TD noté

Métaheuristiques pour l'optimisation d'un clavier

Binôme : NOYER Madison, SUEL Benjamin.

2. Travail à réaliser.

2.1 Conception.

La méthode utiliser est le recuit simulé.

2.1.1 Etat/Individu.

Q. 1 :

26 lettres, 40 positions.

Nous imposons que la configurations optimales correspond à position la touche T1 à la position $P_{1,1}$, la touche T2 à la position $P_{1,2}$.

Dans cette configuration, les connexions imposées correspondent aux connexions avec leurs voisins immédiat.

$P_{i,j}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
2	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20
3	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30
4	T31	T32	T33	T34	T35	T36	T37	T38	T39	T40

(optimal d'après notre hypothèse)

On a donc 40 variables Tlij.

avec DTL € [0,26] représentant les 26 lettres de l'alphabet latin. 0 = rien

et i € [1,4] représentant la ligne.

et j € [1,10] représentant la colonne.

Q. 2 :

L'espace de recherche correspond au nombre de configuration possible. Il y a 26 lettres à placer, ce qui fait 26! configurations possibles.

2.1.2 Fonction énergie/objectif/d'adaptation.

Q.3 :

Fonction objective : pour une paire donnée, calculer la distance ce ces deux lettres sur le clavier qu'on multiplier leur fréquence présent dans le fichier freqbigramme.txt.

Somme des résultats pour les paires obtenues pour trouver la meilleure optimisation.

Q.4 :

quand on tape avec un doigt :

$$\sum dist\left(P_{i,j}, P_{i',j'}\right) * freq\left(P_{i,j}, P_{i',j'}\right)$$

quand on tape avec deux doigt :

$$\sum dist\left(P_{i,j}, P_{i',j'}\right) * freq\left(P_{i,j}, P_{i',j'}\right) * m_{i,j}$$

$$m_{i,j} = \begin{cases} 1 \quad si\left(1 \leqslant i \leqslant 5 \; et \; 1 \leqslant i' \leqslant 5\right) ou \left(6 \leqslant i \leqslant 10 \; et \; 6 \leqslant i' \leqslant 10\right) \\ 0 \text{ sinon} \end{cases}$$

dist(A,B) : La distance entre A et B

2.1.3 Précisez et justifiez.

Q. 5 .a : Pour le recuit simulé :

1. Pour l'initialisation de l'état de départ on est partie sur un clavier générée aléatoirement.

2. Les mouvement possible sont l'inversion de la position des deux lettres choisie aléatoirement. (fonction voisin)

3. Pour la valeur de départ de la température :

1. faire 100 perturbations au hasard
2. évaluer la moyenne des variations absolues
3. choisir le taux d'acceptation initiale
    1. Si on suppose que la configuration initiale est médiocre:T0 = 50 %
    2. Si on suppose que la configuration initiale est bonne : T0 = 20 %
4. Prendre T0 depuis exp(- ΔE/ T 0)= T0

Dans notre code : (fonction T0)

```
1.Si on suppose que la configuration initiale est médiocre
2.Si on suppose que la configuration intiale est bonne
Choisir le taux d'acceptation initiale (saisir 1 ou 2) 2
```

Comme on peut le voir ici, nous donnons à l'utilisateur la possibilité de choisir la configuration initiale. Si la valeur saisie ne correspond ni à 1 ni à 2, nous supposons que la configuration initiale est bonne.
Le calcule de T0 peut mettre quelques minutes..

4.La fonction d'évolution de la température permet de contrôler la vitesse à laquelle la température diminue au fil du temps. Une fonction de refroidissement trop rapide peut empêcher l'algorithme de trouver la meilleure solution possible, tandis qu'une fonction de refroidissement trop lente peut rendre l'algorithme très lent.

Dans notre cas, nous avons utilisé la loi géométrique pour l'évolution de la température : $T_{k+1} = 0.9 * T_k$. Le graphique ci-joint illustre l'évolution de la température au fil des itérations.

