UNIVERSITÉ D'ANGERS
UFR DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

RAPPORT

UE - Algorithme intelligent pour l'aide à la décision

Algorithme génétique appliqué au problème OneMax
Apprentissage par renforcement pour la sélection intelligente des opérateurs de mutation

---

Réalisé par :

Chafik AKMOUCHE

Master 2 Intelligence Décisionnelle

2021 - 2022

Table des matières

Résumé

xxxx

Mots clés : Algorithme évolutionnaire, Algorithme génétique, Problème OneMax, Problème du Bandit, PM, UCB.

1 Introduction

En l'absence d'une méthode de résolution exacte pour un problème, plusieurs méthodes de recherche de solutions approchées sont utilisées dont les algorithmes génétiques.

Dans la première partie de ce papier, nous allons parler brièvement des algorithmes génétiques et de leurs objectifs, ensuite nous allons appliquer cet algorithme à l'un des problèmes les plus connus dans ce domaine, à savoir le problème OneMax. Enfin, nous allons discuter les résultats obtenus en testant les différents opérateurs de mutation, croisement, etc.

Le nombre de paramètres liés à cet algorithme est très important (plusieurs opérateurs de mutation, croisement, sélection, remplacement...) et il n'y a pas d'opérateur meilleur que l'autre car efficacité de chaque opérateur dépend de plusieurs paramètres et c'est ce que nous allons voir dans les parties suivantes. Afin de résoudre ce problème et répondre à cette question : quel opérateur utiliser et quand l'utiliser ? Nous allons essayer de mettre en place d'un mécanisme permettant de choisir un opérateur o à l'instant t afin d'augmenter l'efficacité de l'algorithme génétique.

Dans la deuxième partie, nous allons donc discuter le fonctionnement de l'algorithme génétique implémenté dans la première partie, ensuite nous allons proposer des solutions qui permettent de choisir intelligemment l'opérateur à utiliser à un instant t de l'exécution de l'algorithme génétique. Enfin, nous allons discuter les nouveaux résultats obtenus et faire une comparaison avec les résultats obtenus dans la première partie.

1.1 Rappels sur les algorithmes génétiques

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 de résolution exacte.

- Les algorithmes génétiques se basent au départ sur une population de solutions candidates qui va évoluer de génération en génération jusqu'à la génération qui contient les meilleures solutions.

- Chaque individu comprend des propriétés et il peut être sujet à des transformations génétiques (mutation, croisement...).

- Chaque individu est évalué et cette valeur d'aptitude (fitness) est un critère pour sa survie d'une génération à une autre.

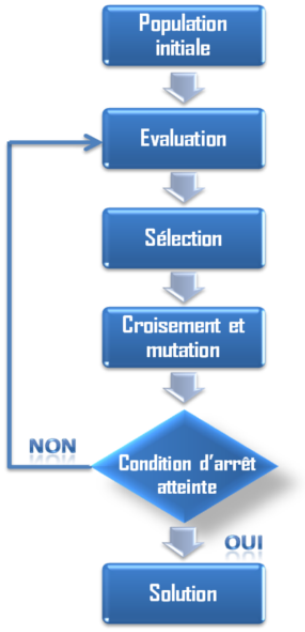

FIGURE 1 – Cycle de vie d'un algorithme génétique

1.2 Rappels sur le problème OneMax

Le problème OneMax est un problème simple qui consiste à maximiser le nombre de uns d'une séquence de bits.

Formellement, le problème OneMax peut être décrit comme la recherche d'une séquence (ou un vecteur) $\vec{x} = (x_1, x_2, ..., x_n)$, avec $x_i \in \{0, 1\}$, qui maximise l'équation suivante : $F(\vec{x}) = \sum_{i=1}^{n} x_i$

2 Application de l'algorithme génétique au problème OneMax

- Une population dans le cadre du problème OneMax est un ensemble fini d'individus ;
- Chaque individu est constitué de gênes (une séquence d'une taille finie de bits 0 ou 1) ;
- La population initiale est l'ensemble d'individus initiaux/première génération (n'ayant pas encore subit d'opérations de reproduction) ;
- La fitness d'un individu est la somme des bits ayant la valeur 1 ;
- La fitness d'une population est la somme des fitness des individus qui la composent.

À partir de la population initiale, l'algorithme génétique sélectionne un ou plusieurs individus et applique des opérations de reproduction (mutation, croisement...) sur ces derniers, ce qui va générer

de nouveaux enfants qui vont remplacer d'autres individus existants dans l'objectif de maximiser la fitness. L'algorithme tourne en boucle jusqu'à ce qu'une condition d'arrêt soit atteinte.

3 Implémentation de l'algorithme génétique

Paramètres de l'algorithme : Taille de la population, taille des individus, nombre max de générations, nombre d'exécutions.

Population initiale : Ensemble d'individus dont tous les gênes sont initialisés à 0.

Sélection : Représente le choix des individus les mieux adaptés pour les différentes opérations d'évolution.

