Rapport IA- Approximation de fonction par Algorithme génétique.

Paul COIFFET TDC A3

Quelle est la taille de l'espace de recherche (utiliser une notation scientifique) ?

On utilise l'espace de recherche indiqué dans l'énoncé

$$A = \{\ a \in \mathbb{R} \mid a \in]0,1[\ \}$$
$$B = \{\ b \in \mathbb{N} \mid b \in\ [1,20]\ \}$$
$$C = \{\ c \in \mathbb{N} \mid c \in\ [1,20]\ \}$$

Quelle est votre fonction fitness ?

La fonction fitness va lire le fichier température_sample.csv, récupérer les valeurs dans la première colonne et les comparer à la valeur de notre fonction après avoir calculé avec nos valeurs a,b et c de notre population.

Plus la différence entre le résultat de notre fonction et la valeur du fichier (donc le coût) est faible, plus nos valeurs a,b,c sont fidèles à ce que l'on devrait trouver.

```
#Fonction de calcul de coût pour un triplet
    def Cout(self):
        cout=0
        file = np.loadtxt('temperature_sample.csv',delimiter = ";", skiprows=1)
        for i in range(len([row[0] for row in file])):
            somme=0
            for n in range(self.c+1):
                somme+=(self.a**n) * math.cos((self.b**n)*math.pi*file[i][0])
            cout+=abs(somme-file[i][1])
        return cout
```

Décrivez les opérateurs mis en œuvre :

On a deux opérateurs qui sont mis en œuvre : un croisement et une mutation.
Le croisement nous permet d'avoir 3 valeurs filles à partir de 3 valeurs prises aléatoirement dans la population.
La mutation permet d'obtenir une valeur fille en inversant les valeurs b et c d'un triplet pris aléatoirement dans la population.

```
#Donne 3n nouveaux individus par croisement génétique
def NCroisement2(n):
    i = 0
    while(i<n):
        r1=random.randint(0, len(listIndividus) - 1)
        r2=random.randint(0, len(listIndividus) - 1)
        r3=random.randint(0, len(listIndividus) - 1)
        f1=Triplet(listIndividus[r1].a, listIndividus[r2].b, listIndividus[r3].c)
        f2=Triplet(listIndividus[r2].a, listIndividus[r3].b, listIndividus[r1].c)
        f3=Triplet(listIndividus[r3].a, listIndividus[r1].b, listIndividus[r2].c)
        listIndividus.append(f1)
        listIndividus.append(f2)
        listIndividus.append(f3)
        i+=1

#Donne n nouveaux individus par mutation génétique
def NMutation(n):
    i=0
    while(i<n):
        r = random.randint(0,len(listIndividus)-1)
        f1 = listIndividus[r]
        f2 = Triplet(0,0,0)
        f2.a=f1.a
        f2.b=f1.c
        f2.c=f1.b
        listIndividus.append(f2)
        i+=1
```

Décrivez votre processus de sélection :

C'est un algorithme de tri qui va garder les 20 meilleurs solutions (les valeurs dont le coût est le plus faible) dans notre population puis va supprimer toutes les autres valeurs.
Elle permet de sélectionner les meilleurs individus pour les remuter et les recroiser par la suite.

```
#Fonction qui d'une part actualise le cout de tous les individus puis garde les 20 meilleurs individus dans une nouvelles liste
def CompareCout2():
    for i in range(20):
        minimum = i
        for j in range(i + 1, len(listIndividus)):
            if(listIndividus[minimum].cout>listIndividus[j].cout):
                minimum = j
        temp = listIndividus[minimum]
        listIndividus[minimum] = listIndividus[i]
        listIndividus[i] = temp
    del(listIndividus[20::])
```

Quelle est la taille de votre population, combien de générations sont nécessaires avant de converger vers une solution stable ?

J'ai choisi une population de base de 100 que je tri directement pour ressortir les meilleurs candidats.
Ensuite, je les croise et les mute pour avoir au final 210 individus (90 par croisement et 100 par mutation).
J'ai besoin de 30 génération pour avoir une solution stable.

Combien de temps votre programme prend en moyenne ?

Mon code prend en moyenne 3,5 secondes pour s'effectuer

Discutez vos différentes solutions qui ont moins bien fonctionnées

Mon plus gros problème était l'utilisation de liste et de dictionnaire. Je n'arrivais pas à faire fonctionner mon code et à trouver un moyen de combiner le coût et les valeurs. J'ai donc décidé de recommencer à zéro en utilisant cette fois ci une classe Triplet qui prend en attribut le coût que je calcule avec une fonction à l'intérieur de la classe. Cela m'a énormément simplifié la vie sur ce code.
J'ai eu au départ des problème pour mes fonctions de croisement et de mutation, je n'arrivais pas à permuter les b et les c et je me suis rendu compte que c'était des problèmes d'indexage. J'ai donc rectifié et c'était bien.
Un autre problème que j'ai rencontré, c'est la fonction CompareCout2, j'ai mis du temps à la faire, ça ne compilait pas ou ça ne fonctionnait pas bien car mes individus n'étaient pas triés.