5. Dans le recuit simulé, plusieurs arrêts sont possibles :

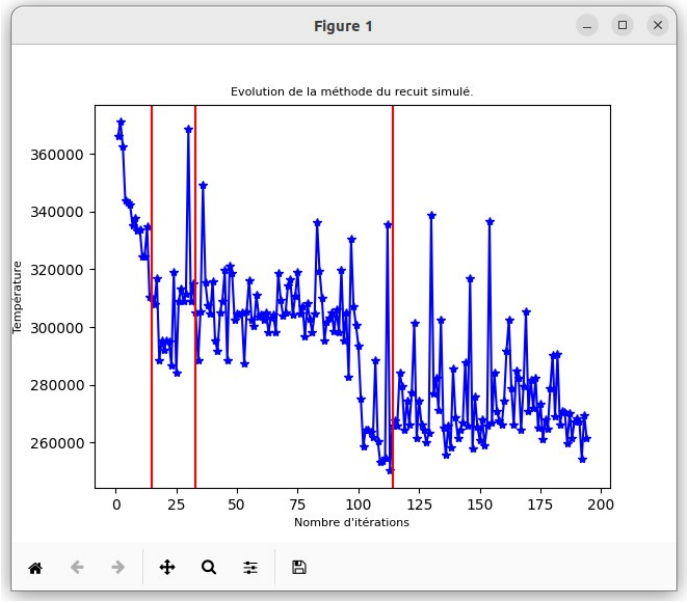
- L'arrêt par nombre d'itérations, ce qui signifie que l'algorithme exécute un nombre d'itérations donné par l'utilisateur.
- L'arrêt par une valeur de temps donnée, ce qui signifie que l'algorithme continue de s'exécuter jusqu'à ce que le temps donné s'écoule.
- L'arrêt lorsque la différence d'énergie entre l'état actuel et suivant est inférieure à une certaine tolérance.
- L'arrêt par stagnation de la température, ce qui signifie que l'algorithme s'arrête lorsque la température n'évolue plus.

Dans notre cas, nous utilisons l'arrêt par stagnation. À partir du moment où le calcul de l'énergie du clavier stagne, le programme s'arrête. Cependant, nous avons ajouté une sécurité pour arrêter le programme en cas de durée trop longue. Il suffit d'appuyer sur la croix de la page qui contient le graphique ou de faire un Ctrl+C dans le terminal. Avec cette technique, nous obtenons un résultat avec le nombre d'itérations effectuées jusqu'à présent.

6. Dans le fichier Readme, on trouve toutes les informations pour exécuter notre code correctement. De plus, on y trouve les bibliothèques à installer. Nous avons utilisé 2 bibliothèques :

- La bibliothèque Pandas. Cette bibliothèque nous permet de lire et de récupérer les informations aux lignes et colonnes souhaitées dans le fichier freqbigramme.txt.

- La bibliothèque Matplotlib. Cette bibliothèque nous permet de gérer l'aspect visuel de notre code (la fenêtre avec le graphe).



Comme on peut voir sur ce graphe, dès qu'un palier est passé, on affiche un trait rouge. Changement de palier de la température : si 12N perturbations ont été acceptées ou si 100N perturbations ont été tentées, avec N le nombre de paramètres à optimiser du problème.

```
Resultat obtenue avec 12 iterations
L'énergie trouver: 359718.558796266
Clavier obtenue (les "_" sont considérer comme des cases vides):
K L I D U M B Q _ V
H _ Z T W J G _ O _
_ P F _ _ _ _ _ R C
_ _ X N _ E _ S A Y
```

On peut voir que j'ai arrêté le programme volontairement après 12 itérations. On affiche le meilleur résultat obtenu après ces 12 itérations.

Fonctions utilisées dans notre code :
- def voisin(clavier) : retourne un clavier voisin du clavier entré en paramètre.
- def frequence(lettreL,lettreC): retourne la fréquence définie dans le fichier donné (freqbigramme.txt).
- def dist(lettreL, lettreC, clavier): retourne la distance entre deux lettres sur le clavier donné.
- def position(clavier, lettre): retourne la position de la lettre sur le clavier.
- def energie(clavier): retourne la température du clavier.
- def on_close(event):permet de fermer la fenêtre du graphe et d'afficher le résultat obtenu après x itérations.
- def handle_signal(signal, frame): permet de fermer correctement le programme avec Ctrl + C.
- def T0(): fonction qui permet de déterminer T0.
- def stop(): fonction qui permet de fermer correctement le programme.

2.2 Implémentation.

On a choisi le langage Python.