'autres représentations d'indicateurs visuels (graphiques) pour afficher des informations et les contrôles utilisateur correspondants, contrairement aux interfaces à base de texte, où les données et les commandes sont en texte. Les représentations graphiques sont manipulées par un périphérique de pointage tel qu'une souris, une boule de commande, un stylet ou un doigt sur un écran tactile.

La nécessité d'une interface graphique est devenue évidente car la première interface de texte homme / ordinateur a été créée via la création de texte au clavier par une invite (ou une invite DOS). Des commandes ont été tapées sur un clavier à l'invite du DOS pour lancer les réponses à partir d'un ordinateur. L'utilisation de ces commandes et la nécessité d'une orthographe exacte ont créé une interface encombrante et inefficace.

2.2.4. Interface de programmation d'application (API):

Une interface de programmation d'application (API) est un ensemble de protocoles, de routines, de fonctions et / ou de commandes que les programmeurs utilisent pour développer des logiciels ou faciliter l'interaction entre des systèmes distincts. Les API sont disponibles pour une utilisation à la fois sur le bureau et sur mobile, et sont généralement utiles pour la

programmation de composants d'interface graphique (GUI), ainsi que pour permettre à un logiciel de demander et de gérer des services d'un autre programme.

2.2.5. Java Swing :

Java Swing est une boîte à outils de widgets d'interface graphique utilisateur Java légère comprenant un riche ensemble de widgets. Il fait partie de Java Foundation Classes (JFC) et comprend plusieurs packages permettant de développer des applications de bureau riches en Java. Swing inclut des commandes intégrées telles que des arbres, des boutons d'image, des volets, des curseurs, des barres d'outils, des sélecteurs de couleurs, des tableaux et des zones de texte. Les composants Swing sont entièrement écrits en Java et sont donc indépendants de la plate-forme.

2.2.6. L'utilisation de JApplet :

JApplet est une classe publique java swing conçu pour les développeurs, généralement écrite en Java. JApplet se présente généralement sous la forme de bytecode Java qui s'exécute à l'aide d'une machine virtuelle Java (JVM) ou du visualiseur d'applets de Sun Microsystems. Il a été introduit pour la première fois en 1995.

JApplet peut également être écrit dans d'autres langages de programmation et peut ensuite être compilé en code octet Java.

3. Présentation de l'application :

3.1.Fenêtre de démarrage :

C'est la page qui permet à l'utilisation de :

a. Choisir l'instance de problème a traité**.**
b. Afficher les dimensions de l'instance de problème
c. Sélectionner le nombre des individus (particules).
d. Sélectionner le temps.
e. Sélectionner le nombre d'itérations.
f. Démarrage de solution.

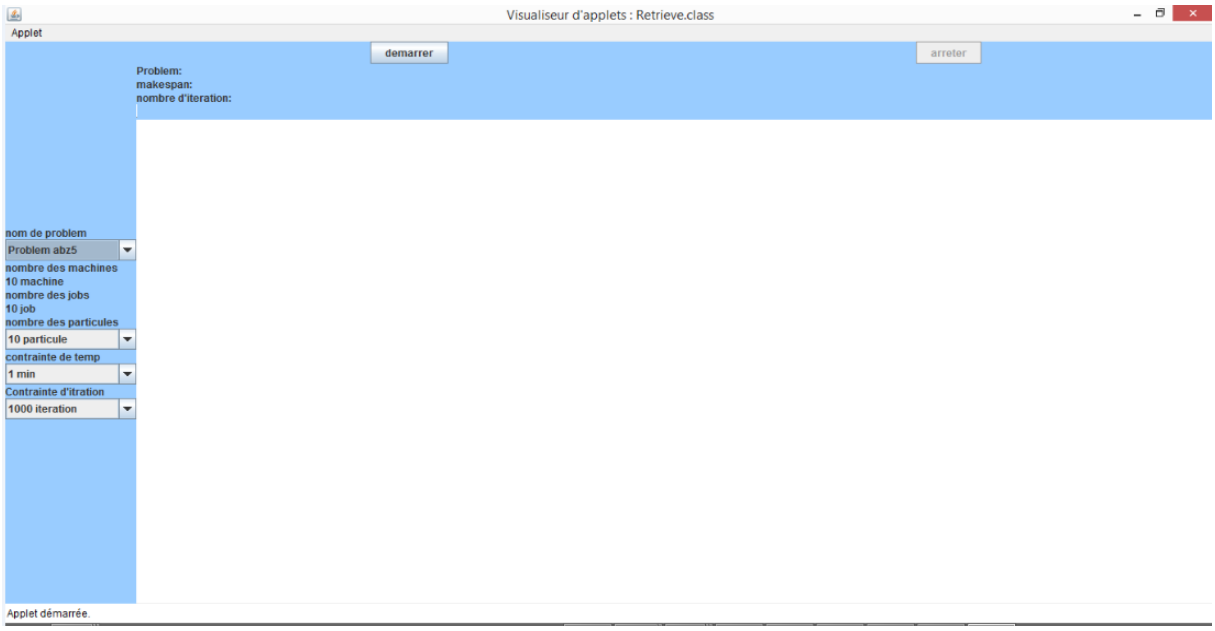


Figure 4.1: Fenêtre de démarrage

3.2.Choix de l'instance de problème à traiter :

Ici on peut choisir l'instance de problème parmi plusieurs instances de dimensions et complexités différentes. Dans ce travail un Benchmark regroupant plusieurs instances de test est utilisé. Ce dernier a été construit par plusieurs auteurs et représente un outil puissant de comparaison.

En effet, il regroupe des exemples de test avec la solution optimale.

