Metaheuristics for Optimization
SERIES 1 : NK-LANDSCAPE MODELS

Romain Mencattini

29 septembre 2016

Table des matières

1 Introduction **3**

2 Hill-Climbing déterministe **3**
2.1 Algorithme 3
2.2 Résultats 4

3 Hill-Climbing probabiliste **6**
3.1 Algorithme 6
3.2 Résultats 7

4 Conclusion **8**

1 Introduction

Le but de ce tp est de pratiquer le problème des *NK-landscape models*. Plus particulièrement de voir l'effet qu'à la modification du paramètre K sur les résultats. Le dit paramètre sert à modifier la *rugosité*. On va donc utiliser deux méthodes : une déterministe et une probabiliste. Pour chacune de ces méthodes, on va faire varier le K avec :

$$K = 0$$
$$K = 1$$
$$K = 2$$

On pose que x est un vecteur de taille N :

$$x = (x_1, x_2, ..., x_N) \text{ avec } x \in \{0, 1\}$$

Il nous faut également une fonction de coût définie comme suit :

$$F(x) = \sum_{i=1}^{N-K} f_K(x_i, ..., x_{i+K})$$

On va donc faire correspondre des groupes de bits de taille K à une valeur $\in \mathbb{N}$ selon ce tableau donné dans le tp1 :

	0	1
f_0	2	1

	00	01	10	11
f_1	2	3	2	0

	000	001	010	011	100	101	110	111
f_2	0	1	1	0	2	0	0	0

Il nous reste à définir les voisins. Le voisinage d'un x est l'ensemble de tout ses voisins. Où un voisin est une chaîne de bits de la même taille que x, mais ayant un seul bit de différent par rapport à x.
Pour différencier deux solutions, nous allons utiliser la distance de Hamming, qui calcule le nombre de bits qui diffèrent entre deux résultats.

2 Hill-Climbing déterministe

2.1 Algorithme

Pour cet algorithme, on prend toujours la meilleure solution. Par meilleure solution, nous entendons qu'une fois que les voisins ont été trouvés, on choisit toujours celui avec la meilleure fonction de coût : i.e.

$$max(F(y)) \ \forall y \text{ où } y \in V(x)$$

Voici comme procède l'algorithme :

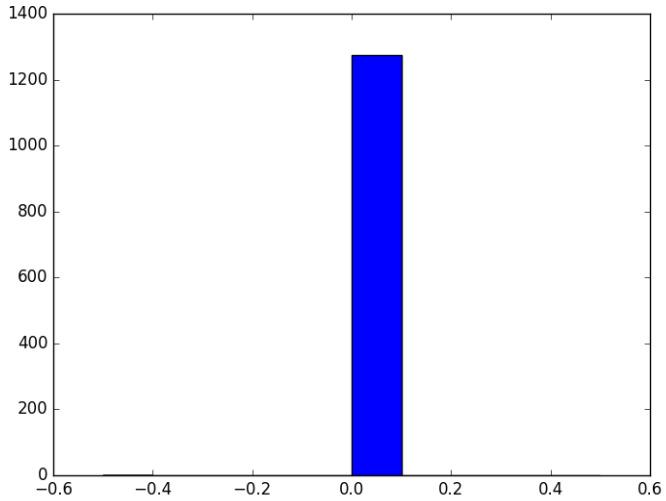
1. génération d'une solution aléatoire de taille N
2. génération des voisins
3. choix du voisins qui permet de maximiser $F(x)$
4. changement de la solution pour le voisin

L'algorithme s'arrête lorsque plus aucun voisin ne permet d'augmenter $F(x)$. Il s'agit donc d'un algorithme glouton car on prend toujours la meilleure solution jusqu'à arrêt du programme.

