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 de Master

Filière : Informatique

Option : Système d'informatique

Thème :

Etude comparative des algorithmes évolutionnaires dédier à l'optimisation multi-objectif

Encadré Par :

Dr. Zemmouchi Fares Mounir

Présenté Par :

Fadel Akila

Juillet 2019

Remerciement

Je remercie tout d'abord notre Dieu qui j'ai donné la force et la volonté pour élaborer ce travail.

J'adresse nos vifs remerciements à mon encadreur monsieur zemmouchi fares mounir pour le suivi de mon travail, pour ses lectures et pour sa grande patience.

Je remercie vont également aux membres de jury pour j'avoir honorés par leur participation à l'évaluation de ce modeste travail.

Notre reconnaissance aussi à tous ceux qui ont collaboré à mon formation en particulier les enseignants du département d'informatique, de l'université 08 mai1945Guelma.

Dédicace

Au bon DIEU le tout puissant pour qu'il m'éclaire le bon et droit chemin,

je dédie ce Modeste travail à :

Ce qui est le plus cher et proche à mon cœur, source et symbole d'amour :

Ma mère qui a sacrifié sa vie pour me rendre heureuse, que DIEU le lui accorde une longue vie pleine de bonheur et santé.

Mon père qui a sacrifié sa vie pour me rendre heureuse, DIEU le lui accorde une longue vie pleine de bonheur et santé.

A mon mari Ridha, ma fille Amira et mon fils Mouad

A mes frères et sœur : Slimane, djouhaina, Alla dine et Jmed dine. Ma belle-mère Zohra et belle-sœur Fairouz

Pour leur encouragement, que DIEU les gardes.

Et à tous mes collègues de promotion M2SIQ

2018-2019 Akila

Résumé

Les problèmes d'optimisation existants sont couramment de nature multi-objectif où plusieurs critères sont à prendre en charge conjointement. Résoudre de tels problèmes relève souvent du domaine de l'optimisation combinatoire multi objectifs.

Dans ce mémoire, le travail établit et fait une étude comparative des algorithmes évolutionnaires dédier à l'optimisation multi-objectif Appliquant une approche méta-heuristique qui est l'algorithme génétique.

Objectif est de montrer les performances de ces méthodes dans ce domaine. Nous considérons pour cela un problème difficile largement exploré qui est le problème d'ordonnancement de taches.

Mots clé :

optimisation multi-objectif, métaheuristque, algorithmes évolutionnaires, ordonnancement.

Table de matière

Table de Figure

Chapitre 1

Chapitre 2

Chapitre 3

Introduction générale

Introduction général

La discipline qui **s'**appelle "recherche opérationnelle" est l'application de méthodes scientifiques pour résoudre des problèmes complexes. L'étude a pour objectif de développer un modèle scientifique dans lequel elle tente de prédire et de comparer les résultats de différentes décisions afin d'aider le décideur à définir sa politique de manière scientifique.

L'ordonnancement joue un rôle essentiel dans de nombreux domaines d'activité tels que l'informatique (L'ordonnancement des opérations, L'ordonnancement du réseau). Les méthodes d'ordonnancement sont variées dans la littérature et diffèrent de la nature du problème à considérer, de la nature des contraintes prises en compte, des objectifs à atteindre et de la nature de l'approche décisionnelle choisie. Les problèmes d'ordonnancement sont généralement difficiles et présentent un grand avantage pratique. Dans ce contexte, nous nous intéressons aux problèmes d'ordonnancement de taches**.**

La plupart des problèmes d'ordonnancement sont si complexes que le nombre de solutions potentielles augmente de manière exponentielle avec la taille du problème. La réalisation du projet nécessite la mise en œuvre préalable de plusieurs opérations soumises à de nombreuses contraintes.

Problématique :

Le Problème est un problème d'ordonnancement, le programme doit trouver un ordre d'exécution des différentes tâches de manière à ce qu'il minimise P : la somme des priorités des tâches qui ne pourront pas être exécutées à temps.

La méthode proposée est d'utiliser un algorithme génétique. C'est une méthode d'optimisation.

Organisation du mémoire

Ce mémoire est organisé en trois chapitres présentant respectivement, l'état de l'art, notre proposition et enfin une réalisation de l'application de l'algorithme génétique.

- **Le premier chapitre :** Le travail présenté dans ce chapitre exposé donne un aperçu sur l'optimisation multi objective et ses différentes méthodes de résolution

- **Le deuxième chapitre :** Dans ce chapitre nous allons nous focaliser sur un problème classique et connu dans la recherche opérationnelle qui est le problème d'ordonnancement.
- **Le troisième chapitre** : Ce chapitre contient les détails du modèle proposé pour une approche basée méta-heuristique, Il contient aussi l'implémentation avec des aperçus montrant les différentes interfaces de l'application.

Nous terminerons notre mémoire par une conclusion générale qui résume notre contribution et quelques perspectives à notre travail.

Chapitre 1 :

Optimisation multi objectifs

1 Introduction :

Depuis une trentaine d'années, le domaine de l'optimisation multi objective connaît une évolution importante. Cette évolution s'est traduite par le développement d'un grand nombre de méthodes. La multitude des méthodes d'optimisation multi objective est perçue comme une richesse incontestable de ce domaine. D'ailleurs, certains la justifient par la diversité des problèmes ainsi que par l'existence de différentes approches de résolution possibles et légitimes de ces problèmes.

Les ingénieurs butent quotidiennement sur des problèmes technologiques de complexité grandissante, qui surgissent dans des secteurs très divers. Le problème à résoudre peut fréquemment être exprimé sous la forme générale d'un problème d'optimisation, dans lequel on définit une fonction objective, où la fonction Coût qu'on cherche à minimiser (ou à maximiser) par rapport à tous les paramètres concernés. Le travail présente dans ce chapitre exposé un aperçu sur l'optimisation multi objective et ses différentes méthodes de résolution.

2 Qu'est-ce qu'un problème d'optimisation ?

Il existe deux types de méthodes d'optimisation, la première est l'optimisation mono-objectif, qui se base sur la minimisation (ou la maximisation) d'une seule fonction objective où le but est de trouver la meilleure solution appelée solution optimale qui est facilement définie suivant une seule performance du problème étudié d'une part.

D'autre part, l'optimisation multi-objectif optimise simultanément plusieurs fonctions objectives qui sont souvent contradictoires, on cherche de trouver la meilleure solution suivant un ensemble de performance du problème.

2.1 Définition de l'optimisation :

Selon **[Paschos, 2005]** une définition générale a été effectuée à ce terme : « 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'extraire les propriétés structurelles et de caractériser les solutions du problème.

Enfin, l'optimisation c'est l'art d'exploiter cette caractérisation afin de déterminer les algorithmes qui les calculent mais aussi, de mettre en évidence les limites sur l'efficience et

l'efficacité de ces algorithmes ». Un problème d'optimisation se définit comme la recherche du minimum ou du maximum d'une fonction donnée.

Un problème d'optimisation se présente mathématiquement sous la forme suivante :

Minimiser	f(x)	(fonction à optimiser)
Avec	g(x)<= 0	(m contraintes d'inégalité)
Et	h(x)=0	(p contraintes d'égalité)

2.2 Concepts principaux en optimisation [smairi, 2013]

Nous allons définir les notions communes à n'importe quelles méthodes d'optimisation multi objectifs :

(a) Fonction objective

C'est la fonction qui modélise le but à atteindre dans le problème d'optimisation sur l'ensemble des critères, elle est appelée critère d'optimisation ou fonction d'adaptation. Il s'agit de la fonction qui doit être optimisée.

(b) Paramètre

C'est la variable qui exprime les données quantitatives et qualitatives sur un problème. Ces paramètres correspondent aux variables de la fonction objective, ils sont ajustés pendant le processus d'optimisation afin d'obtenir les solutions optimales. Ce sont les variables de conception ou du projet appelé aussi variable d'optimisation.

(c) Vecteur de décision

C'est le vecteur qui correspond à l'ensemble des variables du problème.

(d) Critère de décision

Peut-être une variable du problème ou une combinaison de variable. C'est le critère sur lequel sont jugés les vecteurs de décision pour déterminer le meilleur.

(e) Contrainte

C’est la condition que doivent respecter les vecteurs de décisions du problème.

(f) Espace de recherche

C’est l’ensemble des valeurs pouvant être prise par les variables.

(g) Espace réalisable

Correspond à l’ensemble des valeurs des variables satisfaisants les contraintes.

(h) Espace objectif

Représente les ensembles des images de l’espace de recherche à déterminer par les valeurs des fonctions objectives.

2.3 Classification d’optimisation

Selon **[Xin, 2010]** plusieurs critères sont employés pour classifier les problèmes d’optimisation :

- **Classification basée sur le nombre d’objectifs**

Cette classification consiste à définir deux catégories du problème d’optimisation : ceux à un seul objectif et ceux a plusieurs objectifs mais, la majorité des problèmes réels sont multi objectifs en terme de la fonction.

- **Classification en terme de fonction**

Si les fonctions de contrainte et la fonction objective sont linéaires donc, on est devant un problème d’optimisation linéaire autrement, on parle d’un problème non linéaire.

- **Classification selon les contraintes**

Dans cette classification on constate deux catégories qui sont : les problèmes sans contraintes et les problèmes avec contraintes d’égalité ou inégalité ou sa peut être les deux à la fois.

- **Classification selon l’allure de la fonction objective**

Qui peut admettre un seul optimum local qui est aussi l'optimum global, et donc ici on parle d'un problème d'optimisation uni modal tant dis que si la fonction objectifs admis plusieurs on parle alors d'un problème multimodal.

- **Classification selon les types de variables de décision**

Lorsque les variables de décision sont discrètes on parle de problème d'optimisation combinatoire (discret).

- **Le problème d'optimisation déterministe**

Qui se reflète dans la définition exacte des variables de décision et de fonctions objectifs. Le problème alors devient stochastique.

2.4 Formulation du modèle d'optimisation multi objectif [kaidi & chelouti, 2016]

Auparavant le problème d'optimisation multicritère a été étudié par deux communautés qui sont : la communauté de recherche opérationnelle et la communauté de contrôle.

Les problèmes réels prennent en charge les multiples mesures de performance qui doivent être optimisé. En pratique ce n'est pas toujours possible parce que les objectifs pouvant être conflictuels. Dans ce que cas, la qualité d'un individu décrite par un vecteur.

Un problème d'optimisation multicritère consiste à trouver le vecteur de décision idéale tel que la fonction objectif est optimal. Ces objectifs multiples son souvent concurrente ou lors de l'amélioration de l'un entraine la détérioration de l'autre ou des autres.

Dans les problèmes multi objectifs, l'optimum n'est plus une simple valeur mais un ensemble de point représentant les meilleurs compromis et ce n'est ce qui fait la différence entre les problèmes mono et multi objectifs.

Selon **[Yang, Chen & Yeo, 1999]** et **[Oliver, 2000]** la formulation d'un modèle d'optimisation multi objectif se fait comme suit :

1- Fonction objective

Une ou plusieurs fonction(s) objectif(s) à optimiser (maximiser ou minimiser). La fonction objective représenté en décide le but à atteindre.

2- Espace de travail et variable de conception

L'espace de travail c'est l'espace de notre recherche et les variables de conception se sont les variables qui seront optimisées par la suite.

3- Contraintes.

Sont des contraintes d'égalités ou d'inégalités et permettent en général de limiter l'espace de travail (la solution réalisable).