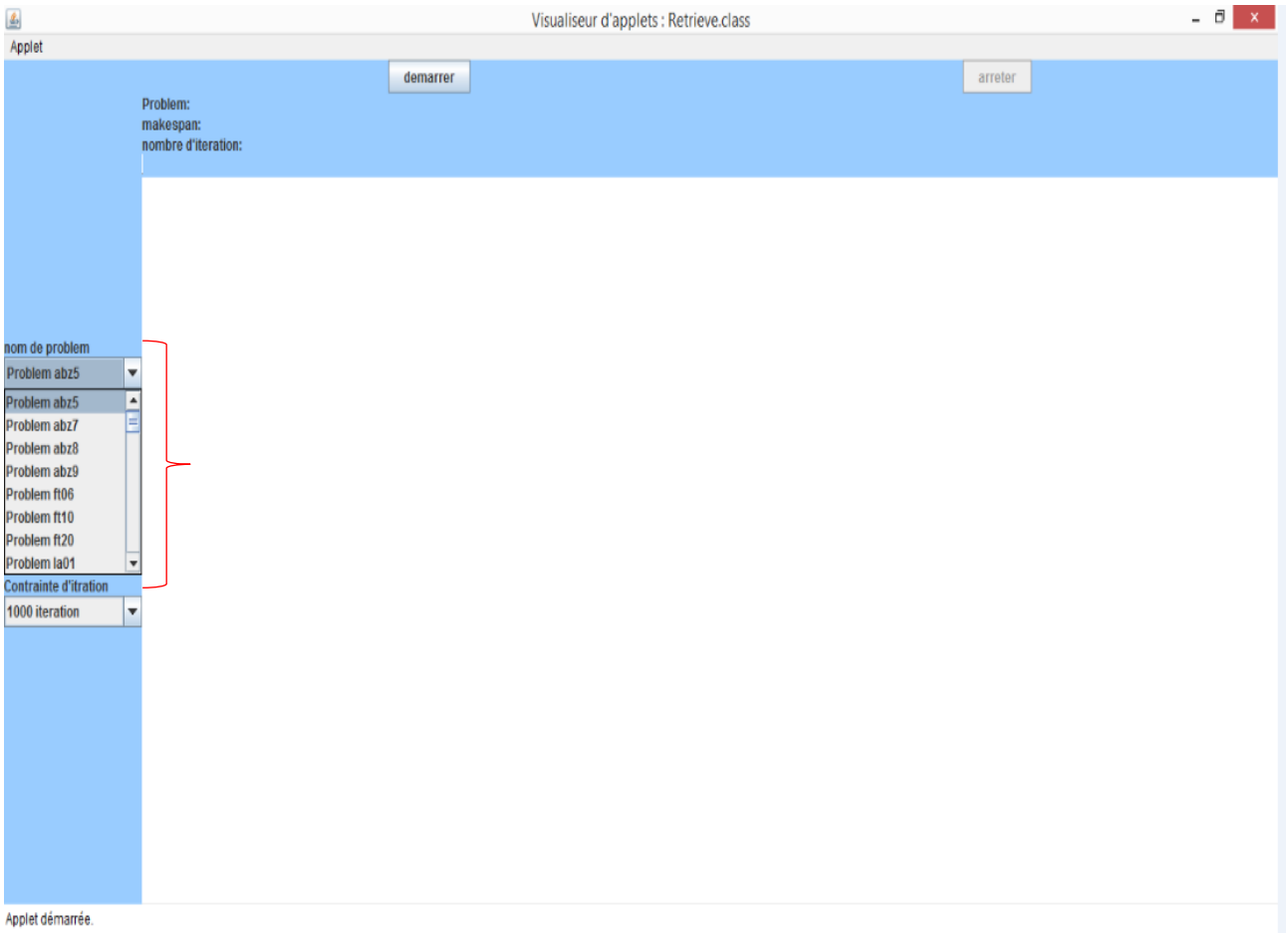


Figure 4.2: Choix de l'instance de problème à traiter

3.3.Affichage des démentions de l'instance de problème :

Après avoir choisi le problème les informations concernant le nombre de machines et le nombre des jobs est récupéré. Ces informations sont :

- Nombre des machines
- Nombre des jobs

Représentée dans la figure suivante :

