TIPE Algorithmes génétiques

TEMPEZ Vladislav

11/06/2014

Table des matières

1 Introduction

1.1 Le problème du voyageur de commerce

Ce problème revient à optimiser la longueur du trajet d'un représentant de commerce qui doit visiter n villes.

Dans le cadre de ce TIPE, les villes seront considérées comme disposées dans un plan, en première approximation ce que l'on trouve sur terre.
Par ailleurs on supposera l'existence d'un "chemin" entre tout couple de villes. La "longueur" du chemin ne dépendra pas du sens de parcours.

Une étude exhaustive est factorielle, c'est la raison pour laquelle on applique la méthode des algorithmes génétiques à ce problème.

1.2 Les algorithmes génétiques

Les algorithmes génétiques s'inspirent de la sélection naturelle.
L'analogie est la suivante : un point de l'ensemble de définition de la fonction est un individu, et l'image de ce point par la fonction est l'adaptation de cet individu à son environnement.
L'algorithme va alors sélectionner, croiser et faire muter les individus pour obtenir un individu bien adapté à son environnement ie un point où la fonction est haute.
On obtient donc un individu bien adapté et non l'individu le mieux adapté. Cet algorithme ne résout donc pas les problèmes d'optimisation au sens strict du terme.

Les trois mécanismes qui permettent à l'algorithme de construire une bonne solution sont la sélection, le croisement et la mutation.

La sélection consiste à choisir certains individus dans la population en fonction de leur adaptation à l'environnement.

Le croisement consiste à prendre deux (ou plus) individus de la population afin d'en construire un troisième, on tente de rassembler dans ce troisième individu les caractéristiques des ses parents.

La mutation consiste à prendre un individu et à modifier un petit peu ses caractéristiques pour obtenir un individu différent mais proche du premier. Cette mutation permet à la population d'acquérir des bonnes caractéristiques non présentes initialement.

Plus concrètement, on peut coder un individu par une chaîne de caractères qui codent les caractéristiques de l'individu, un croisement entre deux individus peut consister à prendre le début de la chaîne codant le premier et à l'accoler à la fin de la chaîne codant le second.

exemple de croisement : 39086 et 36281 donnent 39281

Une mutation peut consister à changer aléatoirement un caractère de la chaîne codant l'individu muté.

exemple de mutation : 39086 donne 39586

Plus précisément, le fonctionnement générique de l'algorithme est le suivant :
- Génération aléatoire d'un ensemble de n individus : la population.
- Croisement des individus entre eux
- Ajout des individus issus de croisement à la population
- Mutation des individus de la population
- Sélection de n individus dans la population
- Application de cet algorithme sur la population formée par les n individus sélectionnés à l'étape précédente.

La condition d'arrêt peut être un nombre d'itérations fixé de l'algorithme, où dès qu'un individu assez bon est dans la population. Ces algorithmes permettent donc l'exploration non exhaustive d'un espace d'individus par un déplacement de proche en proche (les mutations) et un transfert des bonnes caractéristiques qui s'effectue par le croisement la sélection (pour déterminer quelles sont les bonnes caractéristiques).

2 Implémentation

2.1 Codage des solutions

Dans le cas du problème du voyageur de commerce, les individus sont des trajets dans un ensemble de villes. Une liste des villes parcourues dans l'ordre de parcours semble être un codage assez intuitif d'un trajet.
On pourrait aussi considérer un codage dans lequel les numéros des villes sont convertis en binaire, puis sont concaténés pour former une chaîne de caractères comme évoqué dans le cas général.
Cette dernière représentation a l'inconvénient de donner des portions de code qui ne correspondent à aucune ville. Par exemple pour $n = 3$ villes, on doit coder chaque numéro de ville par deux bits.

numéro de la ville	code de la ville
1	00
2	01
3	10

Ce qui laisse la combinaison 11 inutilisée.
Le risque est alors que cette portion de code apparaisse lors d'un croisement ou d'une mutation. Ce TIPE n'utilisera donc que le premier codage des solutions.

La seule information qui nous intéresse dans la distribution des villes est donc la distance entre deux villes quelconques de cette distribution. Cette information va être rassemblée dans une matrice dite "d'adjacence". Le coefficient (i, j) de cette matrice sera la distance entre la ville i et la ville j.