Opérateurs de sélection implémentés : Sélection d'un individu au hasard, 2 individus au hasard, meilleur individu, 2 meilleurs individus et les 2 meilleurs individus sur 5.

Croisement : Consiste à mélanger les gênes des individus choisis afin de reproduire leurs particularités.

Opérateurs de croisement implémentés :

- Croisement uniforme : L'enfant issu de ce croisement prend des gênes des 2 parents.
- Croisement simple : Selon une probabilité, l'enfant prend les gênes du parent 1 ou du parent 2.
- Croisement mono-point : Création de 2 nouveaux enfants à partir de 2 parents sélectionnés, tel que chaque enfant prend une partie des gênes du parent 1 et l'autre partie du parent 2.

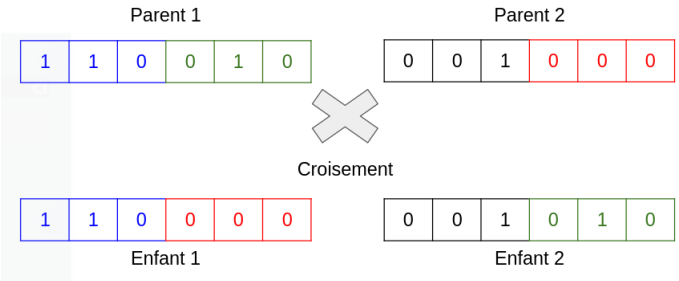

FIGURE 2 – Croisement mono-point

Mutation : Consiste à altérer un gène selon un facteur de mutation (probabilité). La mutation k-flip consiste à altérer k gênes d'un individu selon une probabilité de mutation.

Opérateurs de mutation implémentés : Mutation 1 flip, bit flip, 3 flip et 5 flip.

Remplacement : Représente le choix des individus à remplacer par les enfants issus des différentes opérations d'évolution.

Opérateurs implémentés : Remplacement du plus mauvais individu, des 2 plus mauvais individus et des 2 meilleurs individus

Fitness : $F(I) = \sum_{i=1}^{n} x_i$ tel que I : Individu ; x : gêne

Conditions d'arrêt : Nombre max de générations atteint, population parfaite atteinte (tous les gênes de tous les individus sont à 1).

4 Fonctionnement général de l'algorithme génétique

La figure 3 illustre le fonctionnement général de l'algorithme génétique implémenté ainsi que les différents opérateurs mis en place.

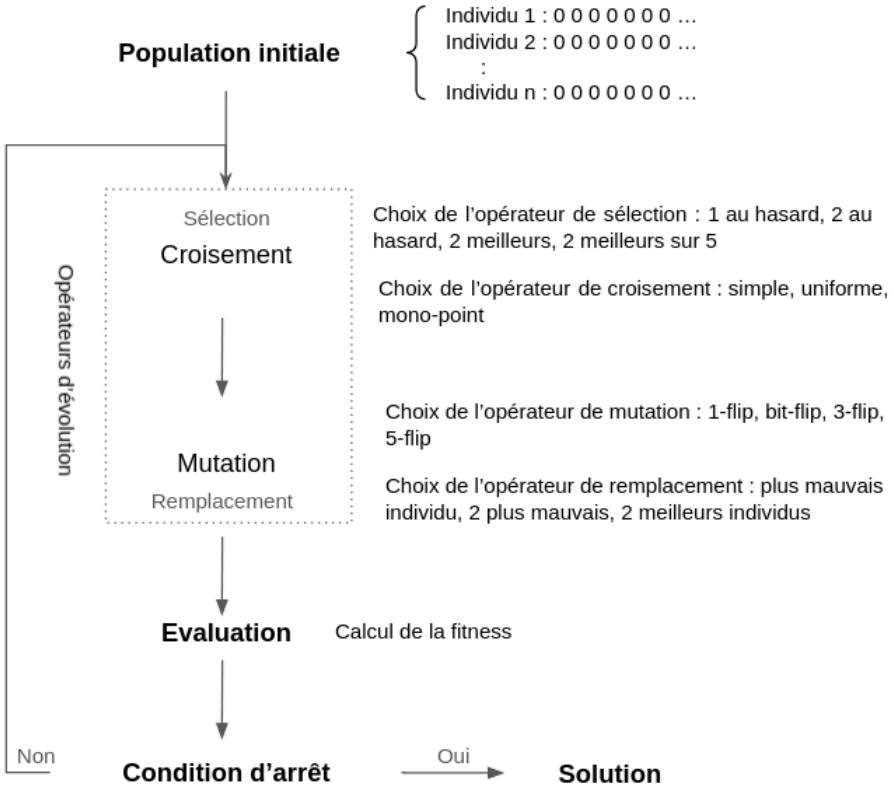

FIGURE 3 – Fonctionnement général de l'algorithme génétique implémenté

5 Résultats

Vu le nombre important de paramètres de l'algorithme (sélection, mutation, croisement, remplacement), nous ne pouvons pas étudier tous les résultats obtenus des différents paramètres. Dans cette étude, nous allons nous focaliser beaucoup plus sur les résultats liés à la mutation.