2.2 Résultats

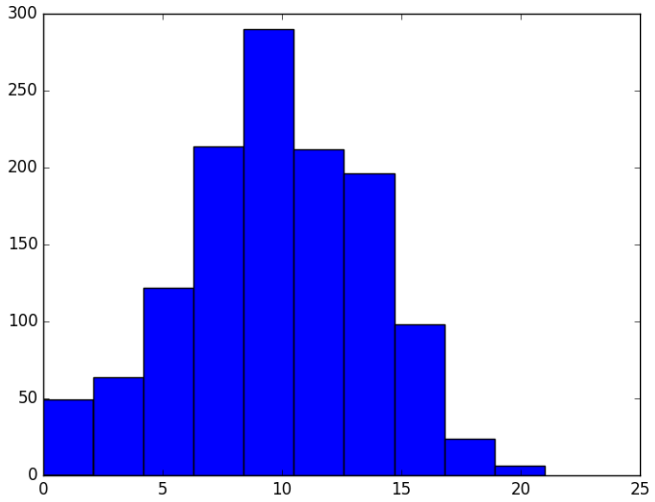
Voici les trois histogrammes obtenus pour chaque valeur de K. Nous avons procédé à 50 lancements pour valeurs.

Pour $K = 0$:



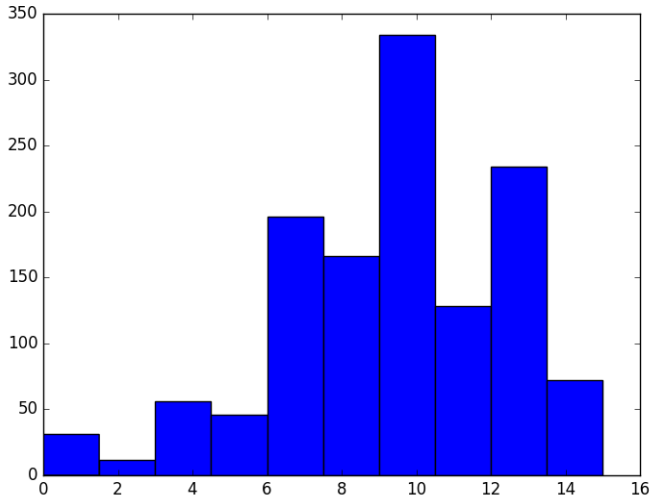
On remarque qu'il n'y a aucune rugosité pour ce graphique. En effet, l'algorithme trouve à chaque fois la même solution. La distance de Hamming est donc nulle dans tout les cas.

Pour $K = 1$:



Dans ce cas-ci, il y a plus de rugosité. On voit apparaître une loi normale centrée en 10. On a donc une majorité de résultats qui se trouvent au centre. Il y a peu d'histogrammes qui se trouvent très éloignés du centre. Donc une variance peu élevée.

Pour $K = 2$:



Cette fois-ci, on remarque qu'il y a une très grande rugosité. Il n'y a pas de loi normale, Les histogrammes sont répartis sur presque toute la largeur avec des hauteurs variable. Il y a donc énormément de solutions différentes trouvées par le programme déterministe. On peut supposer qu'il y a de nombreux maximas locaux et que la version déterministe tombe dedans souvent.

3 Hill-Climbing probabiliste

3.1 Algorithme

Dans cette version l'idée est un peu différente. En voici le déroulement :

1. on génère une solution aléatoire
2. on génère les voisins
3. on associe une probabilité aux voisins
4. on regarde s'il y a un voisin qui permet d'améliorer le meilleur résultat qu'on a eu et on revient au point 2.
5. si le 4. n'est pas vérifié on choisit (en tenant compte des probabilités) un voisin parmi ceux présents

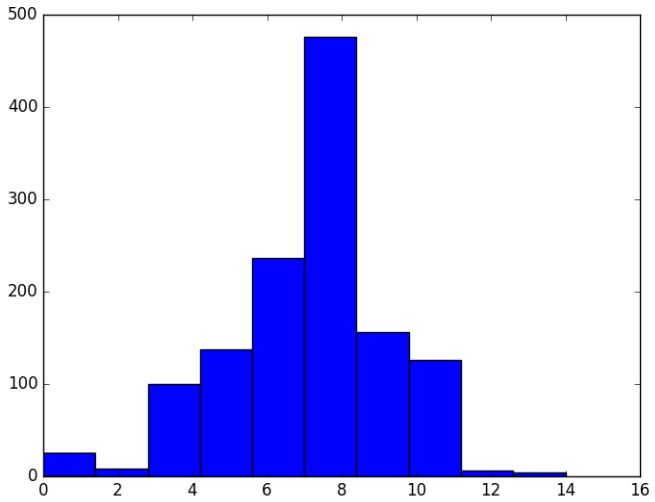
La condition d'arrêt est d'avoir effectué un certain nombre d'itération. On choisit empiriquement de le fixer à dix fois le nombre d'étapes nécessaires pour trouver une solution avec la version déterministe (en moyenne).