2.2 Mutation

On souhaite une fonction de mutation qui génère un trajet assez "proche" du trajet passé en argument. Le cas général présenté dans la première section ne fonctionne pas ici, en effet, si on modifie la valeur d'une case du tableau représentant un trajet, on va obtenir un nouveau trajet qui ne passera pas par toutes les villes et qui passera deux fois par une ville.

Nous verrons trois opérations pour conserver un trajet valable.
La première consiste à échanger deux villes dans le tableau qui représente le trajet. On la note mutation naïve.
La seconde est d'inverser le sens d'une fraction du trajet. On la note mutation inversion.
On peut aussi translater une partie du trajet en décalant le reste. C'est la mutation déplacement.
Nous verrons dans la section suivante que ces trois mutations ne donnent pas les mêmes résultats. On peut déjà remarquer que la seconde semble donner un trajet un peu plus proche du trajet passé en argument.

2.3 Croisement

De la même manière que pour les mutations, l'exemple générique du croisement présenté dans la section précédente ne fonctionne pas ici.
Si la première partie du parent 1 contient la ville i, ville i aussi présente dans la seconde partie du parent 2. On obtiendrait un trajet qui contient deux fois la ville i et donc ne contenant pas une ville que le voyageur de commerce est censé visiter.

On peut effectuer une variation du croisement générique : on prend toujours la première partie du parent 1, et on complète ensuite par les villes qui ne sont pas dans cette première partie, mais dans l'ordre dans lequel elles apparaissent dans le parent 2. De cette manière on conserve toute une partie du trajet du parent 1 et des sections du trajet du parent 2.
On peut garder non pas seulement le début du parent 1, mais le début et la fin, et n'effectuer le remplacement qu'entre deux points du trajet. On appelle cette variation un croisement bipoint.

On constate que ce n'est pas l'ordre dans lequel apparaissent les villes qui importe, mais leur position relative dans le trajet.
Les voisins d'une ville lors d'un trajet sont les données qui importent. On va donc tenter de préserver un maximum ces relations de voisinage lors du croisement.
Pour cela on va écrire une fonction de croisement heuristique qui va construire un trajet qui tente de réutiliser au maximum les relations de voisinage que nous allons appeler arêtes par la suite. Plus précisément on dira qu'un trajet contient l'arête (i, j) si i est visitée juste après j ou j visitée juste après i.
Le procédé est le suivant :

1 On construit la liste l_1 des arêtes présentes dans les trajets des deux parents.
2 On détermine la ville qui est présente dans le moins d'arêtes.
3 Cette ville devient la ville courante v_c
4 On retire les arêtes contenant v_c de la liste l_1 et on en fait une liste l_2 contenant v_c.
5 On place v_c à la fin du trajet.
6 Si la liste l_2 est vide on prend une ville non visitée au hasard, celle ville devient la ville courante v_c et on revient au point 4
7 Si l_2 n'est pas vide, on choisit l'arête telle que la deuxième ville soit présente dans le moins d'arêtes de l_1. Cette ville devient la ville courante et on revient au point 4
8 On s'arrête quand toutes les villes ont été visitées.

2.4 Sélection

La première méthode de sélection est celle qui consiste à classer les trajets par longueur croissante et à ne retenir que les p plus courts.

Cette sélection par classement assure que le meilleur individu de la population sera toujours conservé, mais

nécessite un tri dont le coût est plus que linéaire. Il y a aussi le risque que l'algorithme se retrouve bloqué sur un extremum local de la fonction, risque renforcé par cette sélection déterministe.

Une méthode moins déterministe consiste à sélectionner deux individus dans la population et à ne garder que le meilleur des deux. Répéter ceci p fois pour sélectionner p individus. Cette sélection fait intervenir fortement le hasard. Il y a donc un risque que les meilleurs individus ne soient pas assez favorisés. Il s'agit de la sélection par tournoi.