5.1 Fitness en fonction des opérateurs de mutation

La figure **??** montre l'évolution de la Fitness max obtenue sur 20 exécutions différentes en fonction des opérateurs de mutation (1-flip, bit-flip...).

- Paramètres de l'algorithme :
  - Taille de la population : 1000
  - Taille de l'individu : 1000
  - Max génération : 500
  - Nombre d'exécution : 10
  - Opérateur de sélection : Sélection des 2 meilleurs individus de la population.
  - Opérateur de croisement : Croisement mono-point.
  - Probabilité de mutation : 0.25.
  - Opérateur de remplacement : Remplacement des 2 plus mauvais individus de la population.

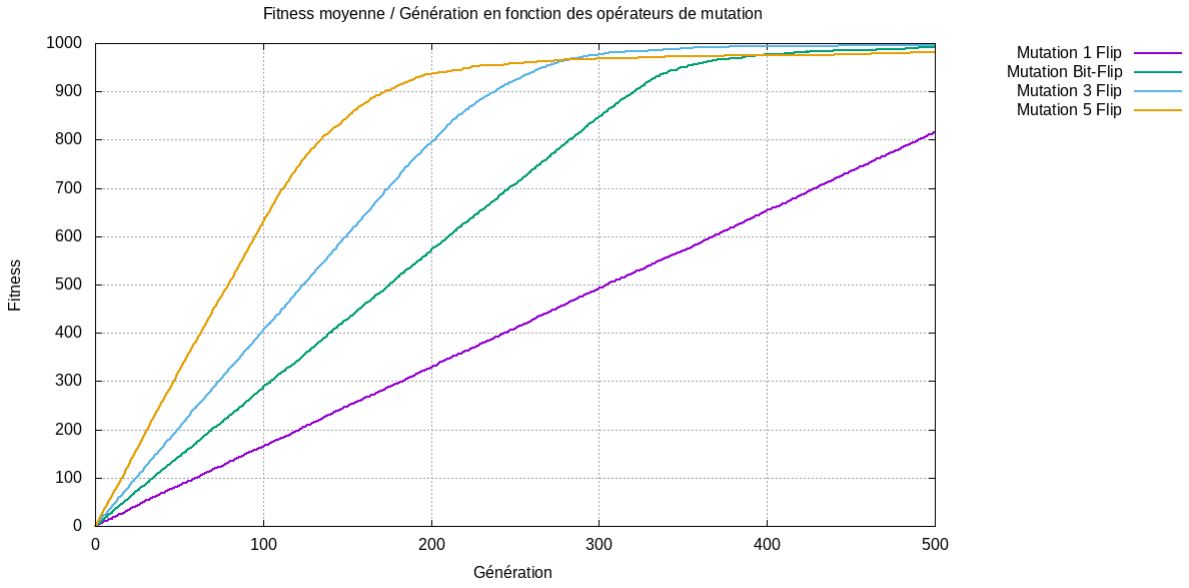

FIGURE 4 – Évolution de la Fitness Max en fonction des opérateurs de mutation

- Discussion :

5.2 Fitness en fonction des opérateurs du croisement

La figure 5 montre l'évolution de la Fitness max obtenue sur 20 exécutions différentes en fonction des opérateurs du croisement (croisement simple, uniforme et mono-point).

- Paramètres de l'algorithme : Nous avons utilisé les mêmes paramètres que (5.1).

  Opérateur de mutation : Mutation Bit-Flip.