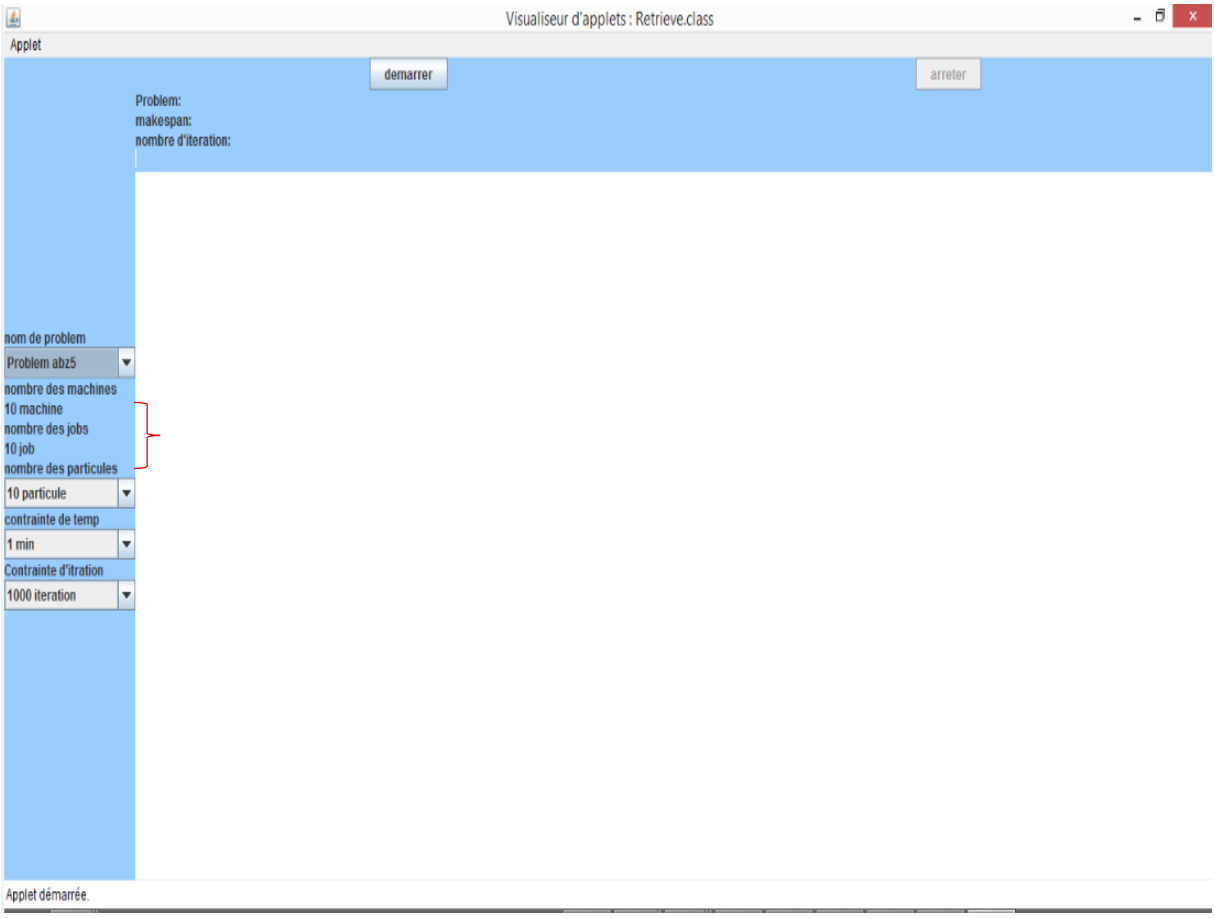


Figure 4.3 : Affichage des démentions de l'instance de problème

3.4.Sélectionner le nombre des individus (particules) :

Cette section est pour déterminer nombre de particule utilisé ou bien la population initiale pour la résolution du problème.

On a quatre choix concernant la sélection de nombre de particule :

- 10 particules
- 20 particules
- 30 particules
- 50 particules

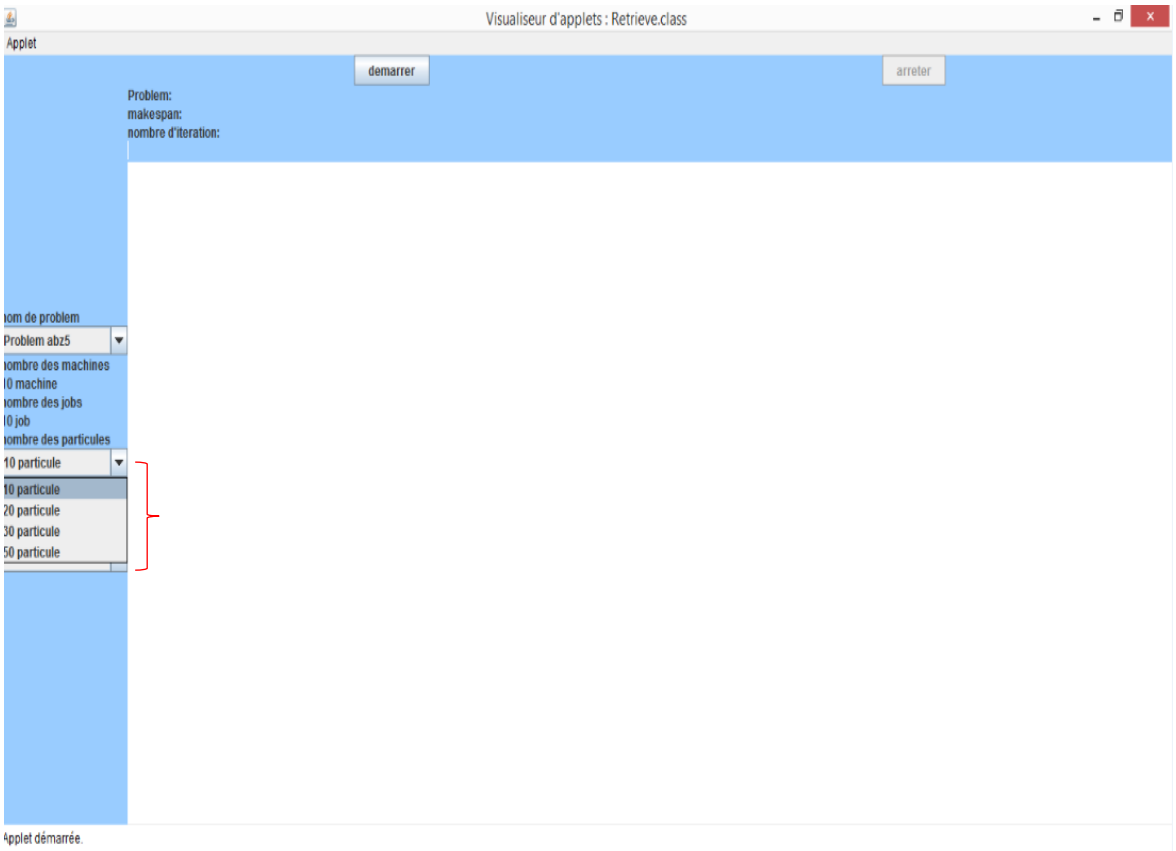


Figure 4.4 : Choix de nombre de particules

3.5.Sélectionner la limitation de temps :

L'arrêt de l'exécution de l'algorithme se fait selon deux contraintes

La première contrainte c'est le temps d'exécution :

On a cinq choix concernant la sélection de contrainte de temps :

- Pas de limitation de temps
- Arrêt dans 1 minute
- Arrêt dans 2 minutes
- Arrêt dans 5 minutes
- Arrêt dans 10 minutes

La deuxième contrainte c'est le nombre d'itérations on a huit choix :

- Pas d'itérations
- 2 itérations
- 10 itérations
- 50 itérations
- 1000 itérations

