Tatoglu Emel
Villaume Lucas

L3 Informatique

Optimisation d’un clavier

Métaheuristique : Algorithme génétique

Table des matières

Index des figures

Tatoglu Emel
Villaume Lucas

Rappel du contexte

Soit les 26 lettres de l'alphabet à placer sur un clavier 4x10. Voici une représentation possible :

W		P	D		I	Z	O	N	
V	J	U		R	M		A		H
	X	Q	F		E	B	S	T	
K		C		L		Y	G		

Figure 1: Représentation d'une solution

Nous avons numéroté les touches du clavier, de 0 à 39, en parcourant les colonnes. Ici, la lettre W est affectée à la touche 0, la lettre V affectée à la touche 1, etc.

1. Etat / Individu

On cherche à affecter les 26 lettres de l'alphabet sur 40 touches du clavier, Il ne peut y avoir de lettre présente deux fois donc, certaines touches ne seront verront affectées aucune lettre.

Une représentation possible est d'utiliser 26 variables pouvant contenir des valeurs allant de 0 à 39. Ainsi, l'état représenté sur la Figure 1 prendrait cette forme :

x0 : 29 / x1 : 26 / x2 : 11 / x3 : 12 / x4:22 / … / x24 : 27 / x25 : 24

D'un autre point de vue, on peut voir cela comme une permutation de k éléments parmi un ensemble de taille n. Cela revient à calculer le nombre de permutation possible, on a donc :

$\frac{n!}{k! \times (n-k)!}$, en l'appliquant à notre cas on obtient $\frac{40!}{26! \times (40-26)!}$ = 23 206 929 840

Donc l'espace de recherche pour cette représentation est de 23 206 929 840, soit approximativement 2,32e10.

2. Algorithme génétique

2.1 Fonction d’adaptation

La fonction d’adaptation, ou fitness, est une fonction permettant d’évaluer l’individu sur son génotype afin de le comparer à ses semblables. Dans notre cas, la fonction d’adaptation s’appuie sur la fréquence d’apparition des bigrammes. Si l’on juge que la position pour un bigramme est adapté, alors on ajoute la valeur à la fitness de l’individu.

Par exemple, soient les bigrammes (A,L) = 12* / (L,A) = 3* / (E,S) = 10* / (S,E) = 10*. Si on décrit les bigrammes (A,L) et, par symétrie (L,A) comme adaptés, alors la fitness de l’individu sera égal à : 12 + 3 = 15.

*Les valeurs des bigrammes sont arbitraires

2.1.1 Cas d’une utilisation à une main

Lorsque l’on utilise un clavier avec une seule main, voir un seul doigt, il est plus optimal de placer les lettres formant les bigrammes les plus fréquent proche l’un de l’autre. Nous définissons qu’une touche est proche d’une autre si cette dernière ce trouve dans un rayon d’une touche. Si l’on prend comme exemple la Figure 1, alors la touche J est proche des touche : W,V,X, P, U et Q.

La formule prend cette forme :

$\sum_{i=0}^{25}\sum_{j=0}^{25} proche(i,j)*val(i,j)$, avec proche(i,j) une fonction booléenne évaluant si i est proche de j, val(i,j) la fréquence du bigramme (i,j).

2.1.2 Cas d’une utilisation à deux mains

Lorsque l’on utilise un clavier avec deux mains (un doigt par main), il est intéressant de diviser le clavier en deux :

- coté gauche : touche allant de de 0 à 19
- coté droit : touche allant de 20 à 39