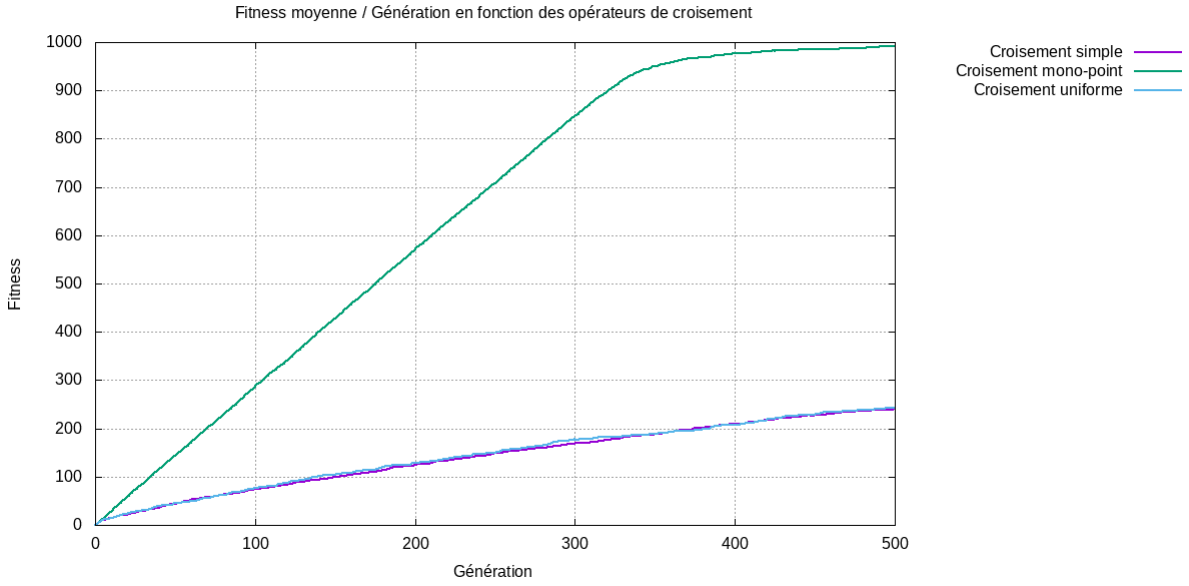

FIGURE 5 – Évolution de la Fitness Max en fonction des opérateurs du croisement

- Discussion :

5.3 Autres testes

Plusieurs autres testes peuvent être effectués sur l'algorithme génétique afin de bien comprendre son fonctionnement et ses comportement : évolution de la fitness en fonction de la taille de la population et/ou individu, évolution de la fitness en fonction de la probabilité de mutation, en fonction des opérateurs de sélection et remplacement, etc.

Les figures 6 et 7 montrent l'évolution de la fitness/génération en fonction des différents opérateurs de sélection et de remplacement. Sur ces deux figures, on remarque que presque dans tous les cas la sélection des 2 meilleurs individus et le remplacement des 2 plus mauvais est meilleur

que les autres opérateur contrairement au opérateurs de mutation qui dépendent d'une manière directe de l'itération.