On a aussi la sélection par roulette. Il s'agit d'une sélection probabiliste. La probabilité de sélectionner un individu est proportionnelle à la qualité de l'individu.
La fonction qui détermine la probabilité de sélection est la suivante $f(i) = \frac{L_{max}-L_i}{L_{max}-L_{min}+1}$. La probabilité de sélection de l'individu i est alors $\frac{f_i}{\sum_i}$. Cette fonction assure une probabilité maximale pour le meilleur individu et une probabilité nulle pour le pire. Le $+1$ intervient pour que la fonction soit toujours définie en cas de population homogène.
Une transformation affine $f'_i = af_i + b$ de cette fonction permet d'obtenir l'importance désirée pour le meilleur individu en conservant la même valeur moyenne.

3 Résultats

3.1 Comportement sur une distribution "simple"

Dans une distribution circulaire des villes, le problème est très simplifié puisque le trajet le plus court est un cercle. L'algorithme converge effectivement vers ce type de trajet.

3.2 Complexités

3.2.1 Sélection

La sélection tournoi a un coût constant.
La sélection classement a un coût quadratique (celui du tri).
La sélection roulette a un coût constant nécessitant un prétraitement linéaire. De même pour la roulette avec transformation affine.

3.2.2 Croisement

Une première version du croisement 1 consistait à insérer les éléments non présents dans la première partie, on avait donc une complexité quadratique (recherche linéaire pour chaque élément à ajouter en fin). Sa seconde version fait appel à un tableau qui recense les villes déjà visitées, permettant ainsi une complexité linéaire au prix d'un peu de mémoire. Sa variation bipoint a un coût similaire
Le second croisement est plus complexe, la première fonction qui liste les arêtes des deux trajets parents est quadratique. La fonction qui détermine la ville ayant le moins d'arêtes non utilisées est linéaire. Celles d'ajout et de retrait sont aussi linéaires. Le second croisement est donc quadratique.

3.2.3 Mutation

La mutation naïve a coût un linéaire (copie du trajet initial).
La mutation inversion a un coût linéaire en la taille du trajet (on recopie le contenu du tableau).
La mutation déplacement a un coût linéaire.

3.3 Comparaison à d'autres algorithmes

Une camarade a résolu ce même problème à l'aide d'un parcours en profondeur d'un arbre recouvrant minimal du graphe. Le temps de calcul est bien moindre (moins de 1s) et les résultats sont très bons (longueur entre 15 et 20).
Un algorithme de plus proche voisin donne des résultats rapides (0,04) mais globalement moins bon (longueur *simeq* 85).

3.4 Influence de la taille de la population

Pour une population trop importante on n'observe aucune convergence de l'algorithme (cf annexe figure 1). On peut supposer que la sélection ne favorise pas suffisamment les meilleurs individus pour obtenir une convergence.
Par ailleurs, l'algorithme a un coût plus que linéaire en la taille de la population (nombre linéaire d'appel à la

fonction de croisement et à la fonction de mutation, donc coût quadratique voire cubique). Une faible population est donc plus intéressante pour cet algorithme.

3.5 Efficacité des Mutations

Les trois techniques de mutation entraînent un coût similaire (comme attendu). Leur efficacité est cependant très variable, et en particulier, la mutation naïve est inefficace, et la mutation inversion se révèle la plus efficace. (cf annexe figure 2).

3.6 Efficacité des Croisements

L'efficacité des trois mécanismes présentés semble être la même, puisque les résultats obtenus sont similaires pour les trois croisements. Cependant, le croisement heuristique a un coût bien supérieur aux deux autres (près du triple sur cet exemple). (cf annexe figure 3).

3.7 Efficacité des Sélections

Les trois techniques de sélection ont des coûts différents, la sélection par tournoi, qui est la moins coûteuse, se révèle incapable de faire converger l'algorithme. La sélection par classement est un peu plus efficace que la sélection par roulette, mais la différence de coût est assez importante. (cf annexe figure 4). La technique de transformation affine de la fonction de fitness semble ne pas être efficace dans ce cas précis.(cf annexe figure 5).

