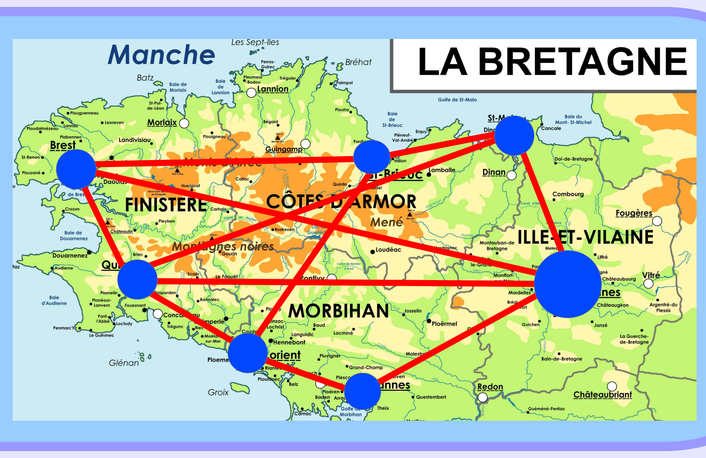
Les algorithmes génétiques au secours du voyageur de commerce

Jules GAUTHE – Mateo MELO – Théo ULVE : Lycée Kerneuzec, Quimperlé
Steven COSTIOU : Université de Bretagne Occitane

Le problème du voyageur de commerce :

Ce dit « problème du voyageur de commerce » consiste à trouver la route la plus courte entre un ensemble de villes et ne passer qu'une seule fois par chacune d'entre elles. *Mais comment calculer rapidement un itinéraire pour le voyageur ?*



L'algorithme Brute Force

Cette méthode consiste à calculer toutes les distances pour déterminer quel est l'itinéraire parfait. Cette méthode est efficace quand on doit calculer un petit nombre de ville, mais cela prend de plus en plus de temps, car le nombre de calculs à effectuer est exponentiel.

5 villes : 12 chemins
10 villes : 180 000 chemins
15 villes : 43 milliards de chemins

Les temps de calculs deviennent longs et ne sont plus raisonnables. Est-il possible d'être plus rapide ?

Expérimentations :

Nous avons programmé un algorithme génétique naïf à partir d'un squelette de code.

```
mutate: enfants
    | enfant v1 v2 |
    configuration mutationRate
        timesRepeat: [ enfant := enfants atRandom.
            v1 := configuration numberOfCities atRandom.
            v2 := configuration numberOfCities atRandom.
            enfant swap: v1 with: v2 ].
    ^ enfants
```

Développement d'un Algorithme Génétique naïf, partie mutation.

Traduction linéaire :
- Création de variables « enfant, v1 et v2 »
- Probabilité de mutation (variable définie au préalable)
- sélection aléatoire d'un « enfant » dans le groupe « enfants »
- sélection d'une première ville au hasard (v1)
- Sélection d'une seconde ville au hasard (v2)
- Inversion de la place des deux villes
- Réinsertion de l'« enfant » muté dans son groupe d'origine « enfants ».

Analyse :

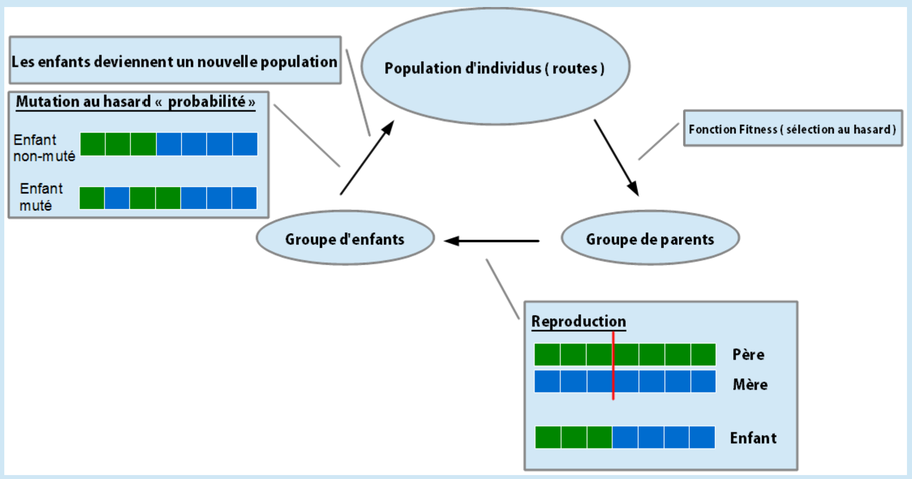
On observe donc que le Brute force est une très bonne solution quand le nombre de routes reste inférieur à 10, mais les Algorithmes Génétiques, même en version naïve, représentent un gain de temps considérable par rapport au Brute Force dès que le nombre de villes augmente.

Conclusion :

Nous avons donc comparé deux algorithmes en version naïve, qui sont le Brute Force et l'Algorithme Génétique. Nous en avons conclu, à la suite de nos expérimentations, que l'algorithme génétique est beaucoup plus rapide pour trouver une bonne solution, mais ne garantit pas la solution parfaite. Malgré tout, il existe d'autres méthodes que le Brute Force, plus efficaces, qu'il serait intéressant de comparer avec notre Algorithme Génétique.

L'Algorithme génétique :

Cet algorithme qui se base sur la sélection naturelle n'a pas pour but de trouver une solution, ou ici une route, parfaite, mais une bonne solution, en ayant un gain de temps non-négligeable quand la Brute force atteint ses limites.



Le principe est assez simple :

1) On sélectionne un nombre donné de « Parents »

2) On fait se reproduire deux par deux les « Parents »

3) Cette reproduction donne un « Enfant », qu'on espère meilleur que ses « Parents »

4) On opère aléatoirement une mutation sur l' « Enfant »

5) On le réintègre dans la population d'enfants

6) On réitère cette opération autant de fois que souhaité

Résultats des expérimentations

Brute Force VS notre Algorithme Génétique

	Brute Force			Algorithme génétique		
	Meilleur chemin	Temps	Nombre de cycles	Meilleur chemin	Temps*	Nombre de cycles
7	1740 Km	36 ms	5 040	1740 Km	4 s 238 ms	1000
8	1815 Km	313 ms	40 320	1815 Km	4 s 138 ms	1000
9	1971 Km	3,273 s	362 880	1971 Km	4 s 150 ms	1000
10	2152 Km	38 s	3 628 800	2152 Km	6 s 532 ms	1000
11	2156 Km	7 m 16 s	39 916 800	2156 Km	2 s 721 ms	1000
12	2183 Km	1 h 29 m	479 001 600	2183 Km	4 s 983 ms	1000
13	2313 Km	21 h 57 m	6 227 020 800	2313 Km	9 s 582 ms	1000

*Temps total pour trouver la solution parfaite (plusieurs essais).

Pour l'Algorithme Génétique, on peut relancer le programme plusieurs fois puisque le temps de calcul pour trouver une bonne solution est très court.