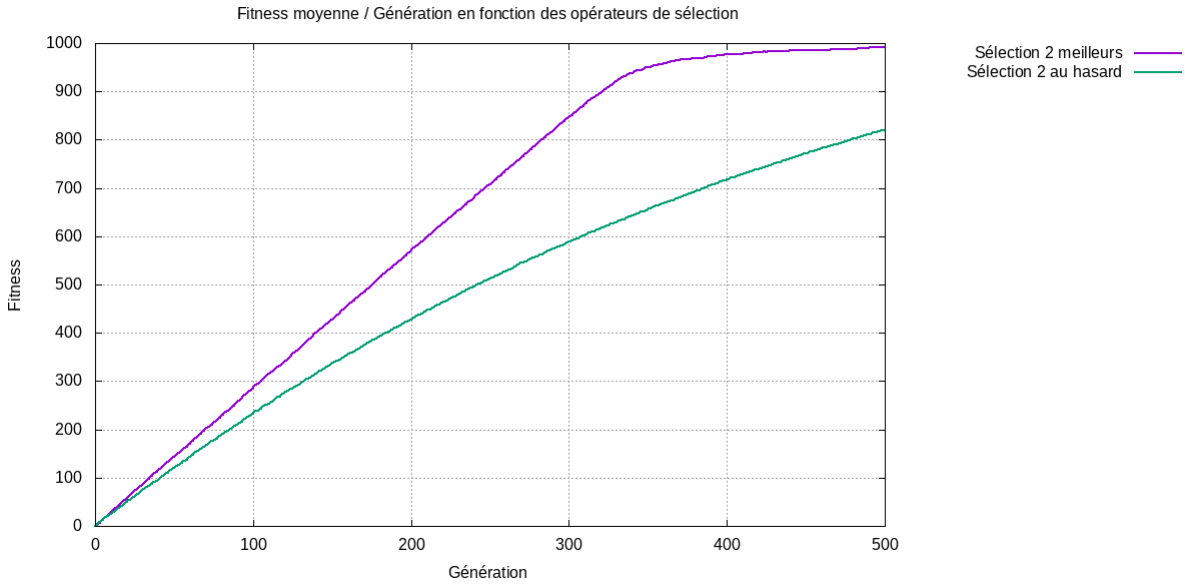

FIGURE 6 – Évolution de la Fitness Max en fonction des opérateurs de sélection

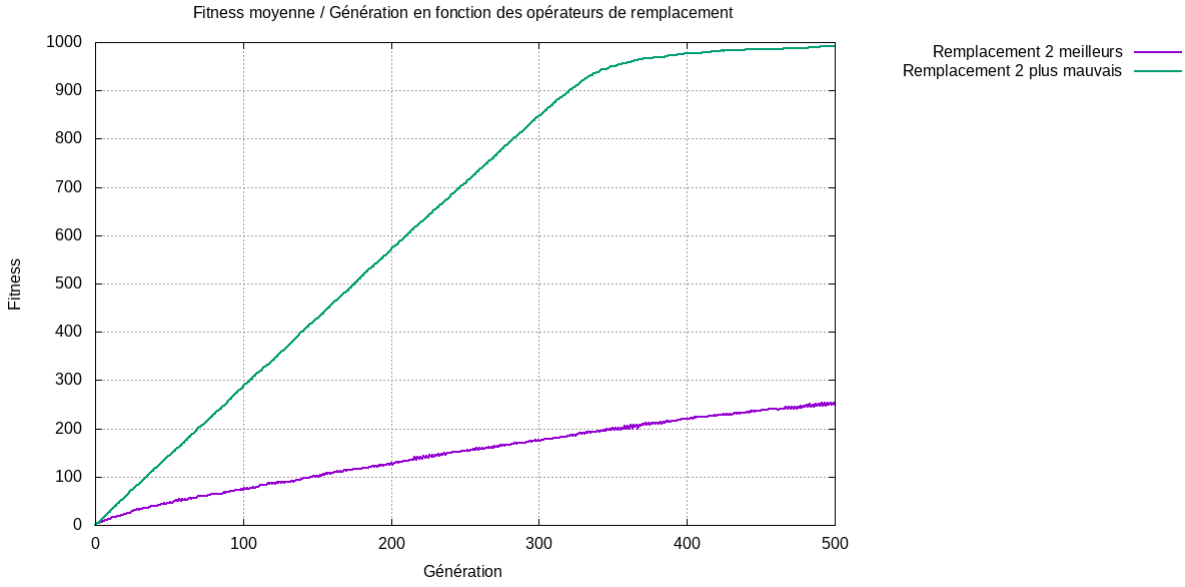

FIGURE 7 – Évolution de la Fitness Max en fonction des opérateurs de remplacement

6 Problématique du choix des opérateurs

Le choix de l'opérateur à utiliser comme nous l'avons vu dans la section précédente affecte d'une manière directe la Fitness ou l'efficacité de l'algorithme génétique d'une manière générale.

Dans cette étude, nous allons nous intéresser aux opérateurs de mutation. Nous avons vu dans la partie précédente que l'efficacité des opérateurs dépende directement de la taille du problème et de l'instant lorsque l'opérateur est utilisé. Nous avons vu qu'au début de l'exécution (0 - 200 génération) l'opérateur 5 Flip était plus améliorant que les autres opérateurs, mais à partir de la génération 200 nous avons remarqué l'inverse.

Une amélioration qui peut augmenter considérablement la Fitness serait donc de mettre en place un mécanisme qui permet de choisir intelligemment à un instant t l'opérateur o à utiliser.

Il existe plusieurs méthodes qui permettent à partir d'un certain nombre de données en entrée de décider de quelle action effectuer à un instant t afin de maximiser le gain.

Définition du problème : Nous avons :

- 4 opérateurs de mutation : 1 Flip, Bit-Flip, 3 Flip, 5 Flip.
- Un nombre max de générations.
- Une fonction Fitness (fitness cumulée qu'on cherche à maximiser).

Objectif : À chaque itération (génération), l'algorithme doit pouvoir choisir le meilleur opérateur de mutation qui permet de maximiser la fitness cumulée.

7 Algorithme du Bandit

Problème du bandit : Afin de simplifier le problème du bandit, nous allons prendre le cas le plus simple. On considère une machines à sous à 2 bras, notés A et B. L'action du bras A (resp. B) donne lieu à un gain qui n'est pas connu a priori, et le joueur cherche à actionner le bras le plus favorable. L'idée consiste à choisir un bras au hasard, mais en modifiant au cours du temps la loi de probabilité du choix du bras, i.e. Si à l'étape n, le bras actionné avait une probabilité p_n d'être choisi, s'il produit un gain, on lui affecte à l'étape suivante une probabilité plus importante $p_{n+1} \geq p_n$.

Algorithme de bandit : L'algorithme du bandit tient son nom du problème expliqué dans le paragraphe précédent où le joueur cherche à maximiser son gain. L'algorithme du bandit est un modèle d'apprentissage par renforcement qui consiste à un instant t de choisir l'action a dont il a appris qu'elle récompense beaucoup.

Dans notre cas, les k-bras de la machine à sous (multi-armed bandit) sont les différents opérateurs de mutation et à chaque génération l'algorithme choisit un opérateur et ajuste les probabilités du choix des opérateurs en fonction des gains.

Récompense : La récompense est la valorisation immédiate d'une action dans un état donné : $R_t = Fitness_t - Fitness_{t-1} + d$
tel que d est la valeur de décalage qui permet de ne pas avoir une récompense négative.

Il existe plusieurs approches (politiques) pour résoudre ce problème : $\varepsilon-$Greedy, Probability Matching, Upper-Confidence Bound (UCB), Lower Upper Confidence Bound (LUCB), etc.

Dans notre cas, nous avons implémenté les deux algorithmes "Sélection par roulette proportionnelle" et "UCB".

7.1 Roulette proportionnelle pour le choix d'opérateur

Au lancement de l'algorithme, tous les opérateurs ont une probabilité P_{init} d'être sélectionné, tel que $P_{init} = \frac{1}{nb\_operateurs}$.

Dans notre cas, nous avons 4 opérateurs, donc la probabilité initiale de chaque opérateur $P_{init} = \frac{1}{4} = 0.25$.