3 Méthodes de résolution des problèmes d'optimisation multi objective [groupe Gotha, 1993]

La résolution de différents types de problème dans la vie quotidienne à pousser les chercheurs à proposer des méthodes de résolution pour améliorer les performances et la qualité de la solution proposée, c'est pourquoi de nombreuses méthodes de résolution ont été proposées.

Selon **[Gherboudj,2013]**, cette variété et ces différences ont permis de regrouper les différentes méthodes de résolution de différents problèmes en deux sous classes principales qui sont :

- La classe des méthodes exactes ;
- La classe des méthodes approchées.

Et l'hybridation de ces deux classes donne une nouvelle méthode qui s'appelle la méthode hybride.

3.1 Méthodes exactes [kaidi & chelouti, 2016]

Il y a très peu de travaux sur les méthodes exactes dans le contexte de la résolution des problèmes d'optimisation multi-objectif, sans doute, à cause de la grande difficulté de ce type de problème.

Les références existantes et qui présentent la plupart des méthodes exactes sont **Ulungu** et **Ehrgott et Gandibleux 2000.**

- **L'agrégation linéaire**

Cette méthode populaire transforme le problème multi-objectif en un problème mono objectif en combinant linéairement les différents objectifs. Ainsi, le nouveau problème obtenu, car il s'agit alors d'un problème différent, consiste à optimiser P i λi fi Le théorème de Geoffrion indique qu'en utilisant différentes valeurs pour le vecteur λ, il est possible d'obtenir toutes les solutions supportées du problème multi-objectif initial. Par contre, aucune solution

non supportée ne peut être trouvée par cette méthode. La méthode d'agrégation linéaire a donc ses limites. Toutefois, elle est intéressante pour des problèmes ayant d nombreux objectifs et/ou un grand nombre de solutions supportées bien réparties. Dans ce contexte, il peut être suffisant de générer les solutions supportées.

3.2 Méthode e contrainte

Cette méthode permet de transformer le problème d'optimisation multi objectif en un problème mono objectif. La méthode consiste à convertir (m - 1) des m objectifs du problème en contraintes et d'optimiser séparément l'objectif restant.

3.3 Les méthodes heuristiques

Ce sont des règles empiriques qui se basent sur l'expérience et qui nécessite une connaissance sur le domaine du problème traité ainsi que les résultats acquis afin d'améliorer les recherches, Généralement, on n'obtient pas la solution optimale mais une solution approchée

- **L'algorithme glouton**

construction d'une solution réalisable en se ramenant à une suite de décisions qu'on prend à chaque fois au mieux en fonction d'un critère d'optimisation local sans remettre en question les décisions déjà prises. Généralement, la solution obtenue est approchée.

Intérêt : algorithmes simples à implémenter.

Défauts :solutions approchées obtenues plus ou moins bonnes, critère local

- **Recuit simulé :**

Donne généralement de bonnes solutions par rapport aux algorithmes de recherche classiques, peut être utilisé dans la plupart des problèmes d'optimisation, il converge vers un optimum global (lorsque le nombre d'itérations tend vers l'infini..

3.4 Méthodes à base de populations ou distribuées

- **Algorithmes hybrides évolutionnistes :**

Résolution des problèmes complexe, parallélisassions, générique et adaptable

- **Algorithme de colonies de fourmis :**

Dû à **M. Dorigo** dans les années 90, s'inspire du comportement collectif des fourmis pour la recherche d'un plus court chemin de la fourmilière à un point de nourriture.

4 Exemples de problèmes d'optimisation [Gherboudj, 2013]

Les problèmes d'optimisations peuvent être des problèmes académiques ou industriels. Enfaite, sur le plan pratique les problèmes industriels sont des projections des problèmes académiques.

Parmi les problèmes d'optimisation académique on a le problème de satisfiabilité booléenne Max-Sat, le problème d'ordonnancement, le problème de coloriage de graphe. En outre, parmi les problèmes industriels, nous citons alors les problèmes de design dans les domaines d'ingénierie, les problèmes d'investissement financier, la planification de trajectoire de robots mobiles …etc

5 Les algorithmes évolutionnaires :

Pour résoudre les problèmes d'optimisation avec plus que des objets, les chercheurs ont proposé plusieurs algorithmes d'évolution à objectifs multiples pour traiter ces problèmes.

Les MOEA sont principalement classés en quatre catégories principales, à savoir, basée sur la :

- Décomposition ;
- La dominance de Pareto ;
- La coévolution de plusieurs populations ;
- L'approche basée sur des indicateurs.

Les chercheurs ont identifié le besoin de développer des algorithmes qui adresseront les MaOP. Plusieurs de ces algorithmes trouvés dans la littérature.

He et **Yen** ont présenté et comparé les approches de visualisation utilisées dans des problèmes d'optimisation à objectifs multiples. Ils ont classé ces approches en cinq catégories différentes. Il s'agit de la visualisation basée sur :

- Le système de coordonnées parallèles ;
- La visualisation des coordonnées radiales ;
- La visualisation basée sur la recherche d'informations locales ;
- La visualisation de coordonnées polaires ;

- La visualisation basée sur des modèles de substitution.

Le but des approches de visualisation est de visualiser la population dans des problèmes d'optimisation à objectifs multiples pour évaluer les performances de l'algorithme et pour la prise de décision.

5.1 Squelette d'un algorithme ´évolutionnaire :

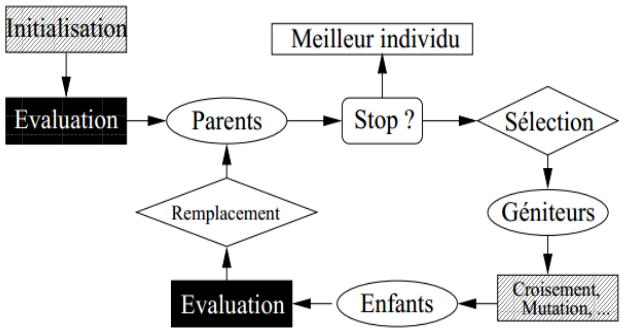


Figure 1.1 : Squelette d'un algorithme ´évolutionnaire

5.2 Quelques algorithmes évolutifs utilisés pour résoudre des problèmes d'optimisation multi objectifs :

- Algorithme d'évolution multi-objectif ;
- Algorithme génétique élitiste basé sur le regroupement ;
- Algorithme d'optimisation à objectifs multiples évolutif ;
- Algorithme génétique **: [Naziha, 2011]**

Les algorithmes génétiques (AGs) ont été mis au point par John Holland de l'université du Michigan aux États-Unis dans les années 60 **[Holland ,1962]**. Ils ont été ensuite raffinés par De Jong **[DeJong,1975]** et popularisés par Goldberg et Holland **[Goldberg 1988, Goldberg 1989]**.

Le principe des algorithmes génétiques est de simuler le processus d'évolution des espèces dans leur milieu naturel, en s'inspirant des théories de l'évolution proposées par **Charles Darwin**.

Les types de représentation (codage) les plus connus de l'algorithme génétique sont :

Le codage binaire et le codage réel.

- La recherche par harmonies ;
- L'intelligence par essaim ;

- L’algorithmes de la sélection clonale ;
- L’algorithme de colonies de fourmis ;
- La recherche coucou ;
- L’algorithme de l’optimisation par coucou ;

6. Qualité de service

La qualité de service fournir des garanties, répondre aux besoins correspondent au concept de la qualité on d’écrit ce concept, propre aux réseaux “classiques”, a travers de ses différentes implémentations, choisir ceux coïncidant avec nos attentes, et ainsi les adapter aux réseaux. De nos jours, l’Internet achemine des paquets en provenance de toutes sortes d’applications : des applications d’accès à un serveur (Mail, FTP, etc.), des applications reliant plusieurs utilisateurs (téléphonie, etc.), et bien d’autres. Quelles qu’elles soient, ces applications détiennent des marges de valeurs tolérées par rapport aux délais de transmission, aux variations de ces délais, à l’intégrité des données transportées. C’est sous ces différents aspects que se traduit la qualité de service. Elle consiste donc à fournir des garanties (latence, débit, taux d’erreur, etc.) aux différentes applications selon les besoins qu’elles auront exprimés Un mécanisme de qualité de service devra donc être capable de fournir des garanties a une application tout en assurant que les besoins exprimés par les autres applications soient toujours satisfaits **[Jérome,2007]**

De nombreux efforts de recherche sur la qualité de service ont été réalisés. Beaucoup d’entre eux portent par exemple sur les réseaux IP et le domaine de l’Internet [**Lochin,2004**]. Les approches à l’origine, le protocole IP n’était pas prévu pour prendre en compte les nombreux aspects liés `a la qualité de service. Le seul service fourni, appelé Best Effort, ´était minimaliste. Concrètement, son objectif ´était de maximiser l’utilisation des ressources et de simplifier l’utilisation le fonctionnement des ´équipements d’interconnexions. Cependant, les d’édits des réseaux ont grandement évolué, et les applications de l’Internet également : de nouveaux besoins sont apparus (voix sur IP, etc.) et nécessitent une réelle qualité de service. Deux approches ont été proposées par l’IETF (Internet Engineering Task Force) pour gérer la bande passante et fournir une qualité de service de bout en bout. La première, IntServ, propose une garantie stricte : après négociation avec les gestionnaires de qualité de service déterminant si le service peut être garanti, les ressources sont réservées, offrant un lien de communication `a qualité constante en terme de débit et de latence. Ce schéma, bien que performant, se heurte au problème du passage à l’´échelle. En effet chaque élément d’interconnexion doit garder

en mémoire l’´etat des flux qui le traversent, ce qui est inconcevable en raison de la multitude de flux circulant sur Internet. Afin de résoudre ces problèmes,

l’IETF a proposé une autre solution : l’architecture DiffServ [**Blak,Black,Carlson,Davies,Wang et Wauss,1998**]. Ce modèle propose de séparer le trafic par flots mais par classes de flots (Behaviour Aggregate). Ce qui limite ainsi le nombre de comportements au niveau de chaque nœud. De cette manière, chaque paquet est doté d’une étiquette qui déterminera le traitement (Per Hop Behaviour) que celui-ci obtiendra du réseau (règles de mise en file d’attente, d’ordonnancement, lissage).

6.1. Ordonnancement et qualité de service :

Les algorithmes d’ordonnancement Quelle que soit la politique de qualité de service à laquelle on a recours, le traitement s’effectue toujours au niveau des listes des paquets en attente. Dans cette partie nous allons décrire plusieurs algorithmes d’ordonnancement utilisés dans les routeurs [**Toumi,2002**] [**Napoléon, Carvajal,2001**].

Fisrt In First Out (FIFO) C’est le plus simple algorithme. Les paquets sont délivrés suivant leur ordre chronologique d’arrivée. Il ne permet pas de différencier les paquets entre eux mais il a l’avantage d’être simple et donc rapide.

Priority Queuing Une priorité est affectée à chaque file d’attente du routeur. Chaque paquet est réparti selon sa priorité et le routeur desservira en premier lieu les paquets de plus haute priorité. Cela permet de gérer le trafic critique mais pour les flux de basse priorité se pose alors un problème de famine (lorsque le trafic haut priorité est important.

Round robin Pareillement à Priorité Queuing les paquets sont répartis dans un certain nombre de files suivant leur classe, ensuite, un tourniquet alterne les paquets à servir parmi les files présentes.