4 Conclusion

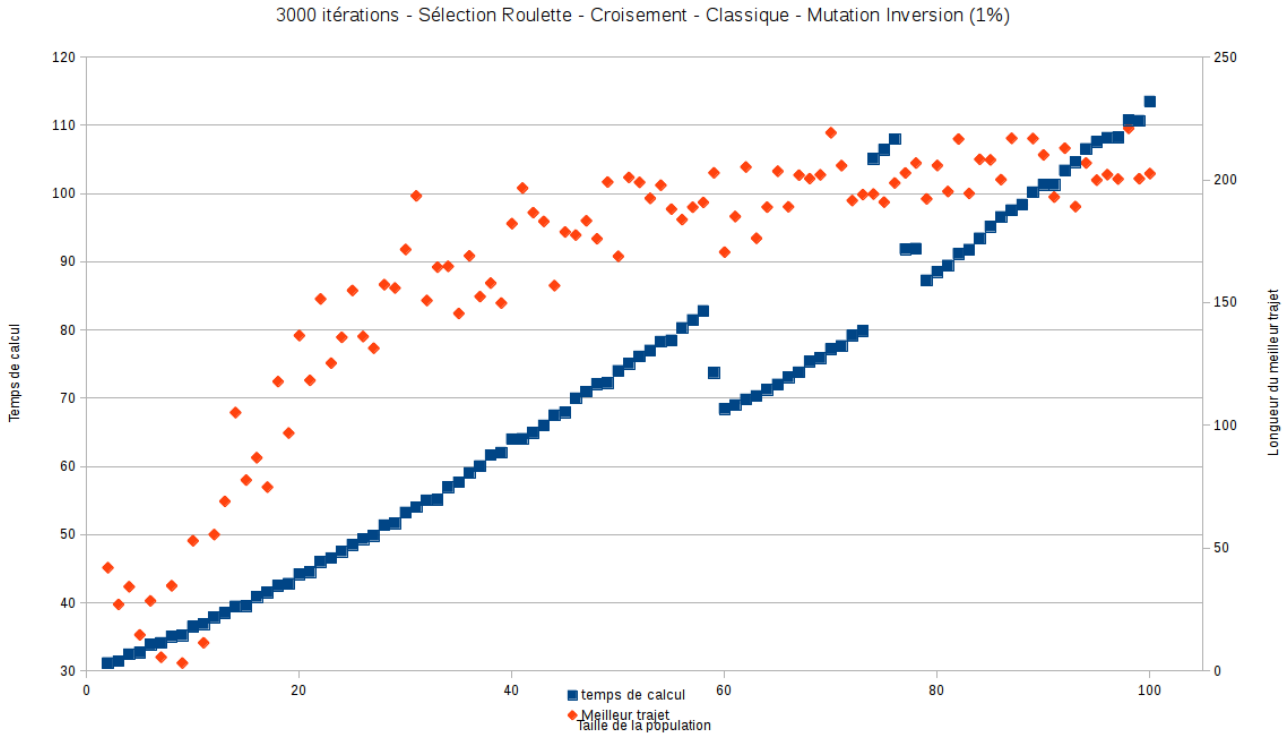
Le choix des différentes techniques de croisement, sélection et mutation apparaît donc crucial, puisqu'il conditionne l'efficacité de cet algorithme.
Par ailleurs, dans le cas présent, les algorithmes génétiques se révèlent plutôt efficaces, même si leur temps d'exécution est assez important. On remarquera toutefois que la population est très souvent homogène au cours des itération de l'algorithme, ce qui pourrait signifier que celle-ci n'est pas renouvelée par les croisements, et que les enfants ne sont pas sélectionnés.

5 Bibliographie

- Genetic algorithms for the travelling salesman problem : A review of representations and operators, P.Larrañaga, C.M.H.Kuijpers, R.H.Murga, I.Inza, et S.Dizdarevich, Artificial Intelligence Review, 13, 129-170, 1990.
- Artificial Intelligence : A Modern Approach, Stuart J. Russell, Peter Norvig, Pearson, Décembre 2010.
- Vie artificielle, Où la biologie rencontre l'informatique, P.Rennard, Vuibert, 2002.
- On Solving Travelling Salesman Problems by Genetic Algorithms, Heinrich Braun, volume 496 of Lecture Notes in Computer Science, page 129-133, Springer, 1990.
- Genetic Algorithms and the Traveling Salesman Problem, Bryant, Kylie, 2000.

