MINISTERE DE LA COMMUNICATION,
DE L'ECONOMIE NUMERIQUE ET DE LA POSTE

Ecole Supérieure Africaine des Technologies de l'Information et de la Communication

ESATIC
ECOLE SUPÉRIEURE AFRICAINE DES TIC

REPUBLIQUE DE CÔTE D'IVOIRE

UNION – DISCIPLINE -TRAVAIL

MULTI-OBJECTIVE GENETIC ALGORITHM (MOGA)

ENSEIGNANT : DR DIABY MOUSTAPHA

ETUDIANT : DIAKITE SOUMAILA

SOMMAIRE

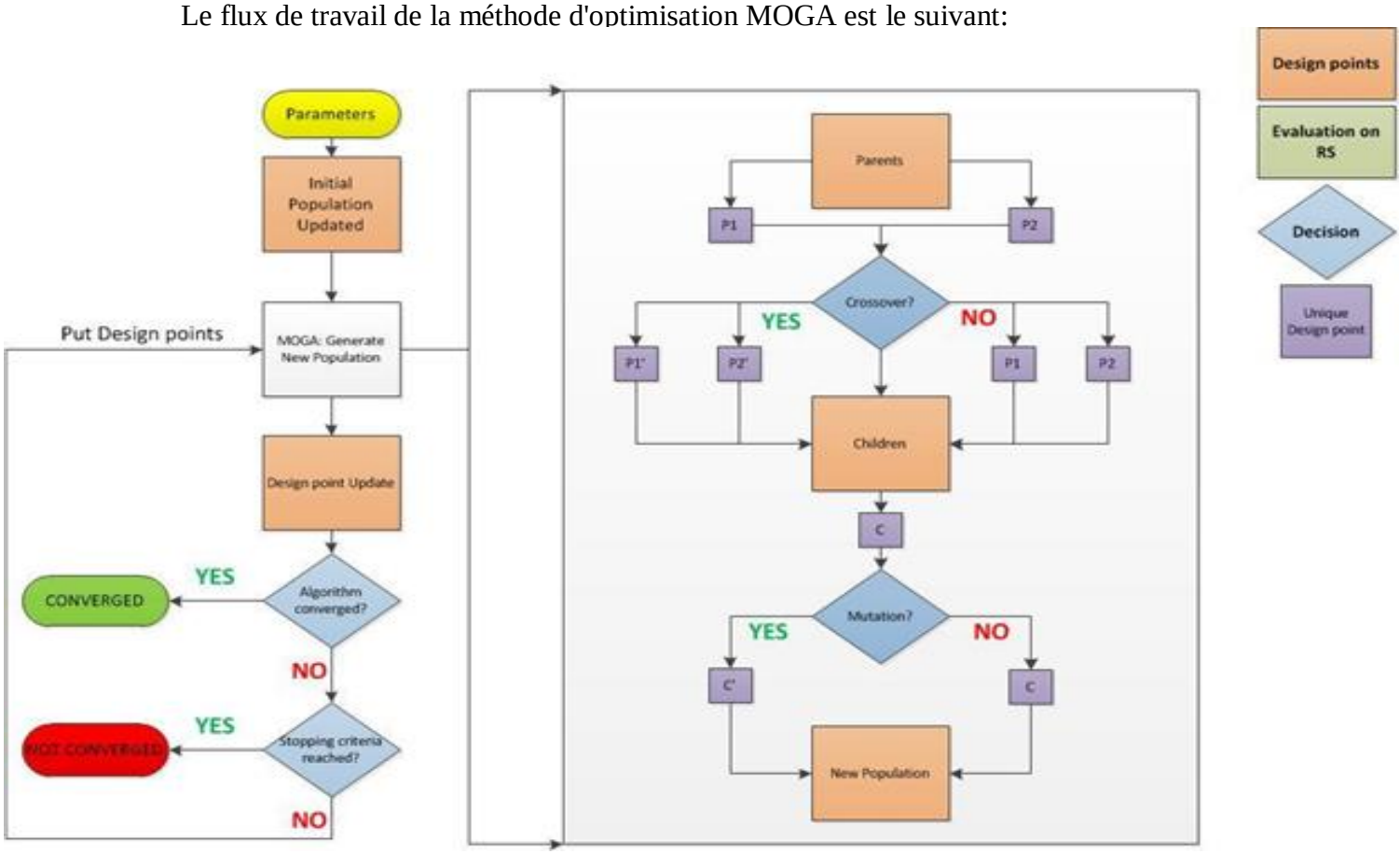
INTRODUCTION

Le MOGA utilisé est une variante hybride du populaire NSGA-II (algorithme génétique trié non dominé) basée sur des concepts d'élitisme contrôlé. Il supporte tous les types de paramètres d'entrée. Le système de classement de Pareto est défini à l'aide d'une méthode de tri rapide, non dominée, d'un ordre de grandeur plus rapide que les méthodes de classement de Pareto traditionnelles.