Weighted Fair Queuing Cet algorithme est une émulation du Round Robin bit `a bit. Les paquets sont donc répartis dans des files d’attente et sont ensuite servis bit `a bit, et de façon circulaire. Chaque file peut être pondérée pour donner plus ou moins de bande passante `a chaque classe de flots. Weighted Fair Queuing est relativement couteux en ressources car le routeur doit calculer `a chaque émission combien de paquets de chaque source seront émis. La gestion des files d’attente Un autre point relatif aux files d’attente des routeurs est la gestion de ces files, notamment en ce qui concerne la prévention de la congestion. Essentiel dans le domaine de l’Internet.

6.2. Ordonnancement et stratégies

Au cœur de cette mécanique se trouve donc le moteur d'optimisation de New Madeleine. C'est en son sein que se déroule la fonction chargée de sélectionner (ou de générer, par exemple avec une accumulation des données) le prochain paquet à soumettre à chacune des unités inactives. De nombreux arguments potentiels paramètrent cette fonction : la taille de la fenêtre de travail, les informations propres à chacun des paquets, les caractéristiques nominales et fonctionnelles du réseau, ainsi que la politique d'ordonnancement des paquets. Tous les flux applicatifs vont être projetés sur les unités de multiplexage logiques selon ces différents paramètres. Les messages courts seront par exemple envoyés directement sur le réseau, alors que les messages dont la taille excède un certain seuil vont être stockés et transférés plus tard en attendant que le récepteur confirme qu'il est prêt à le recevoir (protocole de type rendez-vous). Face à autant de paramètres, il est possible d'user de plusieurs tactiques d'optimisations, mais combiner de façon efficace ces tactiques demeure un problème difficile. Les développeurs ont alors choisi de proposer un ordonnanceur programmable : SchedOpt. La fonction d'optimisation globale est alors définie au travers de l'écriture de stratégies. Chacune de ces stratégies implémente une liste d'opérations élémentaires organisant le comportement global de l'ordonnancement selon un objectif particulier.

Concrètement les stratégies réunissent un certain nombre de méthodes appelées au cœur de l'ordonnanceur. L'objectif de ces stratégies est de définir un schéma d'ordonnancement optimal en fonction de la politique attribuée. C'est notamment le rôle d'une stratégie de mettre en œuvre le protocole rendez-vous et de définir le seuil associé. Une des fonctions des stratégies est de traiter l'envoi d'un paquet. C'est à cet endroit qu'est prise la décision d'envoyer une demande de rendez-vous avant d'émettre les données ou bien de placer les données dans la liste des paquets prêts à être transmis. Lorsqu'une des cartes réseau est disponible, l'optimiseur fait appel à la stratégie en cours pour qu'elle effectue sa passe d'optimisation (dans laquelle elle peut agréger certains paquets avec d'autres pour former la requête) et ainsi obtenir le prochain paquet à transmettre à la carte libre **[Jérome,2007].**

7. Conclusion :

Dans ce chapitre, on a décrit les approches de résolution et de l'optimisation multi objectifs pour la résolution des problèmes multi objectifs, ainsi les différentes qualités de service.

Dans le prochain chapitre, on va décrire une vue détaillée sur un domaine de recherche opérationnelle très actif qui est le problème d'ordonnancement et donnée des méthodes pour résoudre ce genre des problèmes.

Chapitre 2 :
Problème d’ordonnancement

Chapitre 2 : problème d'ordonnancement

1. Introduction

Dans ce chapitre nous allons nous focaliser sur un problème classique et connu dans la recherche opérationnelle qui est le problème d'ordonnancement.

Dans notre travail, nous nous limitons aux algorithmes génétiques pour démontrer l'objectif essentiel de cette étude qui concerne l'application de ce type d'algorithme évolutionnistes pour la résolution d'un problème multi objectives.

2. Problème d'ordonnancement [EL Hilali & al, 2009]

2.1 Concepts et définitions

Le problème d'ordonnancement consiste à ordonner dans le temps un ensemble de taches compte tenue de contraintes temporelle tel que les contraintes de délais, d'enchainement et des contraintes portant sur la disponibilité et l'utilisation des ressources requises.

L'objectif est de minimiser la durée de réalisation compte tenu des contraintes reliant les différentes taches.

- **Une tâche :**

Est une entité élémentaire localisée dans le temps par une durée d'exécution, une date limite d'exécution et une période dont la réalisation et constitué d'un ensemble d'opérations qui requiert pour son exécution certaine ressource dans laquelle il est nécessaire de la programmer de façon à optimiser certain objectif.

- **Les contraintes :**

Expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision on distingue :

- Des contraintes temporelles ;
- Des contraintes de ressources.

Les problèmes d'ordonnancement touchent tous les domaines de l'économie. Les méthodes de résolutions des problèmes d'ordonnancement puisent dans toutes les techniques d'optimisation multi objectif, ces méthodes garantissent l'optimisation de la solution fournie.

Notre objectif est de montrer les performances des algorithmes génétiques (métaheuristque) dans la résolution de ce genre de problème.

2.2 Variante des problèmes d'ordonnancement

Selon la nature des variables et des contraintes mise en jeu, plusieurs variantes des problèmes d'ordonnancement sont proposées dans la littérature

1) Utilisation des réseaux de pétri pour l'ordonnancement hors-ligne optimal des systèmes temps réel. **[Grolleau et Choquet-Geniet,2000].**

- Une méthode d'ordonnancement hors-ligne de système de taches périodique temps réel.

2) Nouvelle méthodes pour les problèmes d'ordonnancement cyclique : **[touria chafaquan ben rahho, 2013].**

- Approche par réseau de pétri.
- Approche par programmation linéaire.
- Approche par théorie des tas.

3) Algorithmique parallèle hétérogène et technique d'ordonnancement **:[olivier et yves,2003].**

- Approche statique et dynamique.

4) Ordonnancement des ressources humaines : étude du cas d'une entreprise d'injection plastique : **[Laur et Pière, 2001].**
5) Contribution au raisonnement progressif et temps réel dans un univers multi-agents : **[abdel-illah,1993].**
6) Un algorithme génétique pour l'ordonnancement robuste : application au problème du flow shop hybride **[Tarek Chaari,2011].**
7) Méthodologie d'aide à la décision multi objectif pour l'ordonnancement d'ateliers discontinus **[M. Antonio & M. Arsène, 2007].**
8) Problèmes d'Ordonnancement De Projeter Méta heuristiques **[badre. E,2016].**

3. Méthodes de résolution des problèmes d'ordonnancement :

Les méthodes de résolution des problèmes d'ordonnancement puisent dans toutes les techniques d'optimisation multicritère, ces méthodes garantissent l'optimisation de la solution fournie. Les algorithmes dans la complexité ne sont pas polynomiaux ne pouvant pas être utilisé pour les problèmes de grandes tailles, d'où la nécessité d'utiliser ces méthodes approchée efficace pour ces problème NP-difficile.

3.1. Méthodes de résolution approchées :

Ce sont des méthodes qui sacrifient le caractère optimale de la solution pour obtenir des solutions optimale de bonne qualité en un temps de calcule raisonnable, Ces méthodes reposent généralement sur un mécanisme de déplacement aléatoire ou non elles ne sont pas exactes mais permettent en général d'obtenir des solutions proche de l'optimum.

3.2. Les méta-heuristiques : [Oumar.k, 2009]

Une méta-heuristique désigne un schéma algorithmique général qui peut s'appliquer à différents problèmes d'optimisation combinatoire.

Plus précisément, elle utilise des stratégies qui guident la recherche dans l'espace des solutions, ces stratégies étan**t** indépendantes du problème auquel on les applique.

Le but est d'explorer le plus efficacement possible l'espace des solutions afin de ne pas rester bloqué dans les minimas locaux et de se diriger rapidement vers les régions les plus prometteuses.

Il existe un grand nombre de méta-heuristiques allant de schémas très simples (qui mettent en œuvre des processus de recherche basiques, comme la descente), à des schémas beaucoup plus complexes (avec des processus de recherche élaborés comme les colonies de fourmis).

Il existe plusieurs méta-heuristique comme la recherche locale, le recuit simulé, la recherche tabou, l'optimisation par colonies de fourmis, les algorithmes génétiques.

3.3. Algorithme génétique : [Merhoum.K, Djeghaba.M,2006]

Les Algorithmes génétiques s'attachent à reproduire l'évolution naturelle d'individus en respectant la loi de survie énoncée par **Darwin**.

Les principes de base des Algorithmes génétiques développés initialement par **Holland** pour répondre à des besoins spécifiques en biologie, ont été rapidement appliqués pour résoudre et avec succès les problèmes d'optimisation multicritère en recherche opérationnelle et les problèmes d'apprentissage dans le domaine de l'intelligence artificielle.

Portmann, introduit et explique comment appliquer les algorithmes génétiques aux problèmes d'ordonnancement.

Dans le cadre de l'application des algorithmes génétique dans un problème d'optimisation multi objectif, une analogie est développée entre un individu dans une population et une solution d'un problème dans l'espace de solutions globale.

Dans le cadre de l'optimisation et par analogie, une solution (individu) est codée par une structure de données qui représente son empreinte génétique, on associe à chaque individu une valeur de la fonction critère, celle-ci permet de lui attribuer sa force d'adaptation « fitness » pour simuler la sélection naturelle, de sorte que les solutions pour lesquelles la valeur du critère est plus éloignée de la valeur optimale soient l'exploration de l'espace de solution s'effectue grâce à l'application des mécanismes des opérateurs génétique le croisement et la mutation sur une population initiale définit aléatoirement.

Leur déroulement sur plusieurs générations dont le nombre est fixé par le critère d'arrêt, nous permet d'espérer obtenir une solution proche de la solution optimale, ou même une solution optimale le déroulement de l'AG standard peut être résumé comme suit :

- La génération de la population initiale.
- Sélection.
- Reproduction (croisement et mutation).
- Remplacement par la nouvelle population.

4. Structure d'un algorithme génétique :

4.1. Génération de la Population Initiale

Plusieurs méthodes existent pour définir le mécanisme de la génération de la population initiale qui représente le point de départ pour la constitution des générations futures : le tirage aléatoire, les heuristiques ou une combinaison de solution heuristique et aléatoire.

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

4.2. Sélection

La phase de sélection consiste à choisir parmi les N individus i de la population courante les plus forts individus à partir desquels la génération suivante sera créée.

Soit pour un ordonnancement i réalisable, on calcul la valeur de la fonction objective pour notre problème c'est effectuer un ordonnancement des taches.

On associe à chaque solution une fonction d'évaluation pour calculer sa force d'adaptation, dans notre cas le problème a pour but de minimiser la fonction objective, Les individu ainsi sélectionnés constituent une population intermédiaire.

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

4.3. Croisement

L'opérateur de croisement est le plus important dans les AGs, il permet d'explorer efficacement l'espace de recherche.

L'opérateur de croisement appliquer sur deux individu parent1 et parent2 choisis parmi la population sélectionnée, permet de générer deux nouvelles solutions enfant1 et enfant2 par combinaison des propriétés des parents 1 et 2 .

4.4. Mutation

Le deuxième opérateur génétique important est la mutation, elle vient en deuxième place sur le plan d'importance par rapport au croisement.

Une mutation est une perturbation introduite sur la composante de l'individu afin de garantir la diversité et élargir le champ d'exploration.

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

4.5. Remplacement

Cette étape constitue la population de la génération suivante à partir des parents et des enfants de la génération courante.

Une fraction de la population est remplacée par sa descendance à chaque génération.

L'écart entre les générations indique la proportionnel de parents remplacés par des enfants.