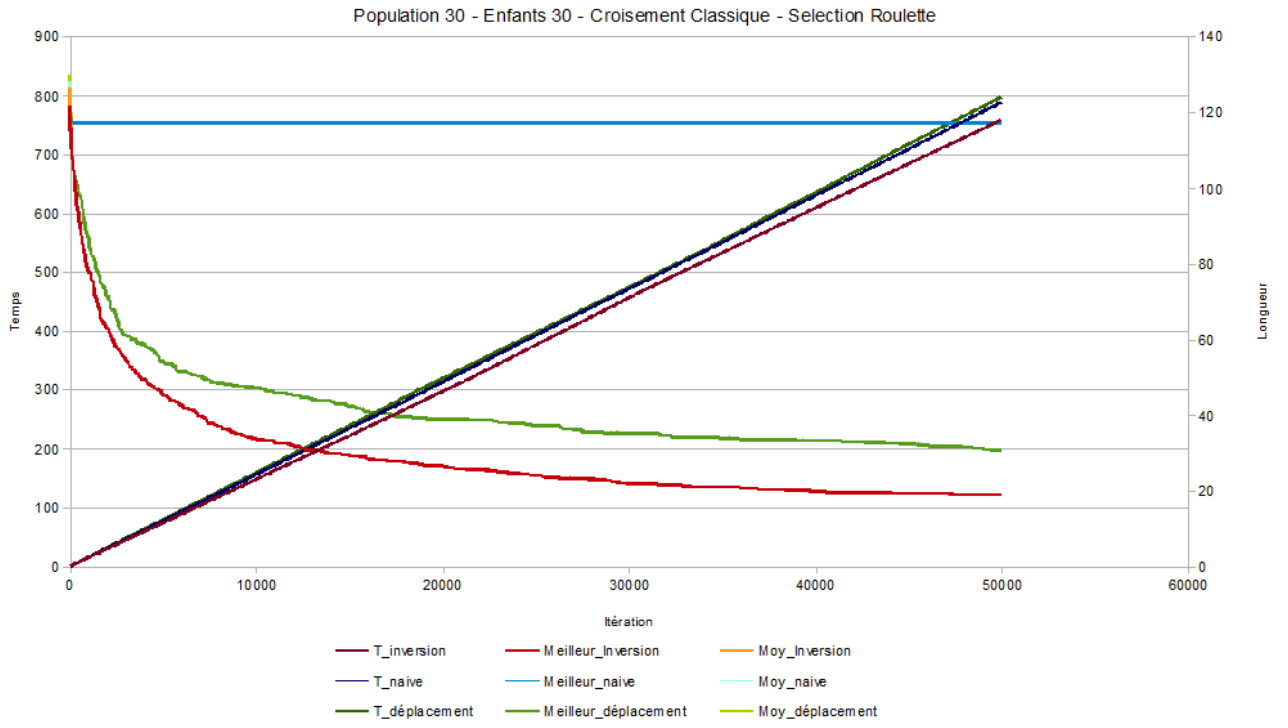
6 Annexe

6.1 figure 1



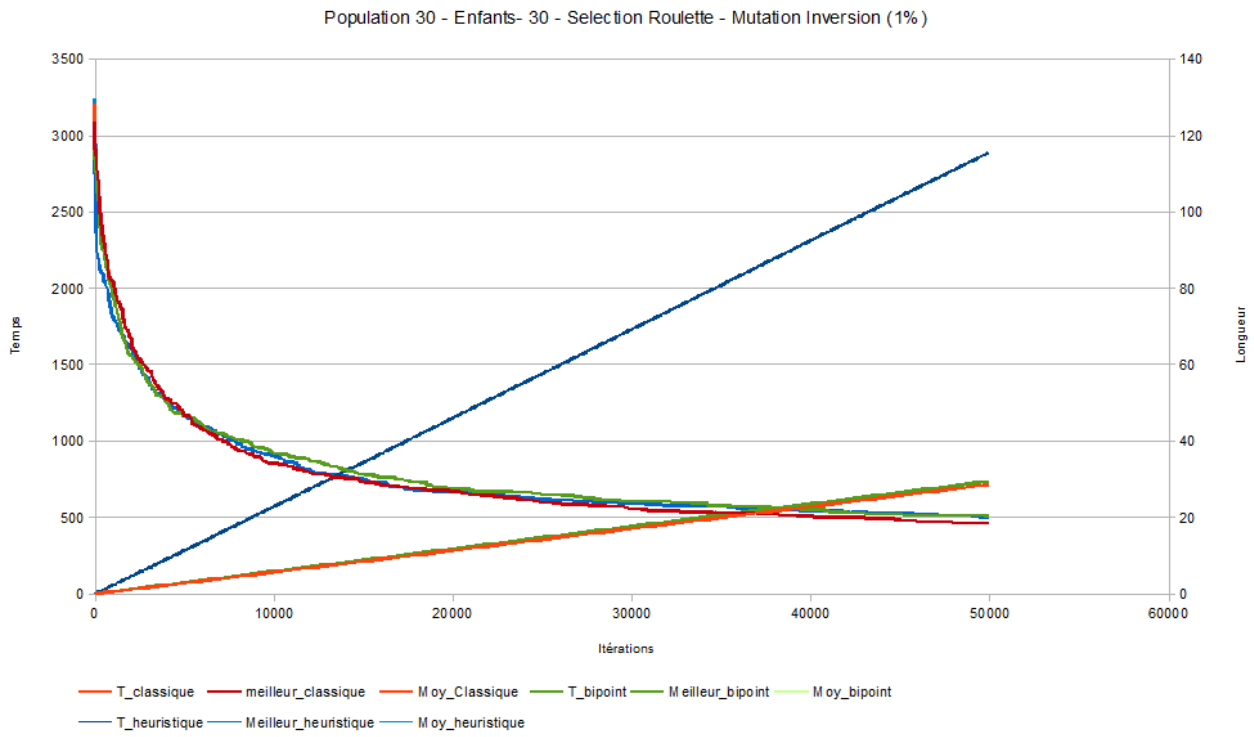
Évolution des résultats de