- 10000 itérations
- 500000 itérations

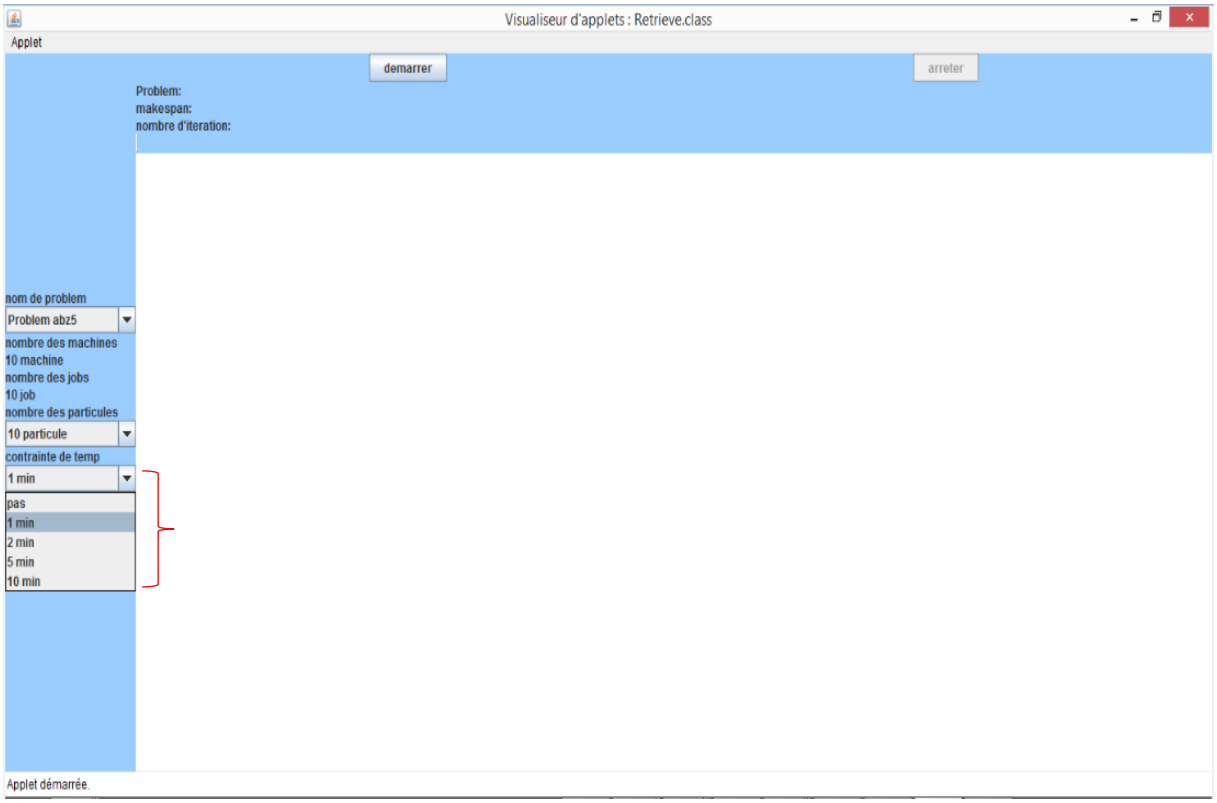


Figure 4.5 : Choix de temps d'exécution.

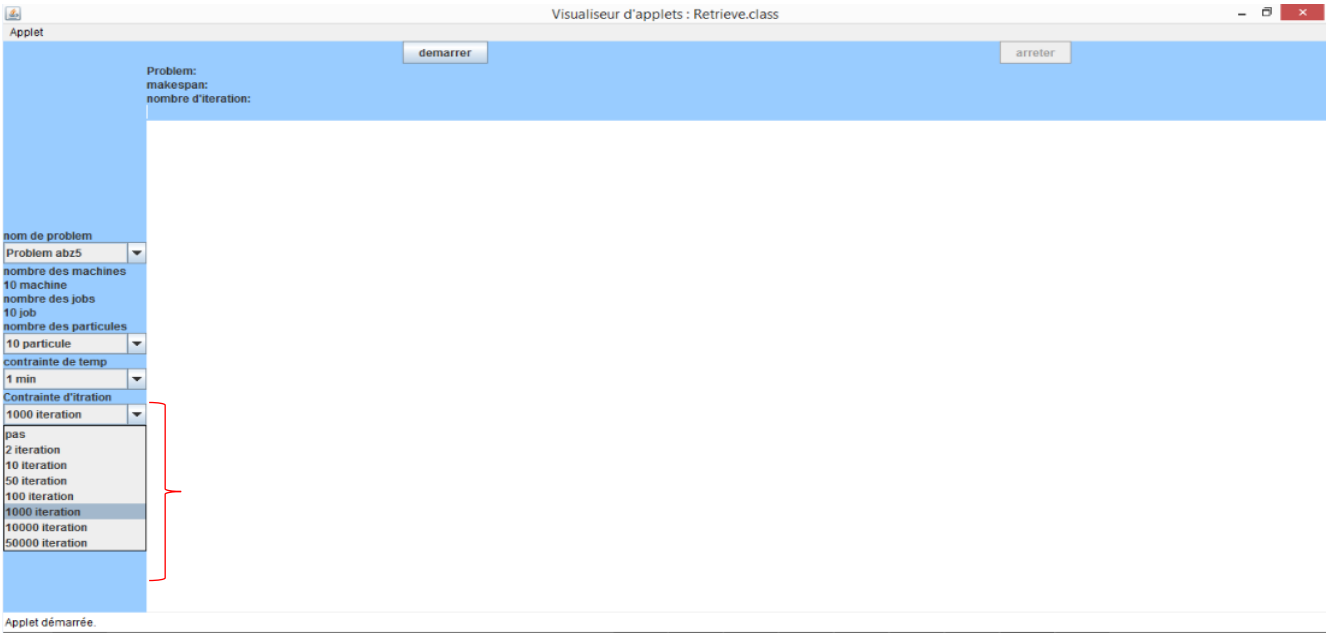


Figure 4.6 : Choix de nombre d'itérations.

3.6.Démarrage de solution :

Le bouton « Démarrer » permis d'exécuter l'algorithme de résolution plusieurs informations sont affichées :

a. Le Makespan (le temps d'achèvement d'exécution)
b. Le nombre d'itération en cours
c. Le nom d'instance de problème

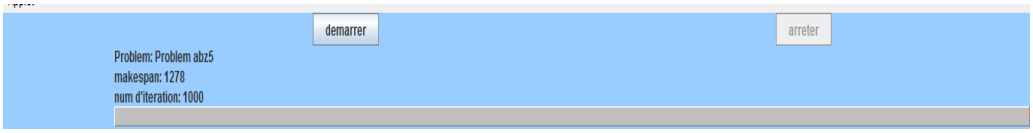


Figure 4.7 : Bouton « Démarrer »

d. Le diagramme de Grant :

Représentation des exécutions des taches sur les machines selon la solution retrouvée par l'algorithme appliqué :

- L'axe vertical représente les machines
- L'axe horizontal représente les jobs
- Chaque job est caractérisé par une couleur spécifiée.