4.6. Critère d'arrêt

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

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

4.7. Organigramme d'algorithme génétique

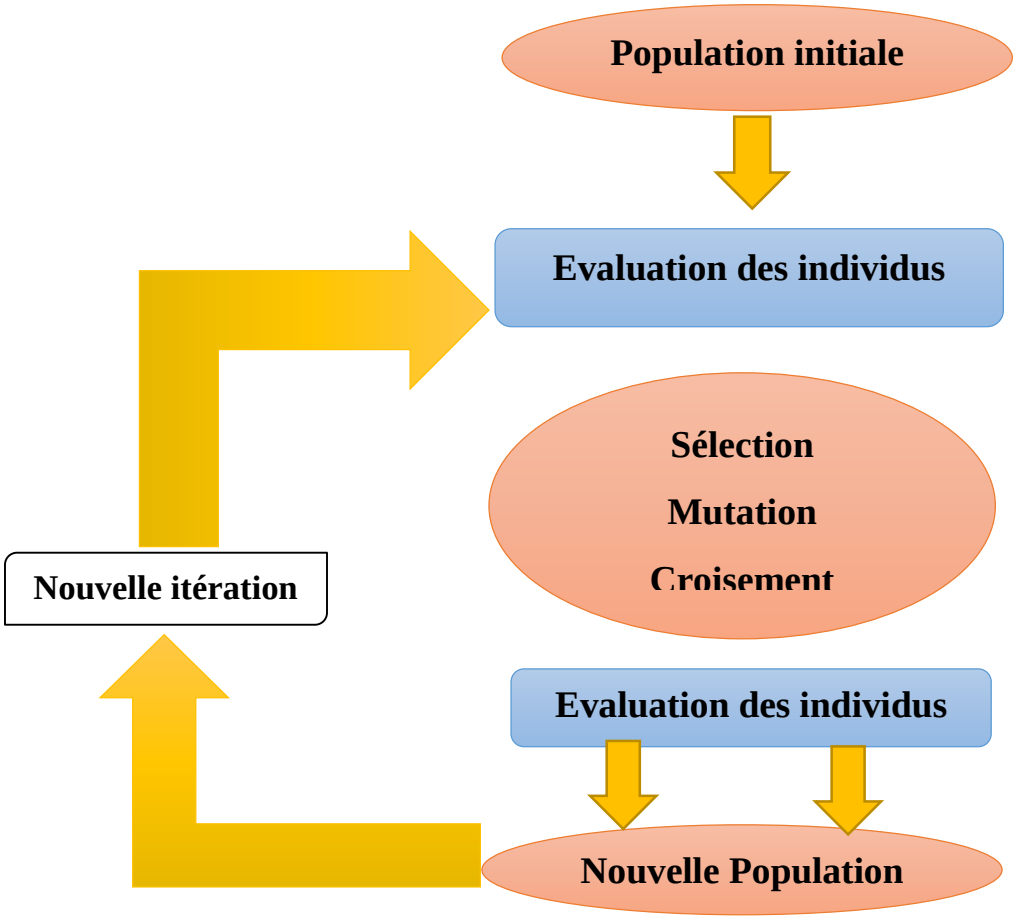


Figure2.1 : Diagramme d'algorithme génétique.

5. Conclusion :

Au témoignage du nombre important d'article qui ont été publiés sur les algorithmes génétique, Ces derniers sont très bien adaptés aux problèmes d'optimisation multi objectif. Ce domaine est très dynamique et ne cesse de se développer

Chapitre 3 :

Conception et implémentation

Chapitre 3 : conception et implémentation

1. Introduction

Dans ce chapitre nous allons présentons les différentes étapes réalisées durant la modélisation et l'implémentation de notre application ensuite nous définissons les fonctionnalités de notre système et enfin nous exposons sur les résultats obtenus.

2. Le sujet

On considère des tâches ayant chacune une durée d'exécution, une date limite d'exécution et une priorité.

La durée d'exécution correspond au temps qu'un opérateur va pouvoir mettre pour l'exécuter, la date limite d'exécution correspond à la date avant laquelle la tâche devra être terminée, la priorité correspond à l'importance de cette tâche à être exécutée avant la date limite d'exécution, c'est une valeur que j'ai défini entre 0 et 9.

Le Problème est un problème d'ordonnancement, le programme doit trouver un ordre d'exécution des différentes tâches de manière à ce qu'il minimise P : la somme des priorités des tâches qui ne pourront pas être exécutées à temps.

Le nombre de tâches étant très important, on cherche un ordre qui ne soit pas forcément le plus adapté, mais un ordre qui soit parmi les mieux adaptés.

La méthode proposée est d'utiliser un algorithme génétique. C'est une méthode d'optimisation.

3. Objectif du travail :

Les méthodes de résolutions des problèmes d'ordonnancement puisent dans toutes les techniques d'optimisation multicritère.

Ces méthodes cherchent à garantir l'optimisation de la solution fournie. La méthode optée dans notre travail est l'application des algorithmes génétiques.

Ces derniers appartiennent à la famille des algorithmes évolutionnistes.

Notre objectif est de montrer les performances de ces méthodes pour la résolution de problème d'ordonnancement de taches.

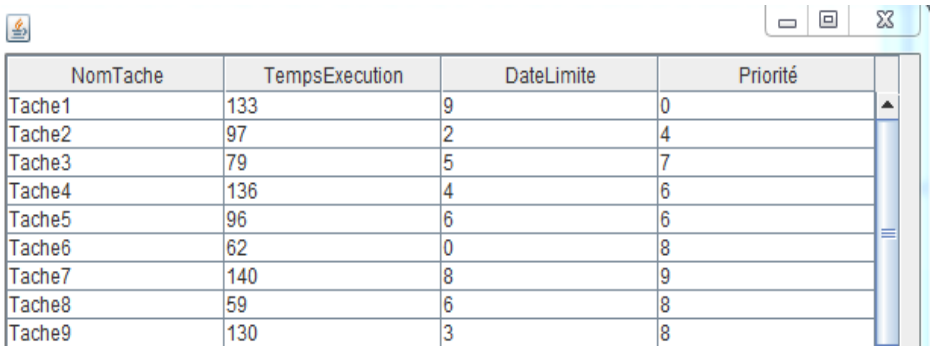
Le nombre de ces dernières étant très important, notre système doit trouver un ordre d'exécution des différentes taches de manière à ce qu'il minimise la somme des priorités des taches qu'ils ne pourront pas être exécutés à temps

4. Base de données

Notre système a une base de données dont l'utilité est de stocker les données nécessaire relié au fonctionnement de notre système.

Cette base de données contient un ensemble de taches à ordonner qui sont généralement conflictuels avec des critères multiples qui sont : temps d'exécution, date limite d'exécution et la priorité.

Les paramètres définissants les tâches (durée d'exécution, une date limite d'exécution et une priorité) sont définis aléatoirement et peuvent donner une solution « toutes les taches sont exécutées à temps » pour une quantité de l'ordre de 30 tâches, il est toujours possible de remplir ces paramètres soi-même.



NomTache	TempsExecution	DateLimite	Priorité
Tache1	133	9	0
Tache2	97	2	4
Tache3	79	5	7
Tache4	136	4	6
Tache5	96	6	6
Tache6	62	0	8
Tache7	140	8	9
Tache8	59	6	8
Tache9	130	3	8

Figure 3.1 : Clin d'œil sur la base de données

5. Modélisation

Nous intéressons à un problème d'ordonnancement, notre application doit trouver un ordre d'exécution des différentes taches de manière à ce qu'il minimise la somme des priorités des taches qu'ils ne pourront pas être exécutées à temps, le nombre de taches étant très important on cherche ordre qui ne sois pas forcément le plus adapté mais qui soit parmi les mieux adaptés.

5.1.Modélisation de l'individu

Une solution représenté sous forme d'individu tel que ce dernier est un vecteur de tache trié.

Figure 3.2 : représentation de l'individu

6. Fonctionnement du système

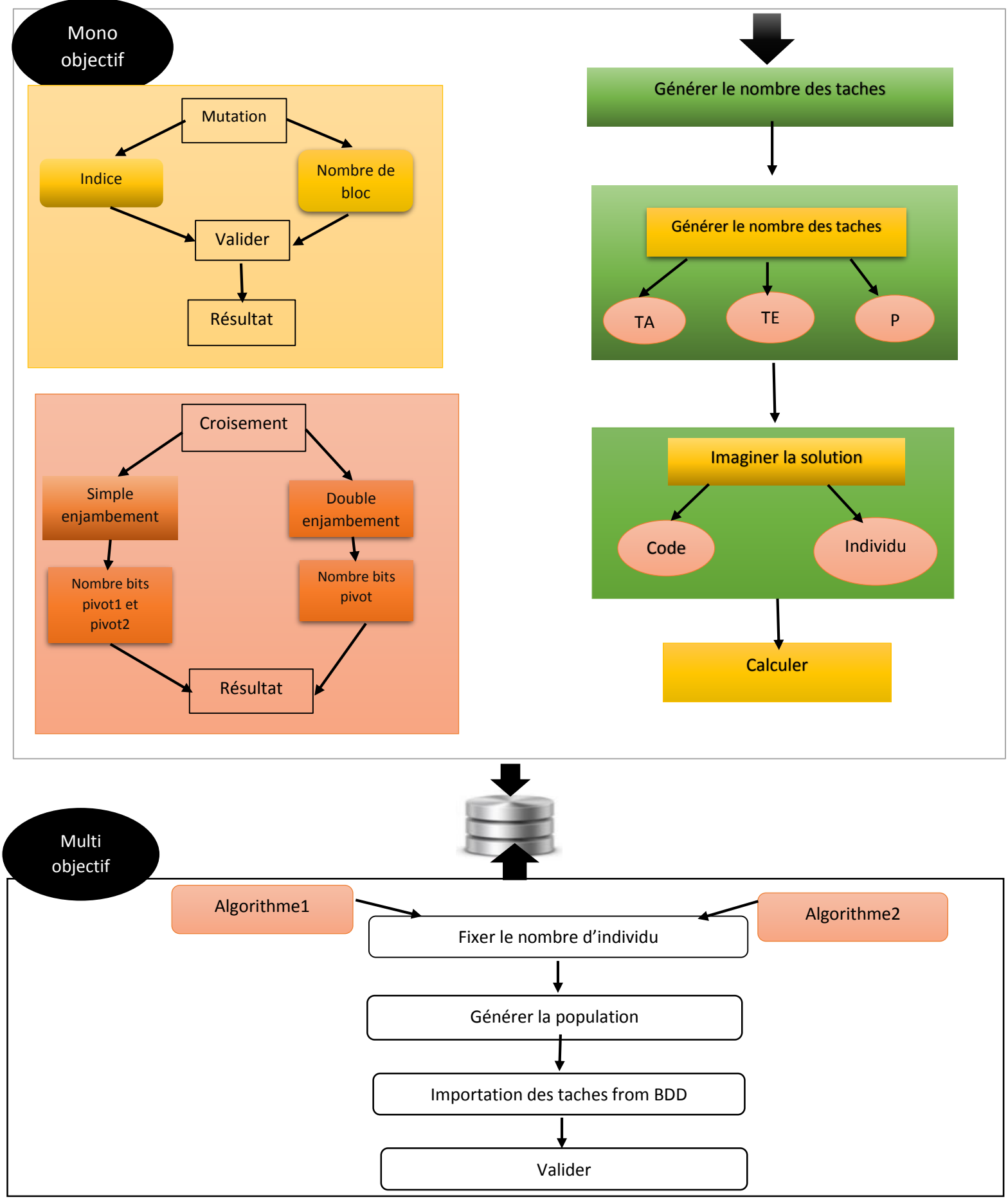


Figure 3.3 : Schéma du processus général du système proposé.

