Compte rendu de projet

1. Choix de conception

* Nous avons décidé de concevoir la population avec deux structures distinctes, la première structure dénommée *Element* et une seconde structure *Population*. Le premier contient le premier élément de la liste de bits constituant un individu sous l’appellation *Elem\_val*, la qualité de l’individu nommée *Elem\_quality*, et l’adresse de l’élément suivant et précédent de la liste doublement chainée. L’objet *Population* contient lui les adresses du début et de la fin de la liste doublement chaine constituée par *Element*. Nous avons décidé d’adopter cette structure afin de pouvoir aisément manipuler la liste dans les deux sens de lecture, nous aurions aussi pu nous passer de la structure *Population* mais avons décidé de la conserver afin d’améliorer la lisibilité des algorithmes mais aussi pour éviter d’utiliser des emplacements de mémoire dans chaque item de la liste d’individu pour deux adresses identiques.
* Suite à des problèmes avec les fonctions aléatoires de forte ressemblance entre des tirages aléatoire, nous avons réalisé une fonction aléatoire que nous appelons au besoin en réinitialisant la chaine aléatoire à chaque fois.
* Nous avons réalisé le *quicksort* de sorte qu’il n’échange que les valeurs et non les éléments de la liste doublement chainée population. Cette technique est plus optimisée, plus lisible et limite les erreurs lors du tri.

1. Algorithmes des sous-programmes

Voir dossier algorithme.

1. Jeux d’essais

Voir Jeux d’essais 1 ou 2 pour la fonction f1

Voir Jeux d’essais 3 pour la fonction f2

Voir Jeux d’essais 4 pour la fonction f3

1. Analyse des essais

* Nous remarquons aves les jeux d’essais 1 et 2 que la population devient une impasse évolutive à mesure que les valeurs s’homogénéisent.
* Le phénomène d’homogénéisation tend à être réduit avec la fonction f2 et la fonction f3.
* Quand le taux de sélection est important, l’homogénéisation est plus lente à apparaitre et l’individu idéal est plus fréquent.