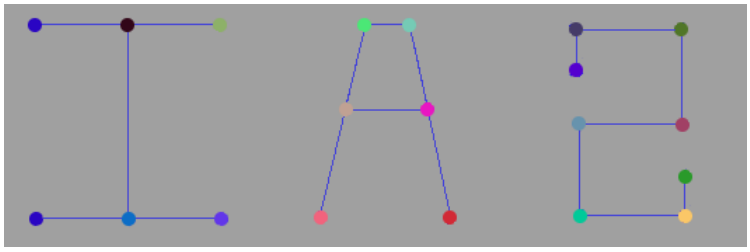
Rapport Intelligence Artificielle

Aragon Nicolas
Taillefer Jordan

Mai 2016



1 Introduction

Dans le cadre du module d'intelligence artificielle 2, nous avons pu découvrir pendant les cours des algorithmes de recherche de chemin le plus court.

Dans ce rapport nous allons voir comment nous avons mis en place de manière pratique l'algorithme génétique et l'algorithme des fourmis. Le but de ce projet est de trouver un chemin hamiltonien dans un graphe complet qui tende à être le plus court tout en gardant une rapidité dans sa recherche. En effet pour trouver le vrai chemin le plus court, on aurait dû tester toutes les chemins possibles, ce qui prendrait énormément de temps.

Afin de réaliser notre projet, nous avons pris l'initiative d'améliorer nos connaissances personnelles en utilisant une nouvelle technologie. Nous avons utilisé une technologie améliorée du Java simple : Java FX. Technologie utilisée en entreprise, nous avons pu rendre notre application plus jolie et plus optimisé comparé à Java Swing (Technologie que nous avons vu durant les cours).

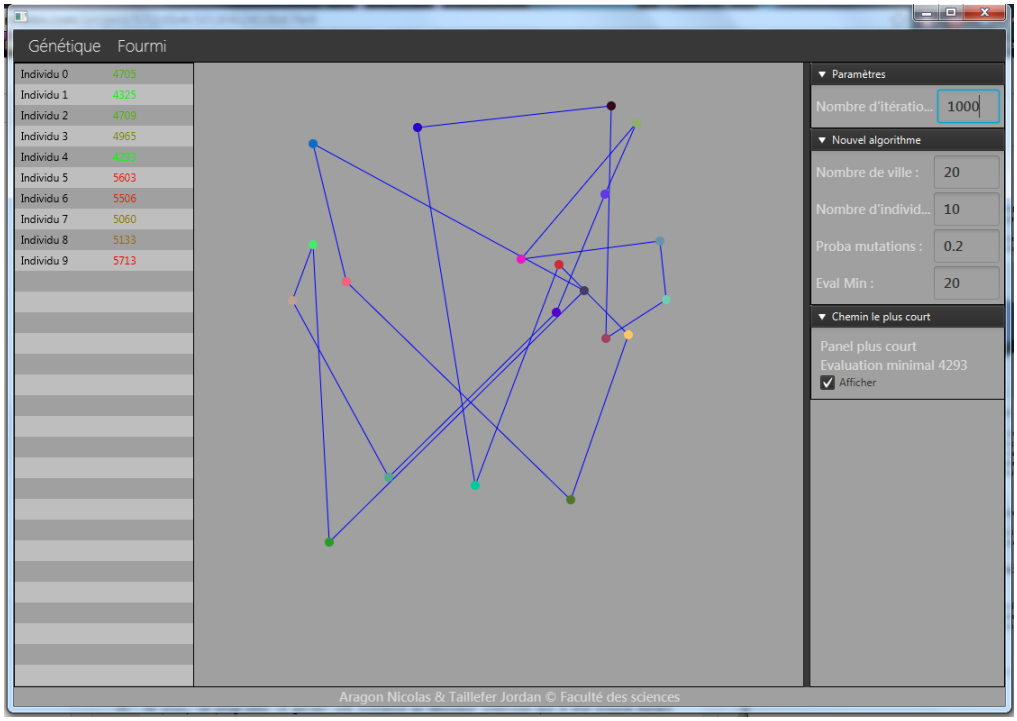


Figure 1: Interface

2 Algorithme génétique

2.1 Principe

Le premier algorithme mis en place est donc l'algorithme génétique.

Ce code à pour but de trouver une solution approché de l'optimal avec un nombre réduit d'itération.