6.1.Scénario du fonctionnement de système

On considère une population d'individus, dans notre cas un individu correspondra à un ordre d'exécution de taches.

Du point de vue génétique l'ordre d'exécution est à chromosome (individu) que la tâche est au gène.

En génétique les individus sont des combinaisons de gènes appartenant à un « alphabet » génétique de quatre gènes, dans notre cas l'alphabet génétique correspond aux différentes tâches. Une solution potentielle sera donc un ordre (ou individu).

En utilisant le pseudo code suivant :

Début

Générer population_initiale.

Calculer fitness_individu.

Tant que « critère d'arrêt non satisfait » **Faire**

Générer Population_Sélectionner.

Générer Population_Croisement.

Générer Population_Muté.

Générer Population_Remplacement.

Calculer fitness_individu.

Fin tant que

Fin.

- A l'initialisation on génère une population de N individus, chaque individu est créé aléatoirement de manière à ce que la population soit répartie uniformément sur l'ensemble de recherche. Ensuite trois opérateurs vont être appliqués pour générer la population suivante.

- **L'opérateur de sélection :**

Dans un premier temps on va sélectionner les individus suivant leur adaptation au problème, à chaque individu est associé une fitness qui est une probabilité fonction de P : un individu ayant une fitness élevée (P faible) aura plus de chance d'être sélectionné qu'un individu de fitness moins élevée.

• **L'opérateur de croisement :**

Il permet de croiser deux individus entre eux : création d'un individu fils à partir de deux individus parents.

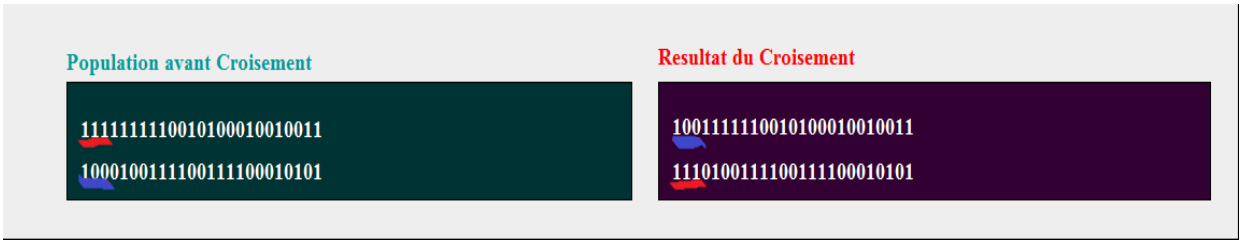
Cet opérateur retournera une population fille à une population mère. Lors de cette opération, deux individus s'échangent des parties de leurs chaînes binaires, pour donner de nouveaux individus.

Ces enjambements peuvent être simples ou multiples.

Dans le premier cas, les deux individus se croisent et s'échangent des portions d'ADN en un seul point.

Dans le deuxième cas, il y a deux points de croisement. Pour les algorithmes génétiques.

Simple enjambement



Double enjambement

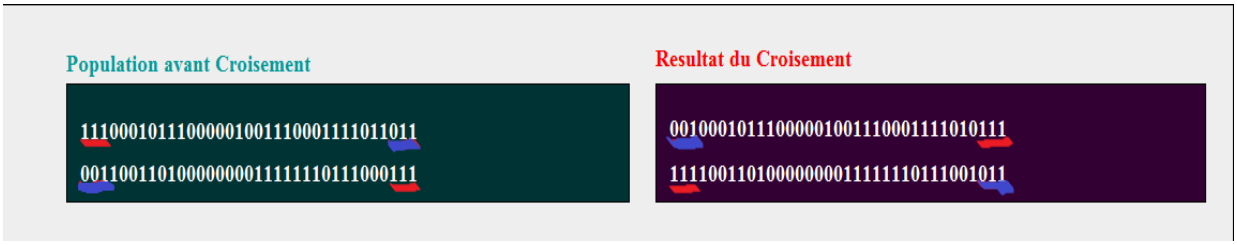


Figure 3.4 Croisement

• **L'opérateur de mutation :** Il modifie faiblement un individu de manière aléatoire. Cet opérateur retournera une population mutée à une population à muter.

Figure 3.5 Mutation

• **Remplacement**

Cette étape constitue la population de la génération suivante à partir des parents et des enfants de la génération courante.

Une fraction de la population est remplacée par sa descendance à chaque génération.

L'écart entre les générations indique la proportionnel de parents remplacés par des enfants.

Variante 1

Dans cette version la probabilité de mutation (Pm) n'a aucune importance. La probabilité de croisement (Pc) définie le pourcentage d'individus qui seront issus d'un croisement, le reste de la population sera issue de mutations.

Variante 2

Cette version quant à elle prend en compte la totalité des paramètres, comme dans l'algorithme 1 la probabilité de croisement définie ici aussi le pourcentage d'individus qui seront issus d'un croisement, mais la nuance est que la probabilité de mutation définie quant à elle le pourcentage d'individus issus de mutations, le reste de la population ne sera issu ni de mutation ni de croisement mais sera pioché au hasard dans le vivier de sélection.

7. Le déroulement de l'application :

On a deux partie dans l'application qui sont :

7.1. Partie 1 : Mono objectif

- **Générer le nombre des taches :**

Pour générer le nombre des taches j'utilise la fonction prédéfinie **Math.random** générer un nombre aléatoire entre 0 et 7.

- **Générer les critères de la tâche :**

  Chaque tache a cinq critères qui sont :

- Le numéro de la tache ;
- Le nom de la tache ;
- Le temps d'arrivé : correspond au temps qu'un opérateur va pouvoir mettre pour l'exécuter ;
- Le temps d'exécution : correspond à la date laquelle la tâche devra être terminée ;
- Priorité : correspond à l'importance de cette tâche ;

Et aussi un code binaire, Chaque tâche était codée sur cinq bits, pour produire ce code binaire j'utilise trois fonctions qui sont :

- La fonction prédéfinie **Math.random** générer un nombre aléatoire entre 0 et 1 ;
- La fonction modulo (%) qui donne le reste de la division par 2 ;
- Enfin la concaténation des résultats obtenus ;

Après il faut faire une vérification du code à ce qu'il existe déjà ou non, si oui le code va supprimer et les étapes doit refaire, sinon c'est le code de la tâche.

- **Choisi un critère :**

  Il faut choisi un critère parmi les trois critères (temps d'arrivé, temps d'exécution et la priorité).

- **Imaginer la solution :**

L'lorsque on appuyons sur Le bouton imaginer la solution les taches va tries selon le critère choisi,

- Si le critère choisi est le temps d'arrivé : l'ordonnancement est ascendant.
- Si le critère choisi est le temps d'exécution ou bien la priorité : l'ordonnancement est descendant.

- **Le fitness :**

  C'est le temps d'attente moyen. Ce calcul comme suit :

  (∑ (temps d'arrivé + temps d'exécution)) / le nombre des taches.

- **Le bouton importer vers la population :**

Pour enregistrer l'individu dans la base de donnée avec ces critères.

- **L'opérateur de mutation :**

- La fenêtre nombre des taches i : l'individu est codé par un nombre de taches i.
- Indice de démarrage mutation : pour préciser le bit pour commencer la mutation.
- Nombre de bit de mutation : c'est le nombre de bit (bloc) pour effectuer l'opérateur de mutation.
- Valider : le 0 de l'individu devient 1 et le 1 devient 0.

La résultat d la mutation dans la fenêtre l'individu après.

- **L'opérateur de croisement :**

On a deux cas :

- **Simple enjambement :**

Le simple enjambement consiste à fusionner les particularités de deux individus à partir d'un pivot. Pour d'obtenir deux enfants.

Pivot : nombre de bit à partir du premier bit de l'entête c-à-d : a partir de la première bit de l'individu le 0 devient 1 et le 1 devient 0.

- **Double enjambement :**

Le double enjambement repose le même principe sauf qu'il y a deux pivots

Pivot 1 : pour l'entête.

Pivot 2 : pour la queue.

La fenêtre population avant croisement : représente les deux individus parents (les individus avant la croisement).

La fenêtre résultat de croisement : représente les deux individus fils (les individus après la croisement).

- **Meilleur individu :**

On a déterminé le meilleur individu à partir de la fitness c-à-d : l'individu qui a le moins de fitness parmi la population c'est le meilleur individu.

- D'abord choisi le nombre des taches (pour choisi les individus) c-à-d la taille des individus entre 1 et 8.

- Valider : dans la fenêtre individu trouver les individus de la taille i (i le nombre des taches) sont affiché suivi par le critère qu'on a choisi au début suivi par la taille d'individu suivi par le fitness.

  Les individus trouvés sont ordonnés par leurs fitness ascendante.

- Dans la fenêtre meilleur individu elle affiche le premier individu qui a le plus petit fitness.
- Le bouton tracer graphe global : pour tracer des graphes pour les individus trouvés de la taille i par rapport à leurs valeurs de fitness.

- Actualiser : pour redémarrer les étapes.

7.2. Partie 2 : mono-objectif

Dans cette partie on a deux types d'algorithmes génétiques :

Algorithme génétique 1 : Dans cette version la probabilité de mutation (Pm) n'a aucune importance.
La probabilité de croisement (Pc) définie le pourcentage d'individus qui seront issus d'un croisement,
le reste de la population sera issue de mutations.

Algorithme génétique 2 : Cette version quand à elle prend en compte la totalité des paramètres, comme dans l'algorithme 1 la probabilité de croisement définie ici aussi le pourcentage d'individus qui seront issus d'un croisement, mais la nuance est que la probabilité de mutation définie quand à elle le pourcentage d'individus issus de mutations, le reste de la population ne sera issu ni de mutation ni de croisement mais sera pioché au hasard dans le vivier de sélection.

Les étapes :

- Choisi un algorithme parmi les deux algorithmes génétiques ;
- Fixé le nombre des taches ;
- Générer la population de la base de données ;
- Clin d'œil sur la base : pour affiche la table qui contient les taches, chaque tache suivie par ces critères (temps d'exécution, date limite d'exécution et la priorité) ;
- Importer tache from BDD : dans la fenêtre de visualisation les taches sont affiché ;
- On a trois fenêtre : Durée d'exécution, date limite d'exécution et importance, pour chaque importation de tache dans ces trois fenêtres nos affichons des valeurs associées à la tâche ;
- Fixé le nombre de génération ;
- Appuyer sur alg1 pour effectuer l'opération d'ordonnancement ;

Dans la fenêtre de visualisation nos affiche :

➔ Génération : le nombre de génération qu'on a appliqué sur cette population ;
➔ Moyen des priorités : (∑(priorité)) / nombre des taches ;
➔ Meilleur individu ;
➔ Plus mauvais individu ;

- On distinguer entre le meilleur individu et le plus mauvais individu à partir du moyen des priorités.

8. Implémentation

8.1.Matériel utilisé

Nous avons utilisé un ordinateur Intel (R) core (TM)i3-4030U CPU @1.90 GHz avec une RAM de 4 Go et un système Windows 8.1 édition intégral.

8.2. Langage utilisé

Java est un langage de programmation orienté objet et un environnement d'exécution, développé par Sun Microsystems.

Il fut présenté officiellement en 1995. Le Java était à la base un langage pour Internet, pour pouvoir rendre plus dynamiques les pages (tout comme le JavaScript aujourd'hui). Mais le Java a beaucoup évolué et est devenu un langage de programmation très puissant permettant de presque tout faire, je dis bien presque car nous verrons pourquoi il ne permet pas de tout faire. Java est aujourd'hui officiellement supportée par Sun, mais certaines entreprises comme IBM font beaucoup pour Java

8.3. Langage Orienté objet

Java est complètement orienté objet. Java vous permet et vous pousse même à développer vos applications d'une façon orientée objet et vous permet d'avoir une application bien structurée, modulable, maintenable beaucoup plus facilement et efficace. Cela augmente une fois de plus votre productivité.