L'algorithme choisit donc aléatoirement un opérateur, calcule l'amélioration/récompense immédiate et le gain, ensuite ajuste les probabilités des opérateurs suivant cette formule :

$\pi_i^{t+1} = P_{min} + (1 - N.P_{min})\frac{u_i^{(t+1)}}{\sum_{k=1}^{N} u_k^{(t+1)}}$, tel que :

π_i^t : probabilité d'appliquer l'opérateur i à l'itération t.

P_{min} : pour assurer une probabilité de sélection non nulle pour tous les opérateurs.

$0 \leq p_{min} \leq \frac{1}{nb\_operateur}$.

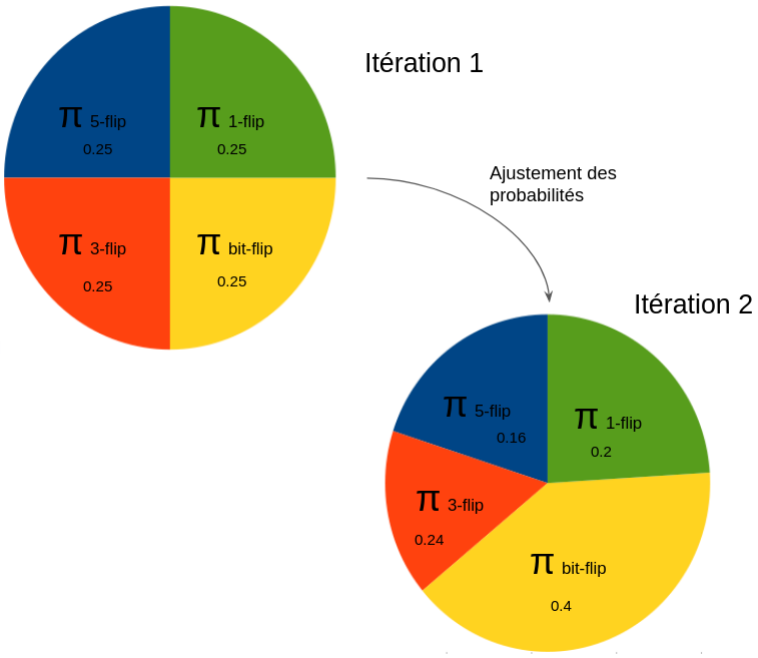

FIGURE 8 – Sélection des opérateurs par roulette proportionnelle - Probability Matching

7.2 UCB

Avec l'algorithme UCB, l'utilisateur calcule la moyenne empirique de la récompense pour chacun des opérateurs :

$$X_j = \frac{1}{T_j} \sum_{i=1}^{t} r_i x_{aj=i}.$$

Tel que :
t : désigne le nombre d'essais réalisés par l'utilisateur.
T_j : le nombre d'essais sur l'opérateur j.
r_i : désigne la récompense obtenue lors de l'essai i.
x : désigne la fonction indicatrice qui indique que l'opérateur j a été choisi pour l'essai i.

Pour calculer l'index, on ajoute un biais qui permet à l'algorithme d'explorer les différents opérateurs :
$B_j = X_j + A_j$, tel que : $A_j = \sqrt{\frac{2log(t)}{T_j}}$

8 Analyse et discussion des résultats obtenus

Dans cette partie, nous allons présenter les différents résultats obtenus en utilisant les deux algorithmes présentés dans la section précédente, et enfin nous analysons et comparons ces résultats

avec les résultats obtenus dans la section 5.

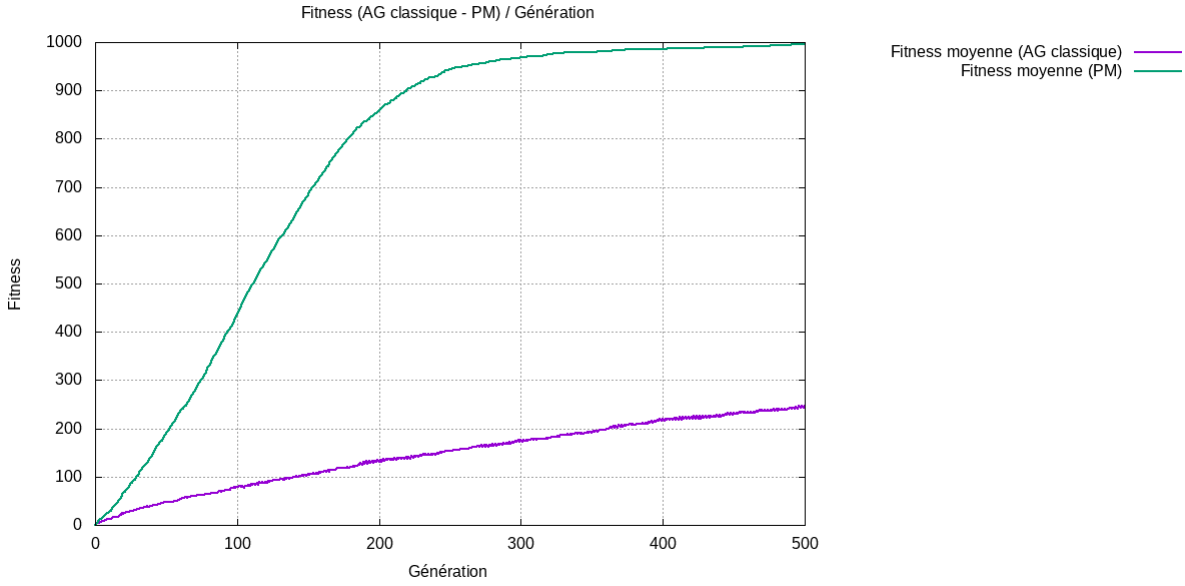

FIGURE 9 – Fitness moyenne de l'algorithme génétique classique et PM (Probability Matching)