l'algorithme en fonction de la population.

6.2 figure 2



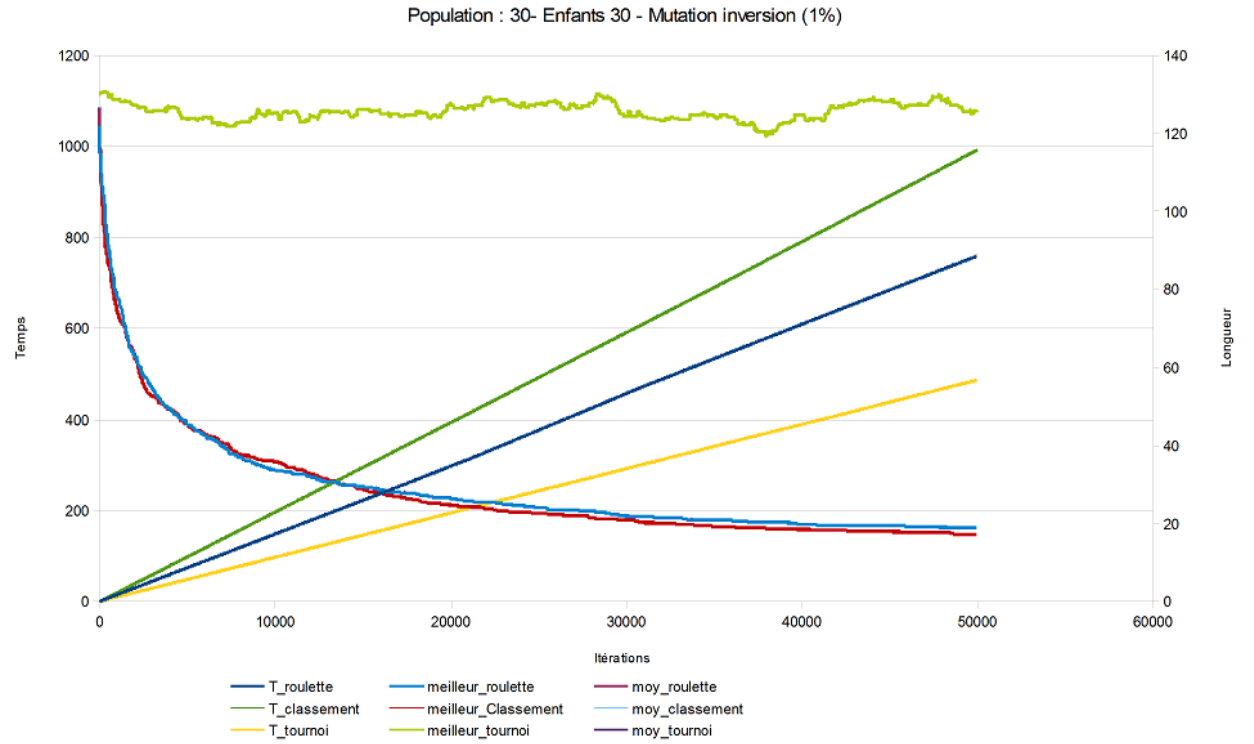
Comparaison des techniques de mutation.