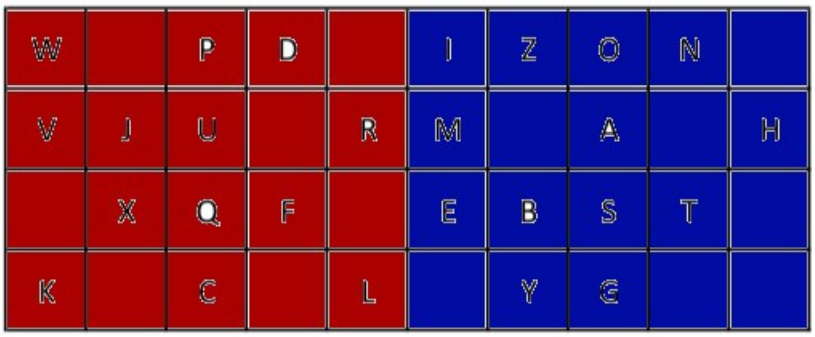

Figure 2: Séparation du clavier

Dans ce genre de configuration, il est plus optimal de répartir les lettres composant un bigramme sur les deux parties, ainsi une main peut se positionner pendant que l'autre tape ce qui permet d'enchaîner les lettres.

La formule prend cette forme :

$\sum_{i=0}^{25}\sum_{j=0}^{25} sep(i,j)*val(i,j)$, avec sep(i,j) une fonction booléenne évaluant si i est sur une autre partie du clavier de j, val(i,j) la fréquence du bigramme (i,j).

2.2 Initialisation

Afin de tirer les meilleurs résultats d'un algorithme génétique, il est important d'avoir une grande diversité dans le génotype de la population. Même si cela peut prendre plus de temps, en augmentant le nombre de générations nécessaires, cela permet de réduire les chances de tomber sur un extremum local. L'une des solutions envisagé pour l'initialisation des états de départ est de s'en remettre au hasard. La propriété principale du hasard est son caractère aléatoire et imprévisible, qui conduit à une diversité importante dans les résultats possibles, ce qui entraîne une hétérogénéité de notre population.

Notre espace de recherche étant de 2,3e11, il est conseillé d'avoir un nombre d'individu conséquent pour les même raisons d'extremum local. D'après nos recherche, il existe plusieurs formules empirique déterminant le nombre d'individu d'une population en fonction du nombre de variable composant son génome :

- 4 fois le nombres de variable du génotype.
- 10 fois le nombres de variable du génotype.
- 100 fois le nombres de variable du génotype.

L'utilisation d'une population de 2600 individus nécessiterait une puissance de calcul importante pour être efficace. Cependant, d'autres recherches ont montré qu'une population plus petite était suffisante. Afin de concilier ces résultats, nous avons opté pour une population de 260 individus.

2.3 Opérateurs

2.3.1 Opérateur de sélection

L'opérateur de sélection des parents possède un rôle important dans les résultats fournit par l'algorithme, il est donc primordial de bien choisir afin de nous préserver de certains biais. Différents opérateurs sont disponibles tels que la *sélection par roue biaisée*, *par rang*, la *sélection élitiste* ou *uniforme*, chacun ayant ses avantages et ses inconvénients. Dans cette section, nous allons nous concentrer sur la **sélection par tournois** sans comparer les différents opérateurs entre eux. Un tournois consiste en une sélection des x meilleurs individus parmi un groupe de k. Cet opérateur est particulièrement efficace pour des populations hétérogènes, au détriment de la diversité de celle-ci. Dans notre cas, le tournois définira deux parents parmi un groupe de quatre individus, ce qui signifie que 50% de la population transmettra son génotype. Cela réduira en partie la perte de diversité. De plus, pour contre-balancer cette potentielle perte, nous introduirons plus loin un opérateur de mutation.

2.3.2 Opérateur de croisement

Pour rappel, le génotype d'un individu est une permutation de 26 valeurs dans l'intervalle [0,39], il est donc important de prendre en compte cette contrainte lors des croisements afin de respecter les propriétés d'une permutation. Quelques opérateurs permettent de cela, notamment *Ordered Crossover* (OX), *Cycle Crossover* (CX) ou *Partially Blind Crossover* (PBX). Nous allons nous concentrer sur *Partially Mapped Crossover* (PMX), un opérateur ayant pour qualité de préserver les caractéristiques des parents tout en croisant les génotypes. Ainsi, cela est particulièrement utile pour les problèmes de permutation, où l'ordre et la position des éléments dans la séquence peuvent jouer un rôle crucial dans la qualité de la solution. L'un des inconvénients de l'opérateur PMX est sa convergence vers un extremum local si la population possède un génotype homogène, ce qui ne nous affecte que peux étant donner le caractère hétérogène de la population de départ (cf. 2.2 Initialisation).