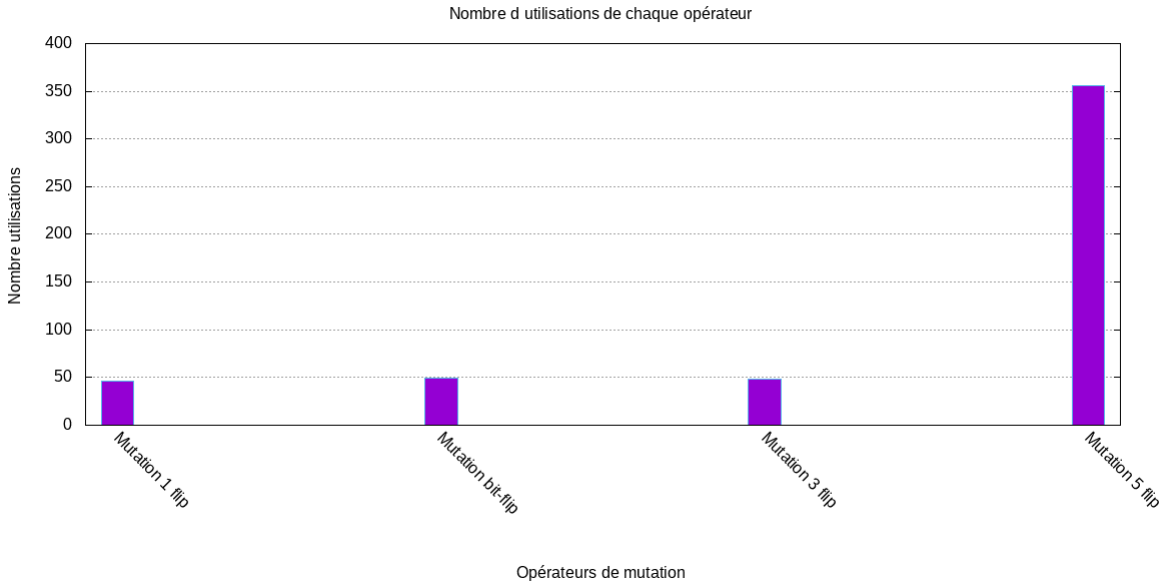

FIGURE 10 – Moyenne d'utilisation de chaque opérateur de mutation par PM (Probability Matching)