6.3 figure 3



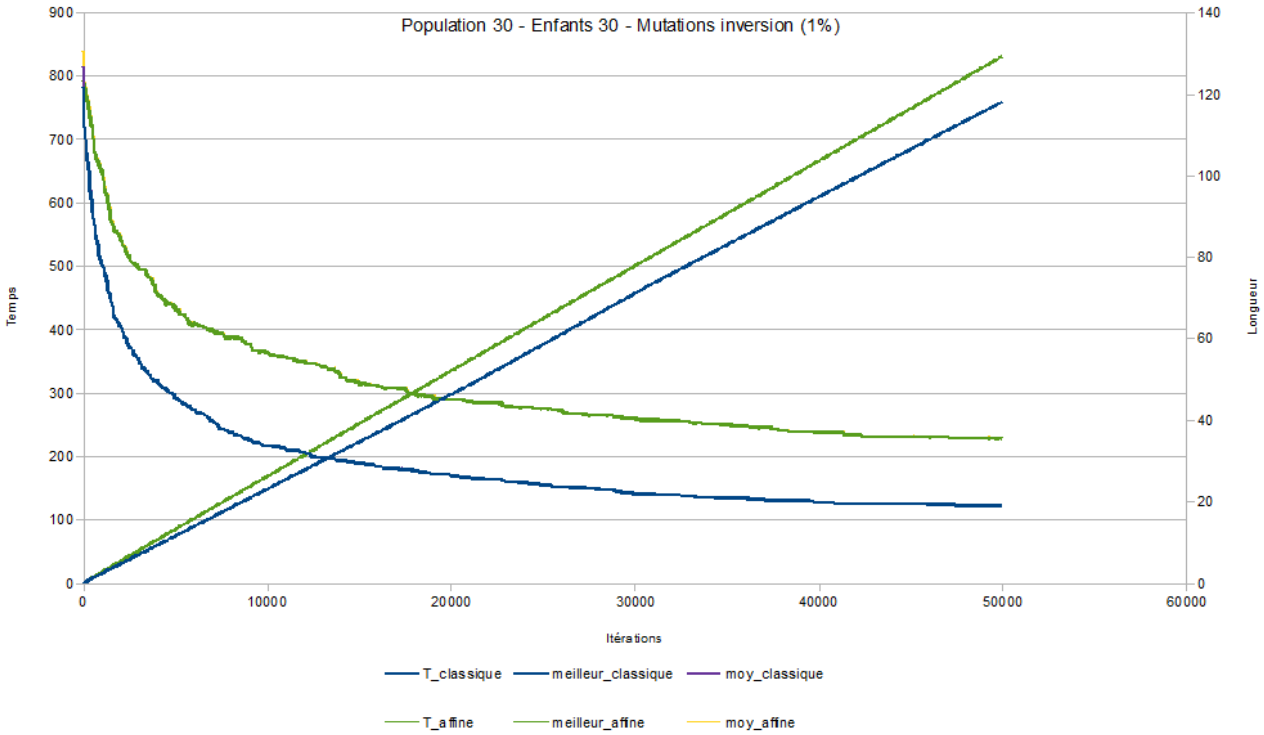
Comparaison des techniques de croisement.

6.4 figure 4



Comparaison des techniques de sélection.

6.5 figure 5



Efficacité de la transformation affine.