Les premières solutions frontales Pareto sont archivées dans un jeu d'échantillons séparé en interne et sont distinctes du jeu d'échantillons en évolution. Cela garantit une perturbation minimale des motifs de front de Pareto déjà disponibles lors des itérations précédentes. Vous pouvez contrôler la pression de sélection (et, par conséquent, l'élitisme du processus) pour éviter une convergence prématurée en modifiant la propriété Pourcentage de Pareto maximal autorisé.

I. MOGA flux de travail

Le flux de travail de la méthode d'optimisation MOGA est le suivant:



II. Etapes de MOGA

1. *Première population de MOGA*

La population initiale est utilisée pour exécuter l'algorithme MOGA.

2. *Génère une nouvelle population*

MOGA est géré et génère une nouvelle population par **croisement** et **mutation**. Après la première itération, chaque population est exécutée lorsqu'elle atteint le nombre d'échantillons défini par la propriété Nombre d'échantillons par itération.

3. *Point de conception*

Les points de conception dans la nouvelle population sont mis à jour.

4. *Validation de convergence*

L'optimisation est validée pour la convergence.

- Oui: optimisation convergée

La MOGA converge lorsque le pourcentage maximal autorisé de Pareto ou le pourcentage de stabilité de la convergence est atteint.

- Non: optimisation non convergée

Si l'optimisation n'est pas convergée, le processus passe à l'étape suivante.

5. *Arrêt de la validation des critères*

Si l'optimisation n'a pas convergé, elle est validée pour le respect des critères d'arrêt.

- Oui: Critères d'arrêt satisfaits

Lorsque le critère Nombre maximal d'itérations est rempli, le processus est arrêté sans atteindre la convergence.

- Non: Critères d'arrêt non remplis

Si les critères d'arrêt ne sont pas remplis, la MOGA est exécutée à nouveau pour générer une nouvelle population (retour à l'étape 2).

6. Conclusion

Les étapes 2 à 5 sont répétées dans l'ordre jusqu'à ce que l'optimisation ait convergé ou que les critères d'arrêt aient été satisfaits. Lorsque l'une ou l'autre de ces choses se produit, l'optimisation est terminée.

III. La génération d'une nouvelle population

Le processus utilisé par MOGA pour générer une nouvelle population comprend deux étapes principales: **Cross-over** et **Mutation**.

1) Cross-over

Cross-over combine deux partenaires (chromosomes) pour produire un nouveau chromosome (progéniture). L'idée sous-jacente au croisement est que le nouveau chromosome peut être meilleur que les deux parents s'il prend les meilleures caractéristiques de chacun des parents. Le croisement se produit au cours de l'évolution selon une probabilité de croisement définissable par l'utilisateur.

- **Cross-over pour les paramètres continus**

Un opérateur croisé qui combine linéairement deux vecteurs de chromosome parents pour produire deux nouveaux descendants selon les équations suivantes:

progéniture1= a*Parent1 + (1-a)*Parent2

progéniture2= (1-a)*Parent1 + a*Parent2

Considérez les deux parents suivants (chacun composé de quatre gènes flottants) qui ont été sélectionnés pour le croisement:

Parent1: (0.3)*(1.4)*(0.2)*(7.4)

Parent2: (0.5)*(4.5)*(0.1)*(5.6)

Si a = 0,7, les deux produits suivants seraient produits:

progéniture1= (0.36)*(2.33)*(0.17)*(6.86)

progéniture2= (0.402)*(2.981)*(0.149)*(6.842)

- **Cross-over pour les paramètres de valeurs discrètes / manufacturables**

Chaque paramètre de valeurs discrètes ou pouvant être fabriquées est représenté par une chaîne binaire correspondant au nombre de niveaux. Par exemple, un paramètre avec deux valeurs (niveaux) est codé sur un bit, un paramètre avec sept valeurs est codé sur trois bits et une chaîne n-bits représentera un paramètre avec 2(n-1) valeurs.