8.4.Avantages de Java

• Portabilité excellente

• Langage puissant

• Langage orienté objet

• Langage de haut niveau

• JDK très riche

• Nombreuses librairies tierces

• Très grande productivité

• Applications plus sûres et stables

• Nombreuses implémentations, JVM et compilateurs, libres ou non

• IDE de très bonne qualité et libres : Eclipse et Netbeans par exemple

• Supporté par de nombreuses entreprises telles que Sun ou encore IBM et des projets comme Apache.

9. Outils utilisé : Netbeans

On a utilisé Netbeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, Netbeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). Compilé en Java, Netbeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

10. Présentation du modèle proposé

Le programme peut être exécuté en tant qu'application indépendante« java interface ». Dans notre application on va présenter les interfaces graphiques associé aux différentes fonctionnalités du système.

10.1. Bibliothèque java utilisée

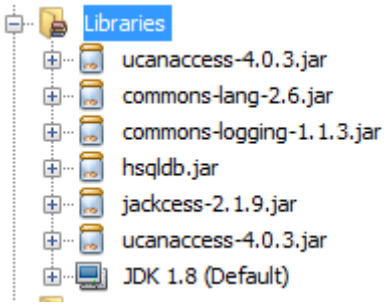
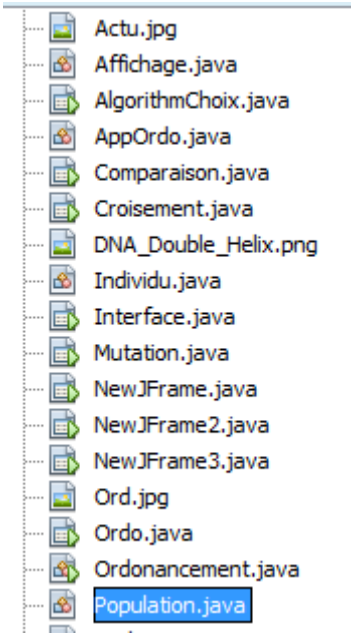


Figure 3.6 : Bibliothèque java utilisée.

10.2. Classes utilisées



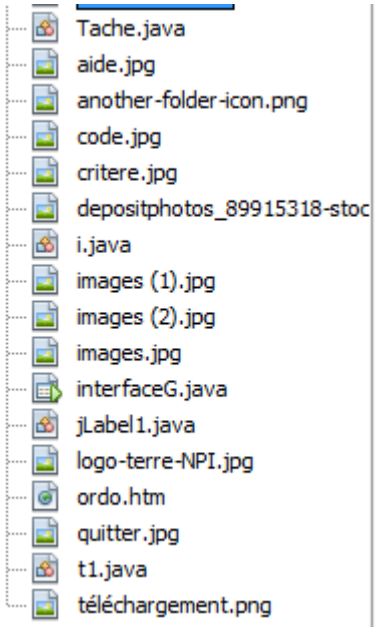


Figure 3.7 : Classes utilisées.

11. Lancement de l'application

11.1. Fenêtre d'accès au système

Après avoir lancé l'application, une fenêtre d'accueil apparaitra qui demande à l'utilisateur de choisir entre « Mono objectif » et « multi objectif » en cliquant sur l'un des boutons

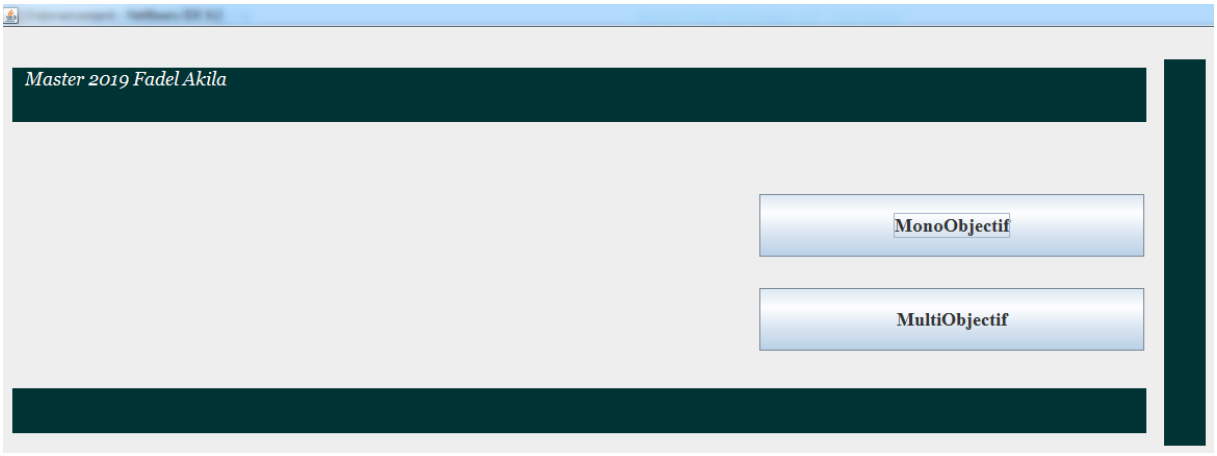


Figure 3.8: Interface d'accueil

11.2. Mono objectif

Afin d'accéder au menu principal de la partie mono objectif de l'application

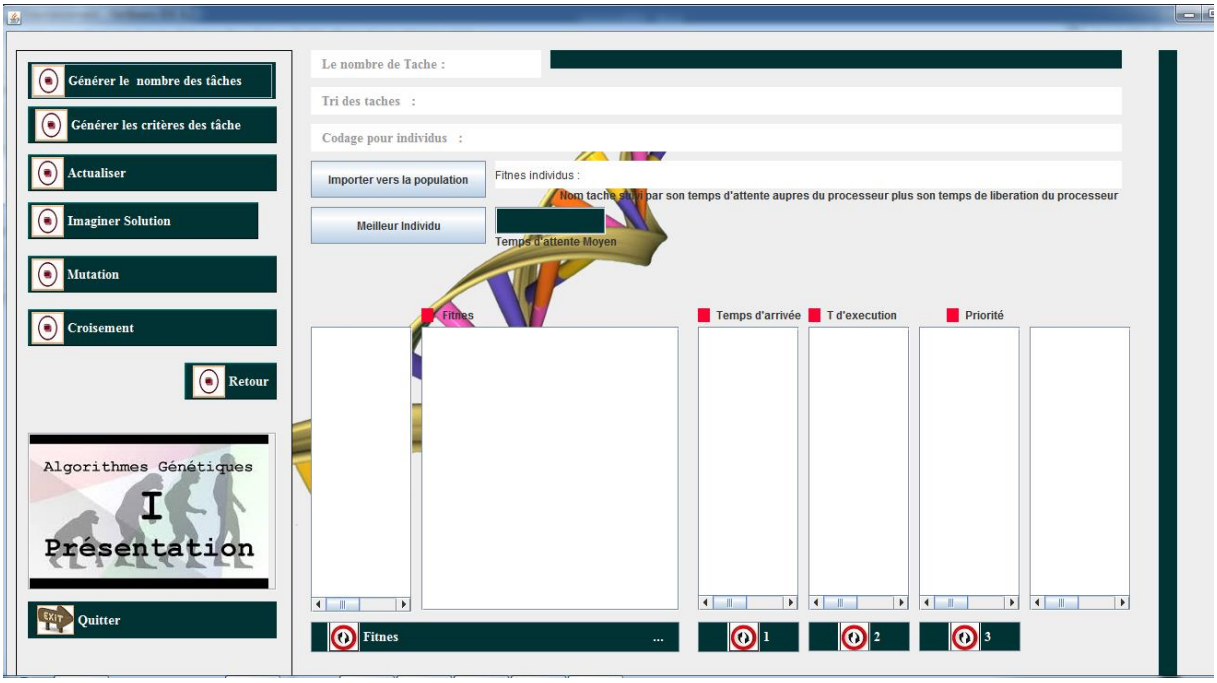


Figure 3.9: menu de la partie mono objective.

Après, on génère un nombre de taches et les leurs critère à partir de la base de donnée puis on affecte un code binaire différent pour chacune de ces taches afin de codifier l'individu.

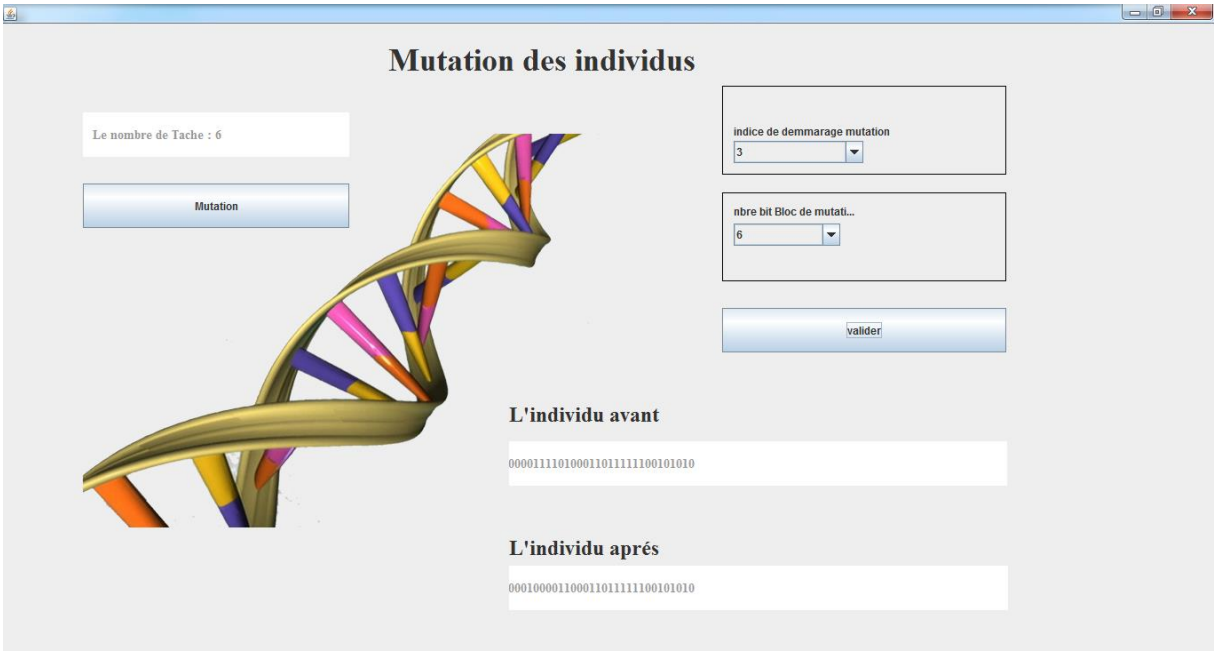


Figure 3.10: Interface de mutation.

Ensuite, on applique l'opérateur Mutation sur l'individu obtenu en choisissant un indice de démarrage et un nombre de bloc de mutation. Cette opération sert à changer le 0 à 1 et le 1 à 0.