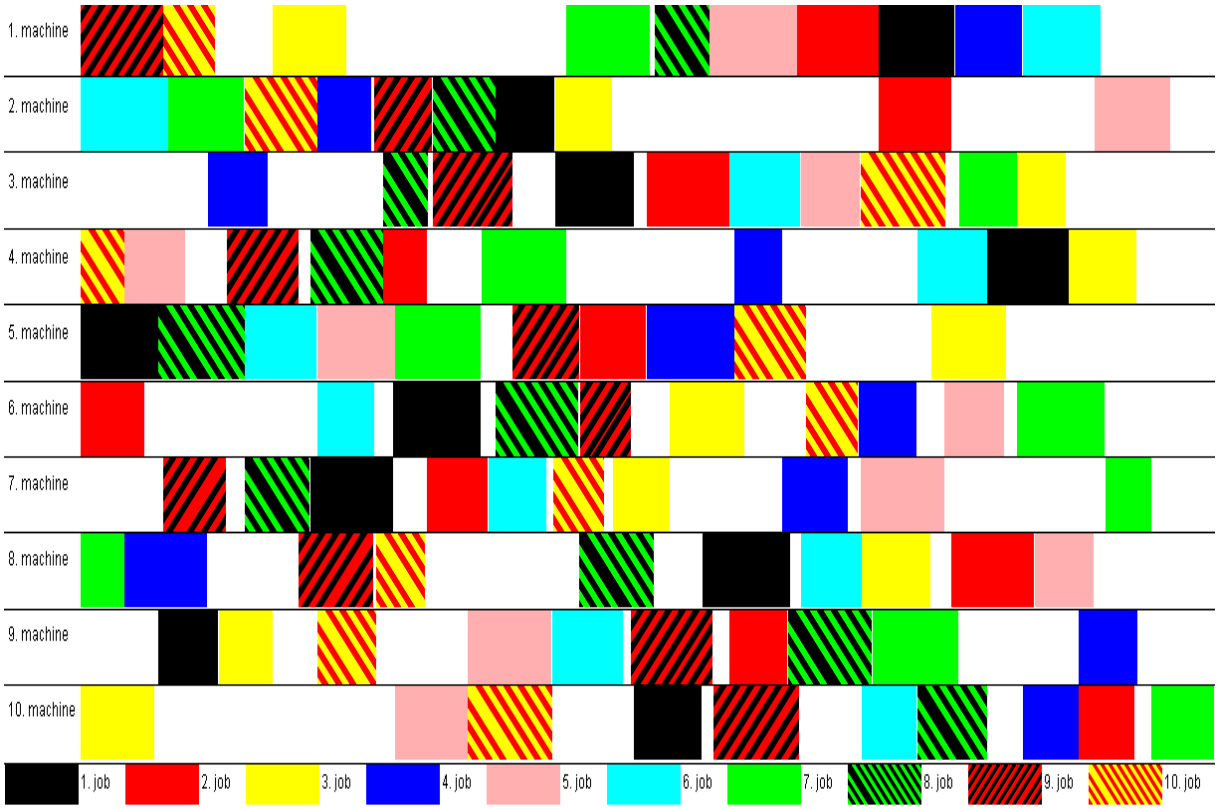


Figure 4.8 : Diagramme de Grant

4. Tests expérimentaux :

4.1.Les représentations de problèmes :

Quand on lance l'exécution, les données de l'instance de problème sélectionné sont récupérées et affichées comme suit :

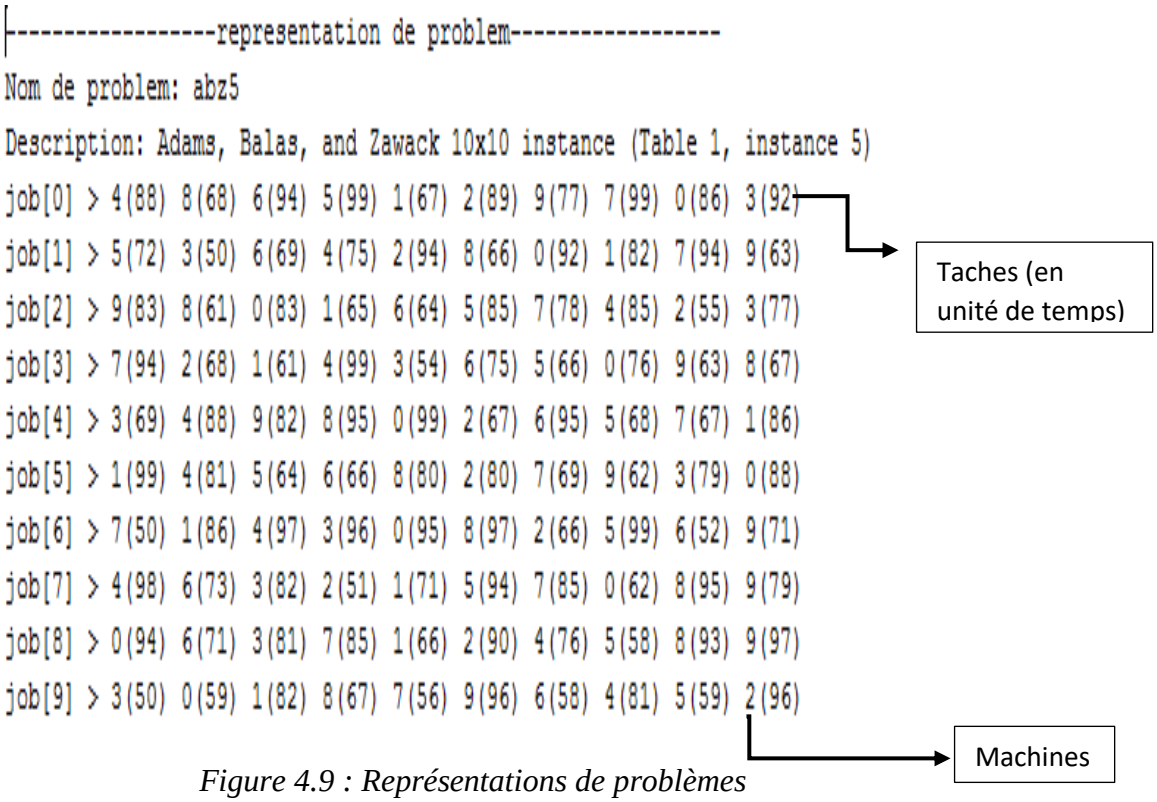


Figure 4.9 : Représentations de problèmes

La représentation affiche les informations suivantes :

- Le nombre et l'ordre des jobs
- La machine utilisée par chaque tache.
- Le temps d'exécution de chaque tache dans la machine (nombre affichée entre parenthèses).