L'algorithme va prendre en compte un graphe complet et une population d'individu(un individu va être un chemin hamiltonien compris dans le graphe complet).

Chaque individu va posséder une évaluation qui va être la base de notre heuristique, cette évaluation est calculer en faisant la somme des distances entre chaque point.

Afin de pouvoir comparer nos 2 algorithmes, nous avons mis en place un système de sauvegarde rapide du graphe afin de pouvoir tester les 2 algorithmes sur le même graphe complet.

Pour les tests individuels nous avons utilisé le graphe suivant

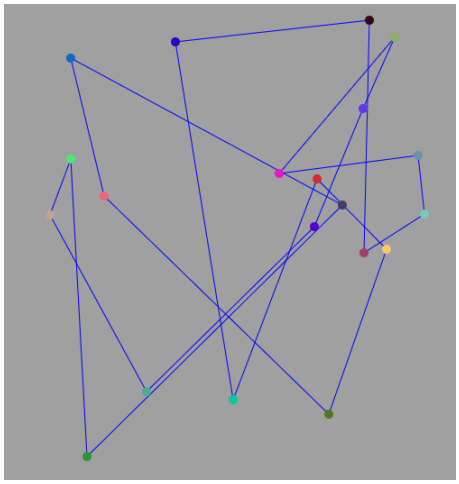


Figure 2: Graphe

2.2 Croisement

Le première étape va être de croiser les individus entre eux. Pour faire cela, on va choisir 2 individus, puis on va croiser certains points de ceux-ci afin de former un nouvel individu. On croiser l'individu 1 avec l'individu 2 puis inversement le 2 avec le 1, ce qui va nous donner 2 nouveaux individus.

2.3 Mutation

Une fois qu'on à obtenu tout nos nouveaux individus, on va parcourir toutes nos populations.

Grâce à un facteur de probabilité (dans notre projet nous sommes partit sur une base de 20% de chance de muter mais un champ permet de régler celle-ci), on va faire muter nos individus, cet à dire qu'en cas de mutation, on va échanger 2 points tirés au hasard du circuit hamiltonien.

2.4 Roulette russe

La dernière étape est de faire revenir notre population à la taille d'origine.

On va donc tirer au sort un individu grâce à un mécanisme de roulette russe, puis le supprimer de notre population.

Pour cela, nous avons récupérer la somme total des évaluations, on obtient donc un pourcentage d'importance de chaque individu.

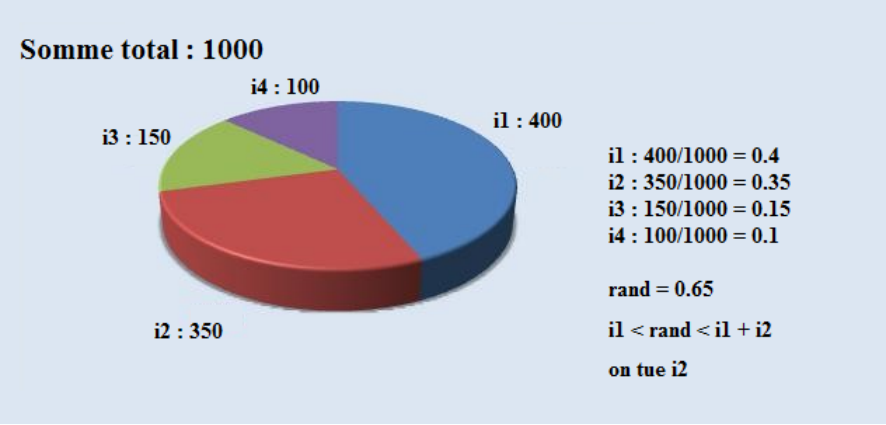


Figure 3: Chart Camembert

On tire ensuite un aléatoire, puis on supprime l'individu qui correspond jusqu'à revenir à la taille d'origine de la population.

Pour nos tests de performance, nous avons choisi d'effectuer 1000 itérations de l'algorithme et de faire varier le nombre d'individu.

10 individus

1	2	3	4	5	6	7	8	9	10	Moyenne
2114	2227	2184	2165	1962	2107	2152	2171	2038	1961	2108.17

25 individus

1	2	3	4	5	6	7	8	9	10	Moyenne
2118	2010	2019	2132	2138	2176	2071	1993	2016	2056	2072.9

50 individus

1	2	3	4	5	6	7	8	9	10	Moyenne
1955	1990	2013	1940	1986	1988	1933	1943	2013	1905	1966.6

100 individus

1	2	3	4	5	6	7	8	9	10	Moyenne
1963	1963	1956	1988	1971	1905	1947	1980	1961	1957	1959.1

Figure 4: Resultat 10 echantillons génétique

On peut voir une amélioration croissante en fonction du nombre d'individu.