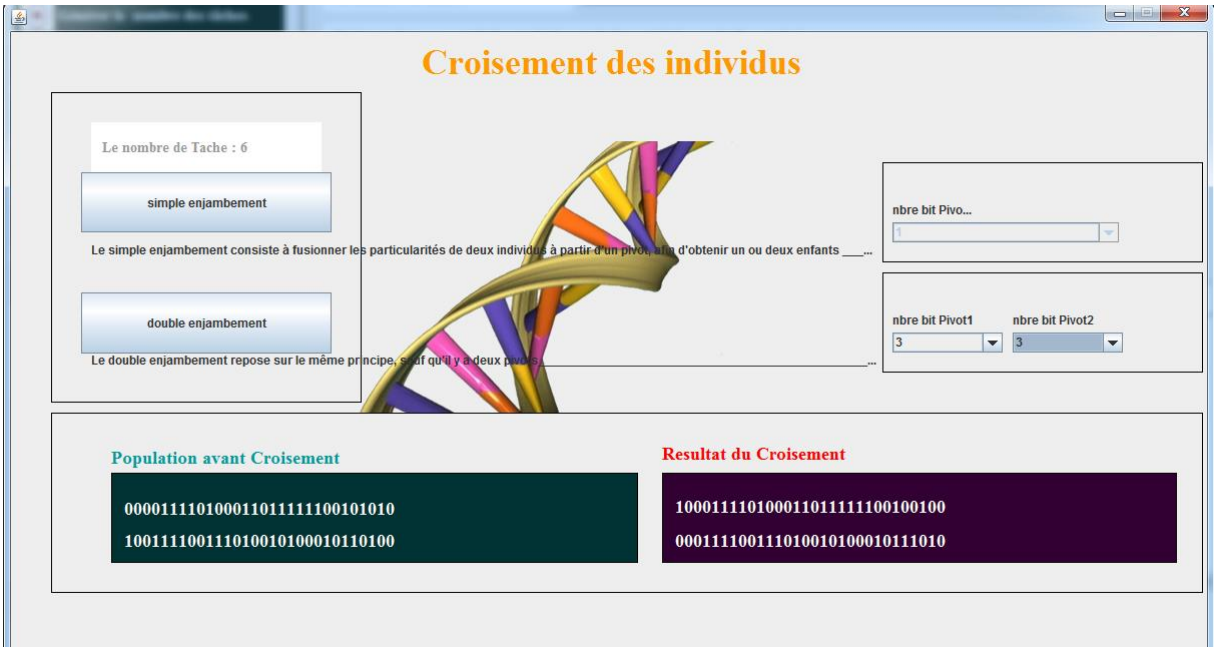


Figure 3.11 Interface de croisement

Pour trouver le meilleur individu en clique sur le bouton Meilleur individu, ensuite fixé le nombre des taches puis valider.

Dans la fenêtre les individus de ce nombre des taches s'affichent ordonner par le fitness et en bas affiche le meilleur individu qui a minimum fitness entre les autres individus.

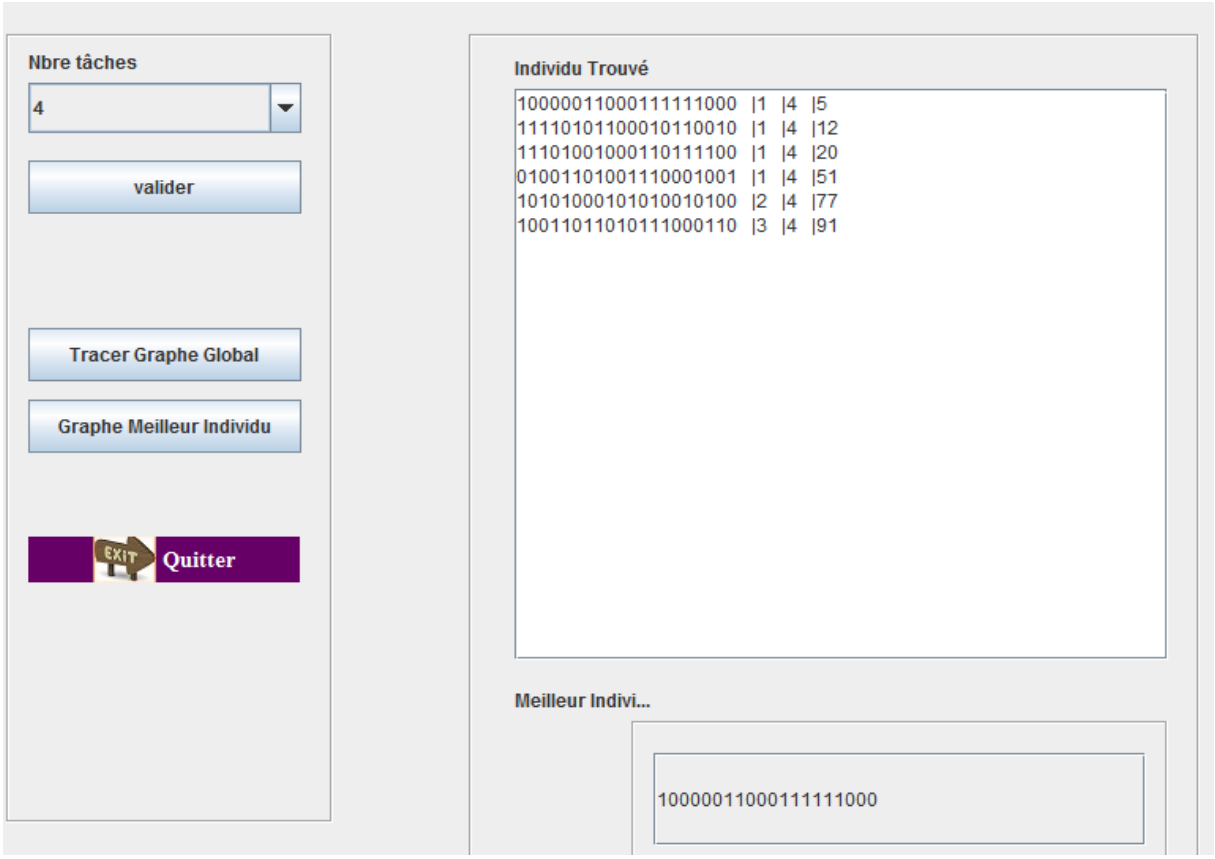


Figure 3.12 Interface meilleur individu

Après en clique sur le bouton trace graphe global pour désigner les graphes des individus, l'application s'affiche 3 versions des graphes