Chaque couple produira deux enfants issue du croisement de leurs génotypes. La population post-croisement culminera donc à 150 individu.

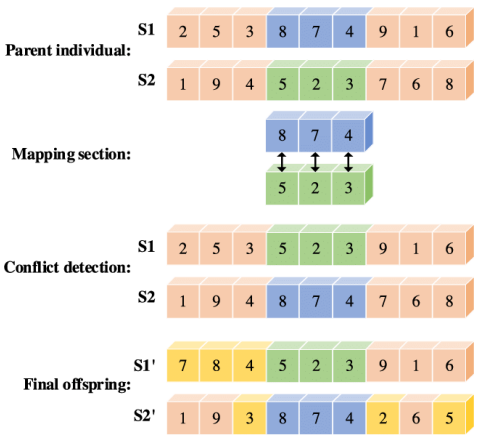

Figure 3: Exemple d'un croisement avec PMX

2.3.3 Opérateur de mutation

La mutation est une étape importante pour éviter de converger vers un optimum local. Elle compense la perte de diversité causée par d'autres opérateurs tels que la sélection des parents. La mutation la plus courante pour les permutations est la "mutation par inversion", qui inverse deux valeurs ou plus. Cependant, dans notre cas, nous avons une permutation de 26 valeurs parmi 40 possibles. La plus grande perte de diversité provient des 14 valeurs non utilisées. Pour remédier à cela, nous avons sélectionné un opérateur de ‘mutation par insertion’. Si l'individu subit une mutation, alors un nombre aléatoire, un index i, est généré dans l'intervalle de 0 à 25, ainsi qu'un autre nombre aléatoire, un entier e, dans l'intervalle de 0 à 39. Si la valeur de e existe déjà dans le génotype de l'individu, alors les valeurs des index i et de la valeur e sont inversées. Cependant, si la valeur de e n'existe pas encore dans le génotype, elle remplace la valeur actuelle de l'index i. Dans le cas où e est égale à la valeur de l’index de i, un nouveau e est généré. Le taux de mutation est fixé empiriquement à 2 %.

2.3.4 Opérateur de remplacement

A ce stade, le nombre d’individu s’élève à 150. Nous allons réduire d’un tiers la population afin de retrouver la taille initiale de celle-ci. Afin de s’assurer que les enfants issue de cette génération soit mieux adapté que leurs aînés, un tournois est de nouveau organisé. Chaque groupe se compose de 2 individus faisant partie de la population de base et d’un enfant.

2.4 Critère d’arrêt

La condition d’arrêt d’un algorithme génétique peut dépendre du nombre de génération ou du temps d’exécution. Dans notre cas, nous avons opté pour la stabilisation de la fitness (fonction d’adaptation) des 10 meilleurs individus. Ainsi, si pendant 10 générations la moyenne des 10 meilleurs fitness ne change pas de plus d’un point, alors on considère qu’un optimum à été atteint. Ces valeurs ont été trouvé de manière empirique, aucune règle ou consensus n’existe à ce sujet.

Conclusion

Après de multiple essai, on se rend compte que les solutions sont similaires dans le fond, mais diffèrent sur certains point tous en ayant des fitness plus ou moins égales. On peut donc conclure en disant qu’il est possible d’optimiser d’avantage la positions des touches en modifiants la fonctions de fitness afin que celle-ci prennent en compte les trigrammes et autres données utiles.

Tatoglu Emel
Villaume Lucas

Bibliographie

Livres :

Goldberg, D. E. (1989). Genetic algorithms in search, optimization and machine learning.

Johann Dréo-Alain Pétrowski-Patrick Siarry-Éric Taillard. Metaheuristiques pour l’optimisation difficile.