2.5 Amélioration

Afin de trouver de meilleurs résultat, nous avons choisi de ne jamais garder 2 fois le même individu.

De plus, le programme va garder une instance du meilleur individu qui à été trouvé durant toutes les itérations que nous avons faites.

3 Algorithme des fourmis

3.1 Principe

Le second algorithme est l'algorithme des fourmis, du fait de son implémentation, le circuit résultant va tendre à être le plus court possible.

Lors de l'initialisation, chaque fourmis va être attribuer à une ville de départ choisi au hasard.

A chaque itération, elle choisit la ville suivante pour son trajet en fonction des phéromones présentes entre la ville où elle se trouve et les villes qui lui restent à visiter pour son trajet (en appliquant la fonction roulette russe)

Lorsque les fourmis ont terminé leur trajet, elles font le chemin en sens inverse en ajoutant une quantité de phéromones inversement proportionnelle à la longueur du circuit trouvé

A chaque fois que les fourmis reviennent à leur point de départ, on atténue toutes les phéromones d'un taux que nous pouvons faire varier pour tester différents paramètres.

Pour nos tests de performance, nous avons choisi de faire varier le nombre de fourmis sur un algorithme d'une durée de 10 sec.

10 Fourmis

1	2	3	4	5	6	7	8	9	10	Moyenne
2290	2309	2063	2263	2307	2326	2460	2036	2340	2221	2262.1

25 Fourmis

1	2	3	4	5	6	7	8	9	10	Moyenne
1947	1946	1972	1999	2229	1983	2042	1887	1777	2082	1986.4

50 Fourmis

1	2	3	4	5	6	7	8	9	10	Moyenne
1907	1966	1886	1927	1820	1959	1881	1772	1948	1908	1897

100 Fourmis

1	2	3	4	5	6	7	8	9	10	Moyenne
1869	1742	1883	1792	1845	1845	1712	1826	1748	1805	1806.7

Figure 5: Resultat 10 echantillons fourmis

Dans ce cas la, on voit clairement une amélioration du résultat sur les échantillons en fonction du nombre de fourmis.

4 Comparaison entre l'algorithme génétique et l'algorithme des fourmis

En reprenant les tests que nous avons vu précédemment, nous avons donc décidé de voir lequel des 2 algorithmes, dans des conditions suffisantes de test, donne le chemin le plus court.

Pour l'algorithme génétique nous avons choisi de prendre comme paramètre 20 villes, 50 individus et 1000 itérations. (Résultat : 1966.6)

Pour l'algorithme des fourmis nous avons choisi de prendre comme paramètre 25 fourmis et une durée de 10 secondes. (Résultat : 1986.4)

Nous avons répété ce test sur 20 graphes différents générés aléatoirement.

	1	2	3	4	5	6	7	8	9	10	Résultat
Génétique	2217	1841	1855	1911	1792	1989	1825	2043	2131	1816	6
Fourmis	2154	1811	1916	2374	1664	2085	1842	2052	2242	1774	4
	11	12	13	14	15	16	17	18	19	20	
Génétique	1674	2000	1822	1936	1944	1748	1900	1820	2253	1941	2
Fourmis	1665	1917	2091	1876	1801	1582	1820	2179	2251	1929	8

Figure 6: Résultat comparaison

On peut voir que le match est resté très serré (8 gagnés pour génétique et 12 gagnés pour les fourmis), mais dans l'ensemble les fourmis sont ressortis vainqueur de ce duel.

5 Conclusion

Pour finir, nous avons pu voir comment mettre en place des algorithmes pour résoudre le problème du voyageur de commerce avec un coup machine réduit.

L'algorithme génétique est clairement meilleur que le génétique lorsqu'on dispose de peu d'élément (10 individu : 2108.7 , 10 fourmis : 2262.7), mais lorsqu'on atteint un grand nombre d'élément, alors les fourmis reprennent fortement leurs places en étant très proche de la solution parfaite (100 Individus : 1959.1 , 100 Fourmis : 1806.7)