Les probabilités sont calculées comme étant la valeur de $F(x)$ divisée par la somme de toutes les $F(x)$.

Avec cet algorithme, il est possible de choisir des solutions moins bonnes. Mais cet inconvénient permet de ne pas rester coincer dans des minimas locaux. On a aussi une "intention" qui ; si on trouve une solution meilleure que toutes les précédentes, nous la fait choisir pour tenter de nous "indiquer" le chemin jusqu'à la solution optimale. On est sûr que si on laisse tourner l'algorithme un temps infini, on va forcément trouver le maxima global, ce qui n'est pas le cas avec la version déterministe.

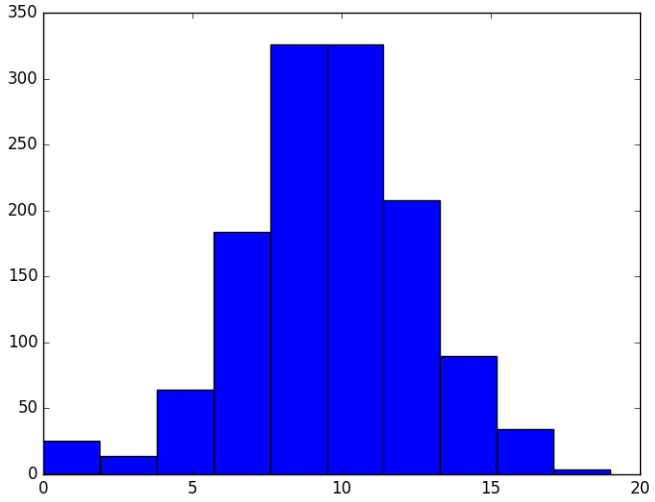
3.2 Résultats

Pour $K = 0$:



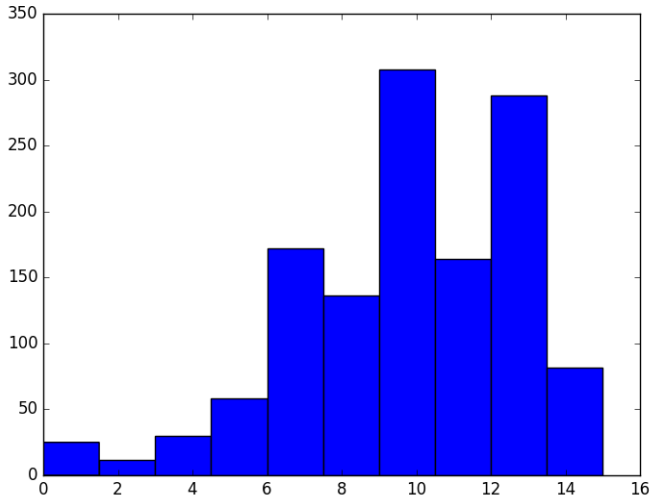
On voit un histogramme qui est bien plus représenté que les autres. Cependant c'est moins net que pour le déterministe. On n'a pas effectué assez de pas pour lisser le graphe.

Pour $K = 1$:



Dans ce cas-ci, les résultats sont ,encore une fois, moins bon que pour la version avec $K = 0$. On voit néanmoins que les résultats sont plus uniformes que pour la version déterministe. Il y a donc moins de résultats différents car les probabilités poussent les réponses à se ressembler.

Pour $K = 2$:



Les résultats sont très chaotiques comme pour la version déterministe.

4 Conclusion

On remarque avec les histogrammes que le fait d'augmenter K rend les solutions moins uniformes, plus aléatoire. Les graphes sont plus rugueux. Il est intéressant de noter que la version probabiliste pour $K = 1$ est moins rugueuse que la version déterministe. Nous pensons que cela est dû aux probabilités et à l'intention qui dirigent les solutions dans le même sens.

Pour $K = 0$, nous n'avions qu'une seule solution optimale et aucun maximas locaux : celle avec que des 0. Dans le cas de $K = 1$ et $K = 2$, il y a des maximas locaux, ce qui se traduit par des résultats moins uniformes. Concernant la version probabiliste, le choix de ne pas toujours choisir la meilleure solution la pénaliser pour $K = 0$.