La concaténation de ces chaînes forme le chromosome, qui se croise avec un autre chromosome.

Trois types de croisement sont disponibles:

- ✓ **Un point**

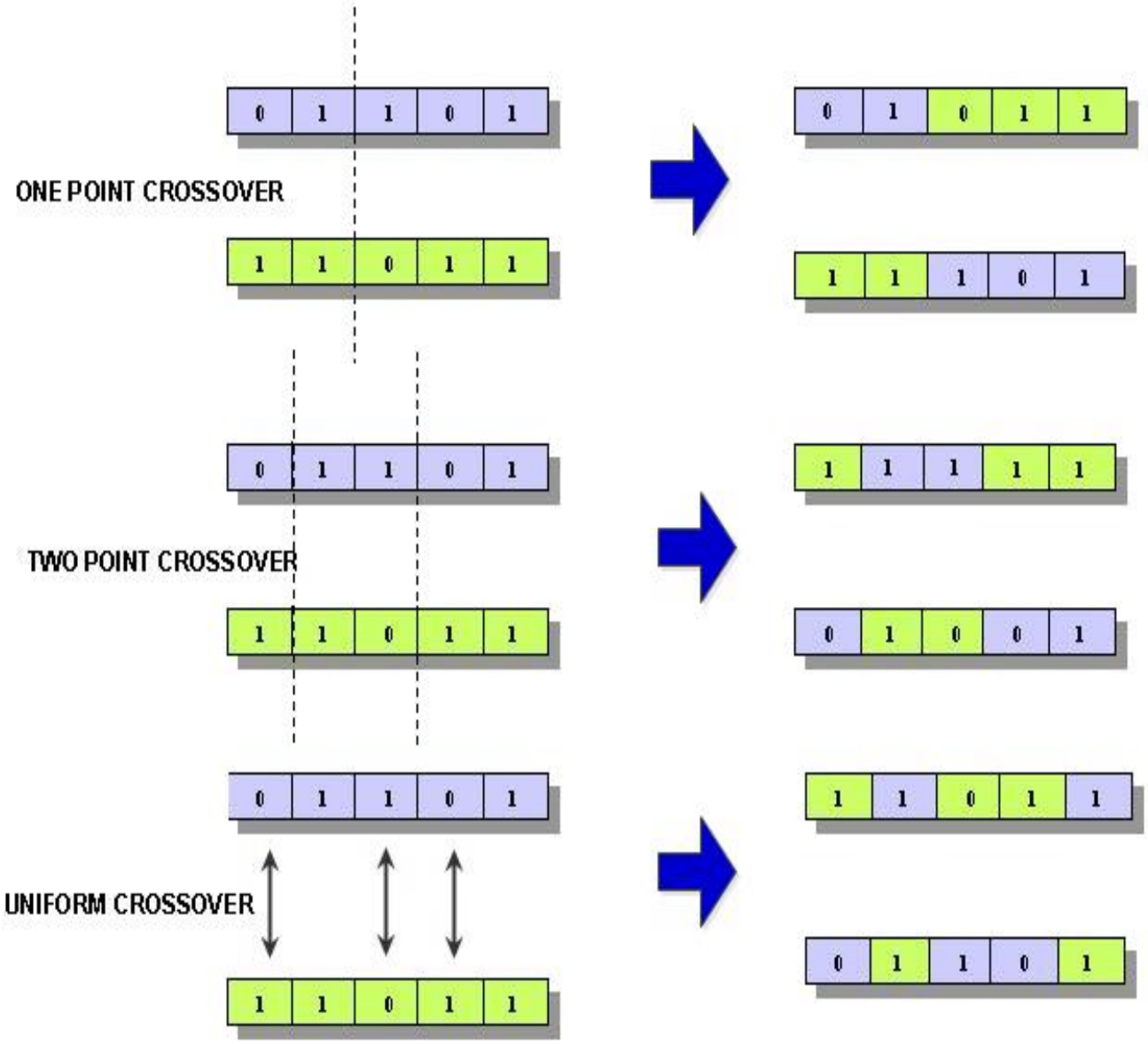
Un opérateur de croisement en un point qui sélectionne de manière aléatoire un point de croisement dans un chromosome, puis échange les deux chromosomes parents en ce point pour produire deux nouveaux descendants.