4.2.Représentation des particules :

- Chaque particule est caractérisée par sa vitesse et position locales.la figure suivante représente l'ensemble des particules obtenues en spécifiant :
- La dernière position obtenue pour chaque particule.
- La vitesse de chaque particule.
- La meilleure position obtenue pour chaque particule.
- Le meilleur ordre obtenu selon la règle SPV.

```
resultats
-----------------solutions-----------------
Particle num: 0>
Makespan =1304
Last position x =[-4.52,-2.01,-10.2,-13.8,-4.65,+19.0,+4.15,+6.66,-2.09,+1.95,-8.97,+34.4,+1380,-12.7,-52.2,-126.,-0.28,-1.26,-5.24,-9.95,+4.91,+4.22,-1.13,-0.61,
Last volicty =[+1.26,+0.03,-15.5,-0.07,-0.71,-7.04,-0.02,-0.40,+0.02,+0.04,-0.09,-15.3,+2661,-0.49,+3.90,-353.,-4.30,-0.57,-0.62,-0.26,+8.51,+1.46,-0.81,+6.11,-0.
Best local position x =[-4.96,-2.00,+4.04,-14.1,-4.22,+4.73,+4.10,+6.24,-1.48,-3.12,-11.2,+76.8,-108.,-13.1,-4.14,-1.73,-0.23,-0.53,-4.99,-10.0,+2.08,+1.17,-2.38,
Best order =[24,12,57,71,73,84,74,86,38,3,89,13,10,19,99,47,52,31,90,93,69,35,59,55,46,91,23,18,39,0,96,40,75,45,27,81,4,14,30,33,76,98,56,87,68,65,25,43,29,50,7
-----------------solutions-----------------
Particle num: 1>
Makespan =1299
Last position x =[-7.12,-2.04,-6.12,-13.4,-3.77,+13.4,+4.21,+16.3,-3.04,-5.91,-8.25,+28.3,-75.4,-14.0,-29.5,-15.4,+5.79,-1.41,-4.68,-9.77,-9.44,-5.08,-0.87,-3.28,
Last volicty =[-2.70,-0.13,-9.23,-1.13,+0.02,+5.73,+0.28,+5.64,-2.59,-0.27,+2.53,-1.13,-6.53,-2.92,+12.0,-4.70,-0.37,-0.34,+0.19,-0.14,-6.20,-14.2,-1.18,-3.34,-4.
Best local position x =[-4.78,-2.01,+4.10,-14.8,-3.77,+4.84,+4.01,+4.51,-1.44,-6.16,-12.2,+54.1,-48.0,-12.9,+5.96,-4.70,-0.23,-0.41,-4.94,-9.64,-6.37,+1.32,-3.20,
Best order =[24,57,12,84,71,73,74,38,86,3,13,10,89,19,99,47,69,55,52,59,20,90,9,35,39,28,46,91,45,81,18,0,96,15,30,77,56,33,98,76,4,68,25,87,43,27,80,50,22,65,29,
-----------------solutions-----------------
Particle num: 2>
Makespan =1295
Last position x =[-6.29,-1.96,-3.22,-13.3,-3.80,+23.2,+4.48,+14.5,-2.53,-5.33,-6.23,+15.5,-31.9,-11.1,-59.7,-21.9,+4.91,-1.67,-4.58,-9.71,-4.97,+5.34,-0.56,-0.39,
Last volicty =[-1.27,-8.79,-4.19,-6.55,-1.15,-5.35,-4.62,-5.13,-1.10,+2.83,-8.28,-2.74,-1.84,-7.13,-5.10,-3.72,-1.69,+3.76,+1.49,-1.05,+1.75,+2.02,+4.22,-1.19,+1.
Best local position x =[-6.29,-1.96,-3.22,-13.3,-3.80,+23.2,+4.48,+14.5,-2.53,-5.33,-6.23,+15.5,-31.9,-11.1,-59.7,-21.9,+4.91,-1.67,-4.58,-9.71,-4.97,+5.34,-0.56,
Best order =[24,69,14,84,71,12,55,73,88,74,15,86,39,58,38,3,47,89,13,52,19,99,35,93,0,10,59,56,91,9,90,96,98,25,20,77,45,18,36,65,94,80,76,82,75,4,30,87,68,29,64,
-----------------solutions-----------------
Particle num: 3>
Makespan =1298
Last position x =[-6.35,-1.99,-12.4,-13.9,-3.82,+1.98,+4.55,+13.8,-2.15,-8.04,-16.5,+16.4,-36.8,-13.4,-106.,-10.4,+1.96,+2.92,-5.42,-9.71,-4.97,+4.29,-0.69,-0.79,
Last volicty =[-0.29,+0.00,-13.4,-0.56,-0.10,-65.1,-0.34,+1.98,+0.21,-5.64,-4.66,-1.10,-2.22,-1.43,-2.11,-8.27,+0.25,+21.4,-0.10,+0.03,-1.41,+0.22,+0.03,-2.14,-4
Best local position x =[-5.96,-2.02,-12.9,-13.7,-3.62,+4.80,+4.06,+12.6,-2.20,-3.11,-12.9,+17.9,-36.3,-13.1,+16.6,-13.7,+0.69,-7.57,-5.01,-9.73,-4.97,+3.62,-0.80,
Best order =[24,84,90,12,69,58,73,71,86,78,74,38,3,15,47,13,10,2,35,89,52,19,93,99,46,17,55,59,0,39,91,96,36,18,20,77,45,25,56,75,33,80,98,65,76,68,44,87,4,64,50,
-----------------solutions-----------------
```

Figure 4.10 : Représentation des particules

4.3.Meilleur solution trouvée :

Les resultats finaux representés dans la figure suivante determinent les informations suivantes :

- Le milleur makespan(temps d‘achevement d‘exucuion).
- La milleur position global des particules de tache rerouvée.
- La milleur sequence d’ordre.