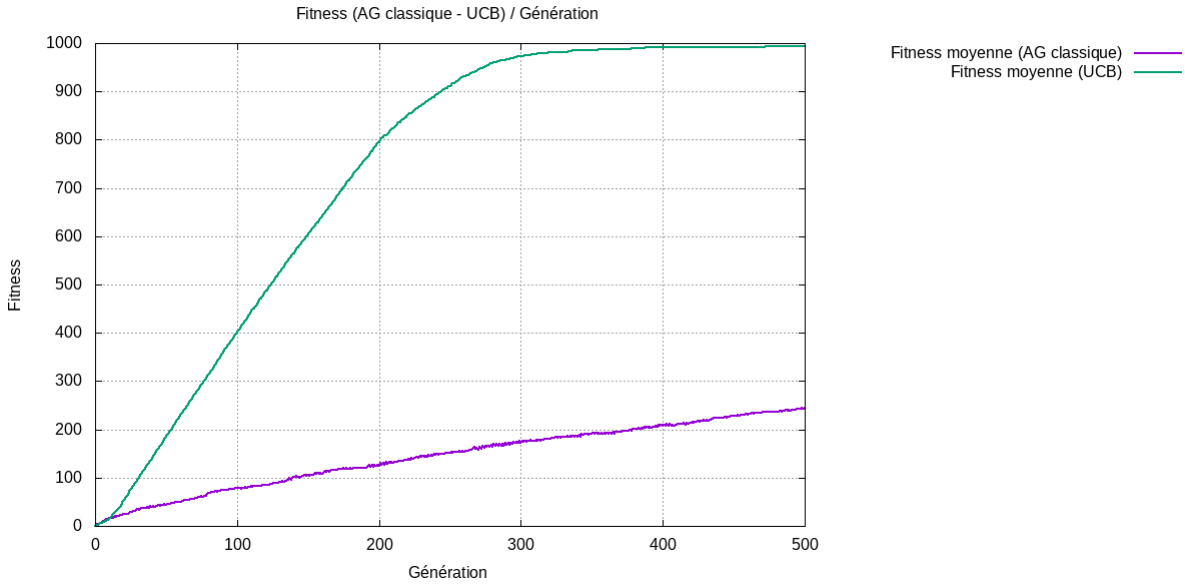

FIGURE 11 – Fitness moyenne de l'algorithme génétique classique et UCB

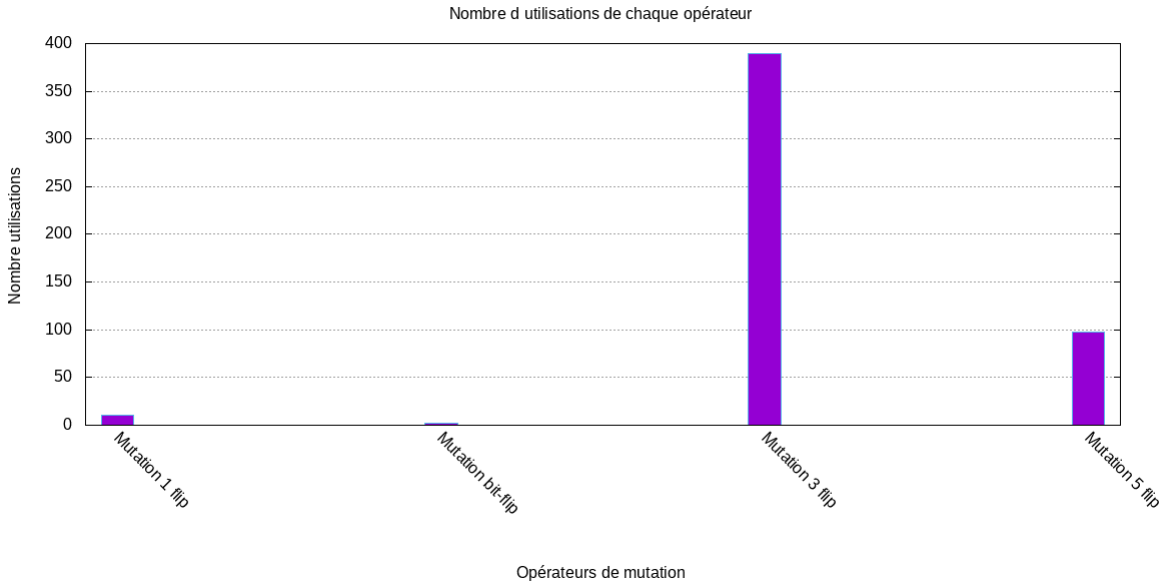

FIGURE 12 – Moyenne d'utilisation de chaque opérateur de mutation par UCB

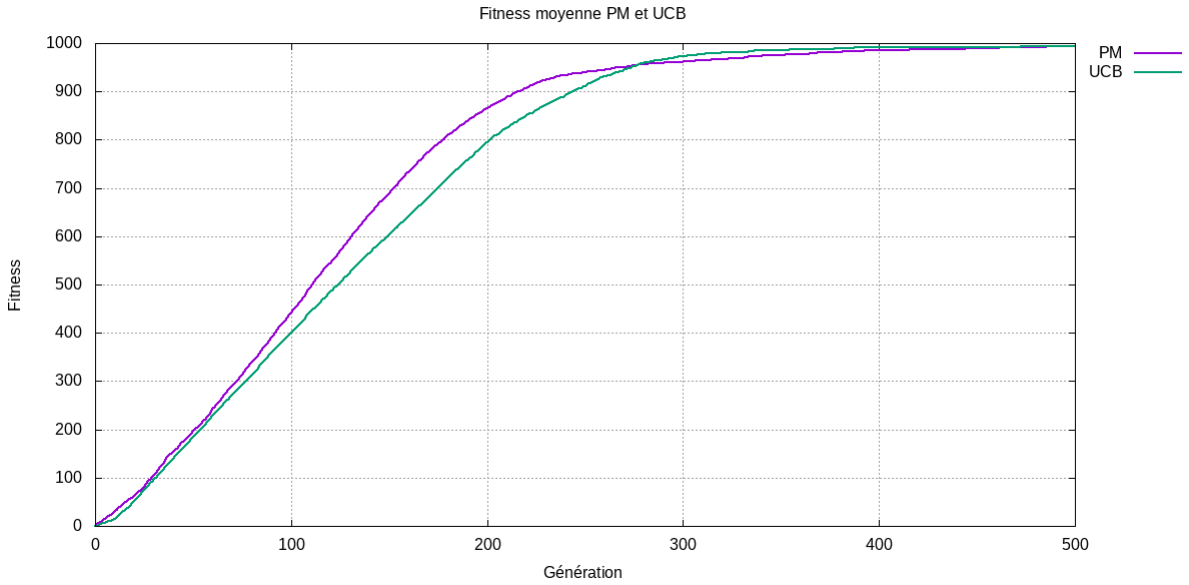

FIGURE 13 – Fitness moyenne PM et UCB

9 Conclusion

9.1 Perspectives d'amélioration