- ✓ **Deux points**

Un opérateur de croisement à deux points sélectionne de manière aléatoire deux points de croisement au sein d'un chromosome, puis échange les deux chromosomes parents entre ces points pour produire deux nouveaux descendants.

- ✓ **Uniforme**

Un opérateur de croisement uniforme décide (avec une certaine probabilité, connue sous le nom de "rapport de mélange") quel parent contribuera à chacune des valeurs du gène dans les chromosomes de la progéniture. Cela permet aux chromosomes parents d'être mélangés au niveau du gène plutôt qu'au niveau du segment (comme avec un croisement à un et deux points). Pour certains problèmes, cette flexibilité supplémentaire l'emporte sur l'inconvénient de la destruction des blocs de construction.


ONE POINT CROSSOVER
TWO POINT CROSSOVER
UNIFORM CROSSOVER

2) Mutation

La mutation modifie une ou plusieurs valeurs de gêne dans un chromosome à partir de son état initial. Cela peut entraîner des valeurs de gènes entièrement nouvelles ajoutées au pool de gènes. Avec ces nouvelles valeurs de gènes, l'algorithme génétique pourrait permettre de trouver une meilleure solution qu'auparavant. La mutation est une partie importante de la recherche génétique, car elle aide à empêcher la population de stagner, quel que soit le résultat optimal local. La mutation se produit au cours de l'évolution en fonction d'une probabilité de mutation définie par l'utilisateur.

- **Mutation pour paramètres continus**

Pour les paramètres continus, un opérateur de mutation polynomiale est appliqué pour mettre en œuvre la mutation.

Child = Parent + (UpperBound - LowerBound) * variation

Où Child est l'enfant, Parent est le parent et variation est une petite variation calculée à partir d'une distribution polynomiale.

- **Mutation pour les paramètres de valeurs discrètes / pouvant être fabriquées**

Pour les paramètres de valeurs discrètes ou manufacturables, un opérateur de mutation inverse simplement la valeur du gène choisi (0 va sur 1 et 1 va sur 0) avec une probabilité de 0,5. Cet opérateur de mutation ne peut être utilisé que pour les gènes binaires. La concaténation de ces chaînes forme le chromosome, qui se croise avec un autre chromosome.

IV. ANALYSE ET CONCEPTION DE L'APPLICATION

1. Technologies utlisées

Python

Ce langage de programmation interprété, multi-paradigme et multiplateforme est utiliser pour réaliser la statistique.

2. Outils

VISUAL STUDIO CODE

3. Architecture du logiciel

Notre Logiciel a été structuré de la manière modulaire suivante :

- Un dossier **File** qui contient le jeu de donnée
- Un dossier **packages** qui contient la classe permettant la réalisation du projet
- Un dossier **save** qui contient notre fichier de sauvegarde de l'évolution de la population
- Un dossier **out** qui contient la solution finale retenue
- Un dossier **view** visualiser la courbe d'évolution

classRingStar

On a les méthodes suivantes :

Contructiongraphe : Qui permet de lire le fichier contenu dans File et remplir le graphe.

GenerationCycle : Qui permet de générer un cycle.

InitialisationPopulation : Qui permet d'initialisation la population.

CoutAnneau : Qui permet de calculer le cout anneau du cycle.

CoutAffectation : Qui permet de calculer le cout d'affectation du cycle.

fonctionTri : Qui permet de trier notre population.

sauvegarde_generation : Qui permet de sauvegarder la population en cours d'évolution.

Mutation : Qui effectue la mutation.Cette fonction permettant de créer des individus mutés de manière aléatoire dans la nouvelle génération.

Croisement : Qui effectue le croisement

selectionIndividu : Qui permet de sélectionner les meilleurs individus de la population total pour les intégrer à la génération en cours.

ConfrontationCycle : Fonction permettant de confronter deux cycles aléatoirement et de sélectionner le meilleur pour la génération suivante.

ValidationConvergence : Fonction permettant de vérifier la convergence de l'algorithme.

CONCLUSION

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnistes. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable. Il existe des inconvénients et des avantages lorsqu'on utilise ces méthodes.

BIBLIOGRAPHIE

https://fr.m.wikipedia.org/wiki/Algorithme_génétique