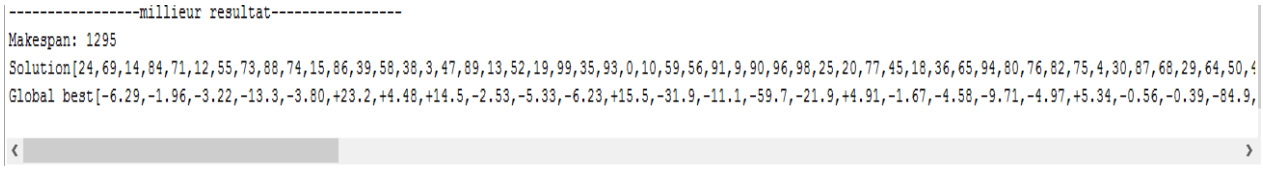


Figure 4.11: Meilleur solution trouvé

5. Discussion de résultats :

Nous utilisons 24 instances issues de ORLibrary 1990[54] comme exemple pour tester l'algorithme proposé, ABZ5, ont été conçus par Adam J,Balas E ,Zawack[55].Les instances LA01 – LA23 qui ont été conçues par Lawerence [56].

Le Tableau suivant représente une évaluation des résultats de makespan obtenus par l'algorithme proposé dans ce travail par apport à la solution optimale donnée par ce Benchmark de test.

La dimension de problèmes représenté sous la forme (m*n) Avec m représente le nombre des jobs et n les nombre de machine

En effet, les problèmes métaheuristiques les solutions retrouvées sont liée de l'initialisation des paramètres de l'algorithme PSO.

Instance	Dimension(m*n)	Meilleur résultats connu	PSO
abz5	10 x 10	1234	1249
la01	5 x 10	666	666
la02	5 x 10	655	655
la03	5 x 10	597	617
la04	5 x 10	590	593
la05	5 x 10	593	593
la06	5 x 15	926	926
la07	5 x 15	890	890
la08	5 x 15	863	863
la09	5 x 15	951	951
la10	5 x 15	958	958
la11	5 x 20	1222	1222

la12	5 x 20	1039	1039
la13	5 x 20	1150	1150
la14	5 x 20	1292	1292
la15	5 x 20	1207	1207
la16	10 x 10	945	979
la17	10 x 10	784	784
la18	10 x 10	848	859
la19	10 x 10	842	866
la20	10 x 10	902	907
la21	10 x 15	1040	1097
la22	10 x 15	927	976
la23	10 x 15	1032	1032

Tableaux 2 : Résultats comparatifs

Les solutions retrouvées se rapprochent des solutions optimales déjà connue dans le benchmark utilisé. La qualité de la solution dépend fortement des paramètres d'initialisation de l'algorithme PSO.

6. Conclusion :

Dans ce chapitre nous avons exposé l'implémentation de la méthode adoptée. Les résultats obtenus sont encourageants et démontrent l'efficacité de la méthode utilisée.

❖CONCLUSION GENERALE

Dans ce mémoire que nous nous intéressons à la résolution d'un problème d'optimisation qu'est le problème de l'ordonnancement. Notre objectif était d'implémenter une méthode approchée à base de l'algorithme d'optimisation par essaim particulaire (PSO) combiner à un algorithme de recherche locale (pour la résolution de ce problème). Nous avons présenté dans notre travail, quelques notions importantes sur les problèmes d'optimisation et les méthodes de résolution dans un première partie de ce travail, ensuite dans la deuxième partie nous avons présenté les méthodes de résolution pour ce problème.

L'implémentation de la méthode appliquée, a permis d'obtenir des résultats satisfaisants en un temps raisonnable. L'évaluation des résultats obtenus avec les solutions optimales en utilisant des benchmarks reconnus dans la littérature montre que les algorithmes PSO sont une bonne alternative pour la résolution de ces prolemmes d'ordonnancement.

En perspective nous pouvons discuter et comparer les résultats obtenus avec d'autre méthodes approchées (les algorithmes génétiques, les algorithmes de colonne de fourmis) pour évaluer les performance de la méthode adoptée. Une hybridation avec d'autre méthodes de recherches locales peut être aussi tester .

❖REFERENCES BIBLIOGRAPHIQUES :

[1] Paschos V, Optimisation combinatoire : concepts fondamentaux, Hermès science publication : Lavoisier. 2005

[2] Alain B . Optimisation Multiobjectif et Stratégie d'évolution en environnement Dynamique.2001

[3] Labed Said.Méthodes bio-inpirées hybrides pour la résolution de problèmes complexes.2013 .

[4]Yezid D & Ramon F.Multi-Objective Optimization in Computer Networks Using Metaheuristics.Auerbach Publications Taylor & Francis Group.2007

[5] Ali Khanafer ; Algorithmes pour des problèmesde bin packing mono- et multi-objectif; 2010 .

[6]Yann C & Patrick S.Multi-objetive Optimization Principles and Case Studies. France: Groupe Eyrolles.2003

[7]Gade P. Multi-Objective Optimization Techniques and Applications in Chemical Engineering. World Scientific Publishing.2009

[8]Fran S & Valder S.Multi-Objective Optimization Problems Concepts and Self-Adaptive Parameters with Mathematical and Engineering Applications. Sringer Nature.2017

[9]Mardle, S., Pascoe, S., and Tamiz, M. An Investigation of Genetic Algorithms for the Optimization of Multiobjective Fisheries Bioeconomic Models.in Proceeding of the Third International Conference on Multi-Objective Programming and Goal Programming: Theory and Applications (MOPGP98). Quebec City, Canada.1998

[10]Tapia, M. G. C. and Coello, C. A. C. 2007. "Applications of Multi-Objective Evolutionary Algorithms in Economics and Finance: A Survey." IEEE Congress on Evolutionary Computation. CEC. 25-28 Sept. Singapore.

[11] Jena, S. Multi-Objective Optimization of The Design Parameters of Shell and Tube Type Heat Exchanger Based on Economic and Size Consideration. 2013

[12] Lawler E, Wood E. Branch and bound methods: A survey.Operations Research Vol. 14, No. 4, pp. 699-719.1966

[13] Gomory R.Outline of an algorithm for integer solutions to linear programs. Math. Soc.Vol. 64 No. 5,pp 275--278.1958

[14] Barnhart C, Johnson Nemhauser G, Branch-and-price: Column generation for solving huge integer programs, Operations Research.Vol 46. No 03.pp 293-304.1998.

[15]Bellman, R. On the theory of dynamic programming.Proceedings of the National Academy of Sciences.Vol 38.No 8 pp 716–719.1952.