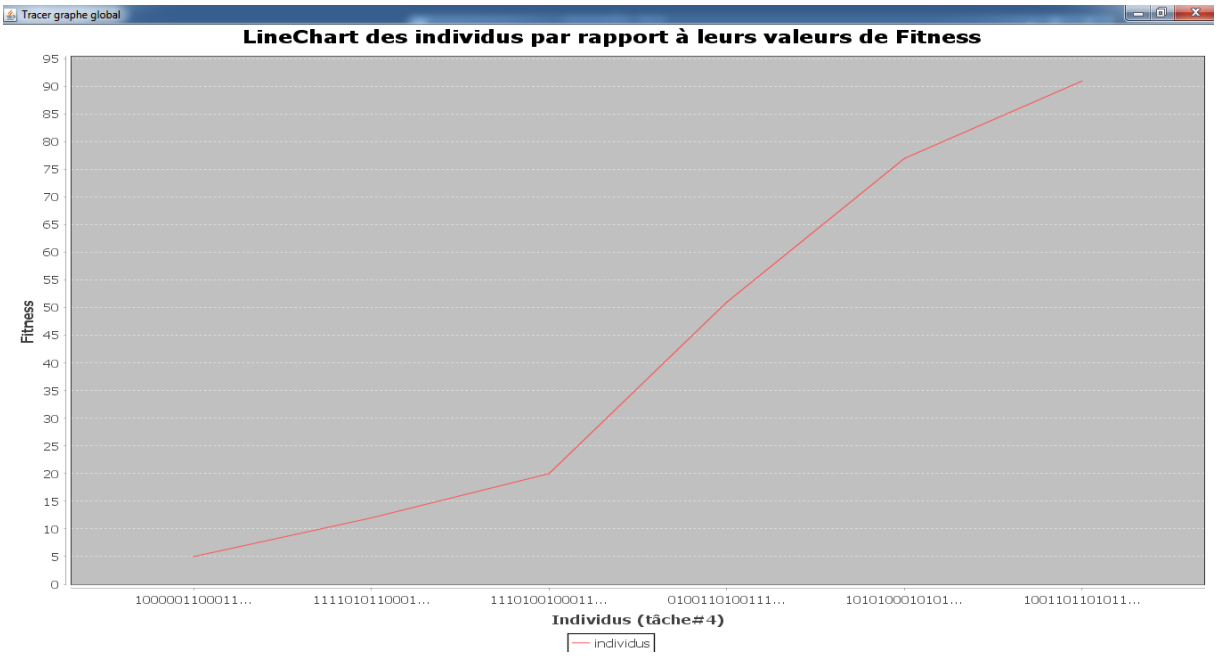


Figure 3.13 Interface lineChart des individus

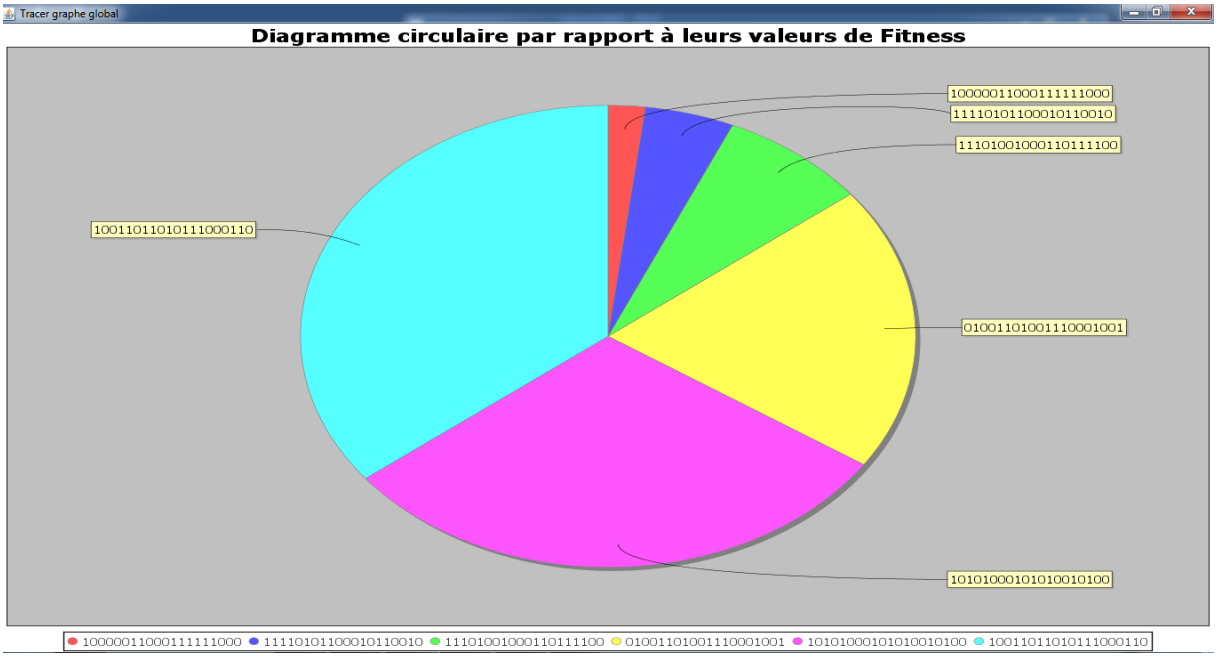


Figure 3.14 Interface diagramme circulaire

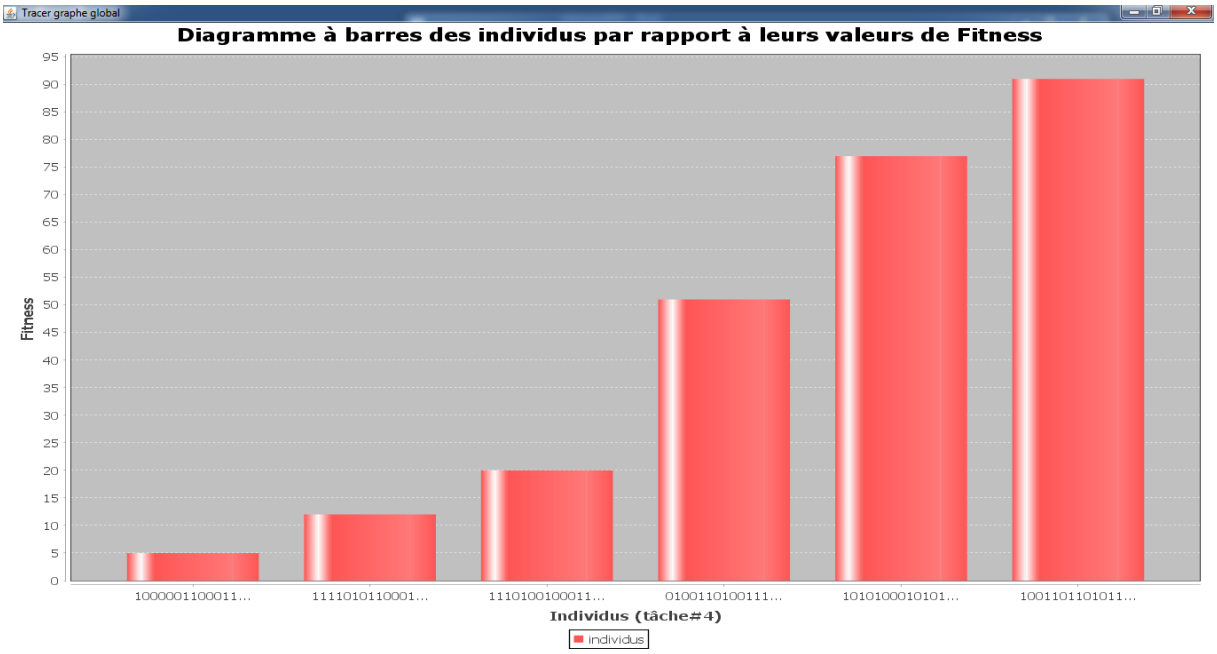


Figure 3.15 Interface diagramme à barres

11.3. Multi objectifs

Afin d'accéder au menu principal de la partie multi objectif de l'application, en a un choix entre 2 algorithmes génétiques 1 et 2

Figure 3.16 : Interface de choix

Après le choix on aura cette fenêtre de visualisation qui sert à manipuler les différentes fonctionnalités du système qui commence par la génération de la population

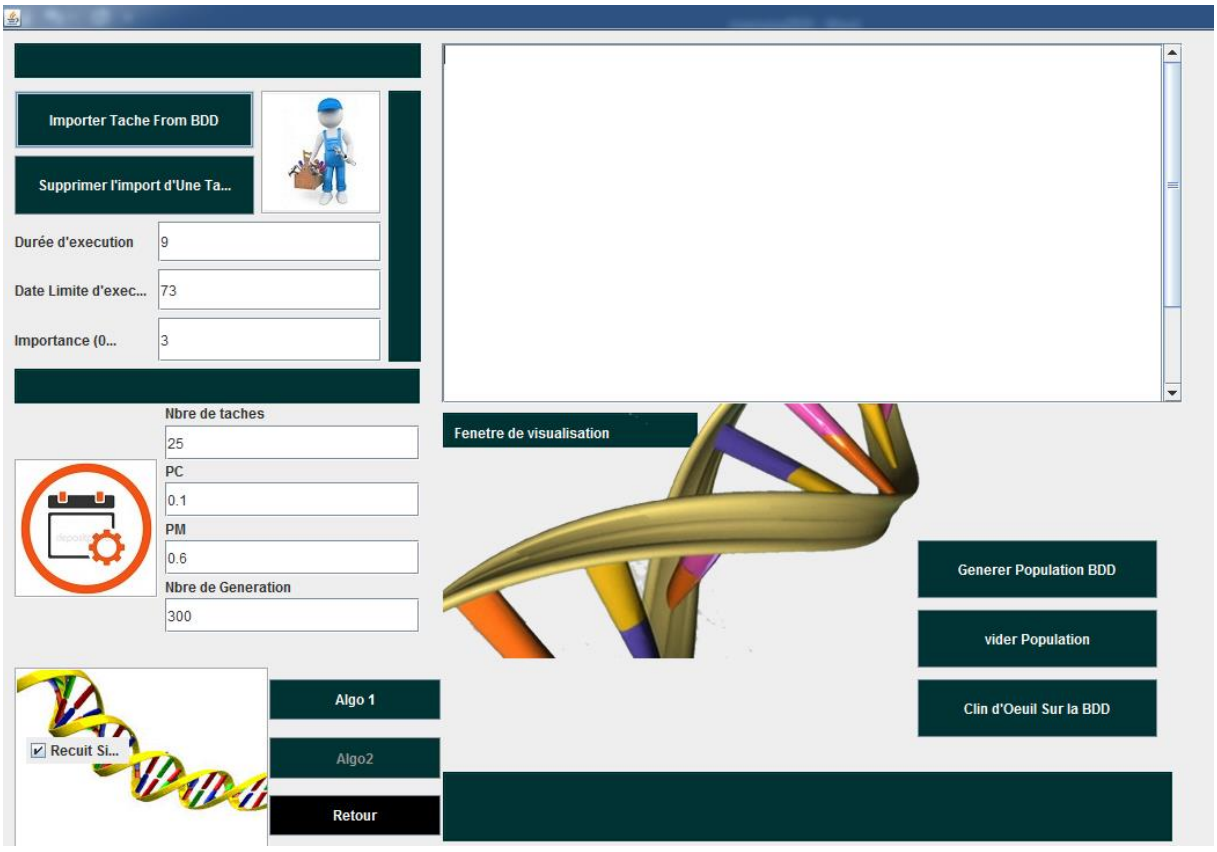


Figure 3.17: Interface multi objectif

Résultats obtenues :

Figure 3.18: Fenêtre des résultats

Les résultats obtenus présente la solution optimale représenter en : Meilleur individu et formule aussi le plus mauvais individu calculer à partir de la fitness.

12. Conclusion

Dans ce chapitre, nous avons essayé d'exposer l'ensemble des idées caractérisant le système proposé en se concentrant sur la conception et l'implémentation de l'application.

Nous avons présenté les interfaces et les captures d'écrans de notre application. Les résultats obtenus en appliquant l'algorithme génétique sur le problème d'ordonnancement de tache.

Conclusion générale

Conclusion générale

Les problèmes d'ordonnancement se retrouvent souvent dans la vie courante a travers de très nombreuses applications, et suscitent donc un intérêt croissant.

Leur but est de déterminer la meilleure manière d'effectuer un ordre sur un ensemble de solutions aux membres d'une population afin d'optimiser un objectif donné. Dans ce travail, nous avons proposé un cadre pour l'optimisation multicritère pour un problème d'ordonnancement des taches utilisant l'algorithme génétique. Nous avons appliqué le principe de recherche d'une meilleure solution.

L'application de l'algorithme génétique a permis de construire une solution de bonne qualité avec un temps raisonnable pour le problème d'ordonnancement avec un nombre élevé de taches. Les algorithmes génétiques semble être une technologie prometteuse et s'adapte mieux aux applications complexes.

En perspective nous proposons d'hybrider les méthodes heuristiques telles que le recuit simulé pour l'amélioration de la solution retrouver par l'algorithme génétique et aussi comparer les résultats obtenus avec d'autre méthode d'optimisation.

Ce mémoire présente un cadre de travail conceptuel et architectural pour la mise en place d'une bonne optimisation multi objectif.

Bibliographie

Bibliographie

[Holland,1962] J. H. Holland, “Outline for a logical theory of adaptive systems,” Journal of the Association for Computing Machinery, vol. 9, no. 3, pp. 297–314, 1962.

[DeJong,1975] K. A. De Jong, “An Analysis of the Behaviour of a Class of Genetic Adaptive Systems”, Ph.D. Thesis, University of Michigan, 1975.

[Goldberg, 1988] D. E. Goldberg AND J. H. Holland, "Genetic Algorithms and Machine Learning",Machine Learning, vol. 3, pp.95-99, 1988.

[Goldberg, 1989] D.E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley Longman Publishing, 1989.

[Groupe gotha,1993] GOThA Groupe d'Ordonnancement Théorique et Appliqué, « Les problemes d'ordonnacement »,1993

[Blake,Black,Carlson,Davies,Wang and Weiss,1998] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service, 1998.

[Yang, Chen & Yeo 1999] Yang G., Chen I-M., and Yeo S-H., Design consideration and kinematic modeling for modular reconfigurable parallel robots, In 10th World Congress on the Theory of Machines and Mechanisms, pages 1079-1084, Oulu, 1999.

[Oliver 2000] Oliver C., Machines-outils rapides à structure parallèle, Méthodologie de conception applications et nouveaux concepts. Thèse de doctorat, Université de Montpellier II, 2000.

[Grolleau et Choquet-Geniet,2000] A Choquet-Geniet, E Grolleau, F Cottet - TSI. Technique et science …, 2000 – Lavoisier

[Napoleon, Carvajal,2001] Octavio Napoleon Medina Carvajal. Etude des algorithmes d’attribution de priorités dans un Internet `a différenciation de services. Thèse de doctorat, Université de Rennes I, March 2001.

[Toumi,2002] Leila Toumi. Algorithmes et mécanismes pour la qualité de service dans des réseaux hétérogènes. Thèse de doctorat, Institut National Polytechnique de Grenoble, Décembre 2002.

[Olivier et yves,2003]L Deslandes, J **Olivier**, N Peeters… - Proceedings of the …, **2003** - National Acad Sciences.

[Lochin,2004] Emmanuel Lochin. Qualité de Service dans l'Internet. Garantie de d´ebit TCP dans la classe AF. Thèse de doctorat, Université de Paris VI, Décembre 2004.

[Paschos,2005] V. Th. Paschos, Optimisation combinatoire : concepts fondamentaux , Hermès cience publication, pp. 17-60, Lavoisier, 2005.

[**Jérôme,2007**] Jérôme Clet-Ortega. Ordonnancement et qualité de service pour réseaux rapides, octobre 2007.

[EL Hilali et al 2009] « Initiation à la recherche opérationnelle », 2009.

[Naziha, 2010] Melle Naziha Ali Saoucha "paramétrage des algorithmes génétiques pour l'optimisation de la QOS dans les réseaux radios cognitifs,"

[Xin 2010] Xin-SheYang, Engineering Optimization: An Introduction with Metaheuristic Applications - Published by John Wiley & Sons, Inc., Hoboken, New Jersey,ISBN 978-0-470-58246-6, 2010.

[Gherboudj 2013] Méthodes de résolution de problèmes difficiles acad émiq ue s :Amira Gherboudj, 2013

[Smairi 2013] Optimisation par essaim particulaire : adaptation de tribes à l'optimisation multiobjectif (Doctoral dissertation, Université Paris-Est),2013

[Kaidi & Chelouti, 2016] S **Chelouti**, K **Kaidi** - Mémoire de Master, Université M. Bougara Boumerdes, **2016**