[16]Jourdan L .Basseur M,.Talbi E.Hybridizing exact methods and metaheuristics: A taxonomy. European Journal of Operational Research. Vol 199.No 3 .pp 620-629.2007

[17]G. Polya,.How to Solve It., Princeton Univ.Press, 1945

[18]Mihai G. Heuristic and Metaheuristic Optimization Techniques with Application to Power Systems. Technical University of Iasi Romania conference

[19]Glover F. Tabu Search Methods for Optimization. Amsterdam: European Journal of Operational Research. Vol 20 No 4. pp 74-94 .1990

[20] Stefan V.Meta-heuristics: The State of the Art Proceedings of the Workshop on Local Search for Planning and Scheduling-Revised Papers vol 2148 pp 1-23.2001

[21] Blum C& Roli A, Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR). Vol 35. No 3. pp 268-308 September 2003

[22] Marc S.Metaheuristiques Strategies pour l'optimisation de la production de biens et de services.2004

[23] Goss S, Aron S, Deneubourg L & Pasteels M.Self organized shortcuts in the Argentine ant. Science of Nature. Vol 76. No 12 pp579-581.1989

[24] Dorigo M, Maniezzo V & Colorni A. The Ant System: Optimization by a colony of cooperating agents, Submitted to IEEE Transactions on Systems. Vol26 . No 1 .1996

[25]David E. Goldberg.Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley1989.

[26]John H. Adaptation in Natural and Artificial Systems. MIT Press.1975

[27] Kennedy J & Eberhart R. Particle Swarm Optimization, In: Proc. IEEE Int. Conf. On Neural Networks.conf 1995.

[28] Kirkpatrick S,. Gelatt C, & Vecchiet M. Optimization by Simulated Annealing. Science. Vol. 220, No 4598, pp. 671-680.1983

[29] Glover F and Laguna M. *Tabu Search*. Kluwer Academic Publishers,1997.

[30] Xue-Feng Z. Miyuki K.Combining Hiroshi F. Ryuzo H.PSO and local search to solve scheduling problems. 13th Annual Genetic and Evolutionary Computation Conference.2011

[31] Zhixiong L. Investigation of Particle Swarm Optimization for Job Shop Scheduling Problem. Third International Conference on Natural Computation.IEEE. Haikou, China.2007

[32] Sirinivas P.Ramamchanra R.Rao C .Particule swarm Optimization approche for schudling of flexible job shops.Internaional jounal of Enineering reaserch and technolgy .Vol 1 No 5.2012

[33] Penido L . Scheduling - Theory, Algorithms, and Systems.Springer.2012.

[34] Carlier J . Chretienne P**.** Problemes d'ordonnancement. Elsevier Masson.1998

[35] french s. Sequencing and scheduling: An introduction to the mathematics of the job-shop. Ellis Horwood series in mathematics and its applications.1981.

[36] Maroua N. Implementation d'une methaheurestique embarquée pour resoudre le probleme d'ordennancement dans un atelier flexible de production.2017.

[37]: Lopez P. & Esquirol P .Lordonnancement, Economica, Parise 1999.

[38]:B. Roy, « Algèbre moderne et théorie des graphes, volume 2 ». Editions Dunod, Paris.

[39] Tasgetiren, M.F., Liang, Y.-C., Sevkli, M., and Gencyilmaz, G. Particle swarm optimization algorithm for makespan and maximum lateness minimization in permutation flowshop sequencing problem. In Proceedings of the 4th International Symposium on Intelligent Manufacturing Systems (IMS2004), Sakarya, Turkey, pp. 431-441. 2004.

[40] Jacques Carlier and Ismaïl Rebaï. Two branch and bound algorithms for the permutation flow shop problem. European Journal of Operational Research, Vo l90 .No 2 .pp238–251.1996.

[41] Mohammad M .Mohammad R. A branch-and-bound algorithm for two-machine flow-shop scheduling problems with batch delivery costs. International Journal of Systems Science : Operations & Logistics, Vol 1.No 2.pp 94–104.2014.

[42] Ansis O. Improved bounded dynamic programming algorithm for solving the blocking flow shop problem. *Central European Journal of Operations Research*, pp 1–24, 2017.

[43] Ji Ung S. Dynamic programming approach to a two machine flow shop sequencing with two-step-prior-job dependent setup times. 5 :126–131, 01 2007.

[44] Ibrahim H O . Pots C. Simulated annealing for permutation flow-shop scheduling. Omega, Vol 17 No 6 pp 551–557.1989

[45] Ruiz R, Maroto C, and Alcaraz J. Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, Vol 34 pp 461–476.2006 .

[46] Chuen-Lung C, Venkateswara S V, and Nasser A. An application of genetic algorithms for flow shop problems. European Journal of Operational Research.Vol 80 No 2 pp389–396.1995.

[47] Srikanth K . Barkha S. Improved genetic algorithm for the permutation flowshop scheduling problem. *Computers & Operations Research*.Vol 31 No 4 pp 593–606.2004.

[48] Eric T. Some efcient heuristic methods for the flow shop sequencing problem. European journal of Operational research, Vol 47 No 1 pp 65–74, 1990

[49] Eugeniusz N . Czesław S. A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research.Vol 91 .No1 PP 160–175.1996

[50] Jen-Shiang C, Jason Chao-Hsien P, Chien-Min L. A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem. Expert systems with applications Vol 34 No 1 pp 570–577.2008.

[51] Ching-Jong L, Chao-Tang T, and Pin L. A discrete version of particle swarm optimization for flowshop scheduling problems. Computers & Operations Research. Vol 34 No10 pp 3099–3111.2007.

[52] Qan-Ke P, Ling W, M Fatih T, Bao-Hua Z. A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion. The International Journal of Advanced Manufacturing Technology,Vol 38 No3 pp 337–347. 2008

[53] Zhigang L, Xingsheng G , and Bin J. A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Chaos, Solitons and Fractals, Vol 35 No 5 pp 851 – 861. 2008.

[54]Beasley, J. E. Or-library: Distributing test problems by electronic mail. Journal of the Operational Research Society, Vol 14, pp 1069–1072.1990.

[55]Adam J,Balas E ,Zawack,the shifting bottleneck procedure for job shop schudling.Management Sci . (34).pp 149-156.

[56]Lawrence, S. An experimental investigation of heuristic scheduling techniques. In Supplement to resource constrained project scheduling, GSIA. Pittsburgh, PA: Carnegie Mellon University 1984