**I –Le problème du voyageur de commerce**

1. **Introduction**

Le problème du voyageur de commerce a été posé pour la première fois sous la forme d’un jeu par William Rowan Hamilton en 1859. Il le formalise de la manière suivante : « Un voyageur de commerce doit faire sa tournée en visitant une seule fois un nombre fini de villes avant de revenir à son point d’origine. »

Ce problème d’optimisation appartient à la classe des problèmes NP-Complets. Cela signifie qu’on ne connaît pas d’algorithme trouvant la solution en temps polynomial, même si l’on peut vérifier toute solution rapidement. Ainsi les algorithmes permettant de trouver la solution exacte au problème ont un coût temporel très élevé et on leur préfère les algorithmes d’optimisation, plus rapides, qui approchent la solution.

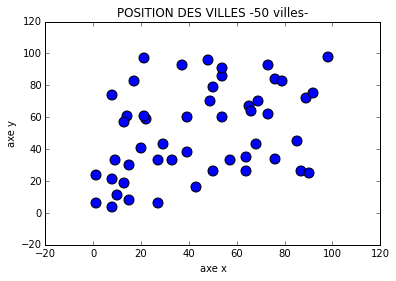
Le problème du voyageur de commerce peut être modélisé à l’aide d’un graphe constitué d’un ensemble de sommets et d’un ensemble d’arêtes. Chaque sommet représente une ville, une arête symbolise le passage d’une ville à une autre, et on lui associe un poids qui représente la distance entre celles-ci.

Résoudre le problème du voyageur de commerce revient à trouver dans ce graphe un cycle passant par tous les sommets une unique fois (un tel cycle est dit hamiltonien) et qui soit de longueur minimale.

**2 – Méthodes de résolution**

Pour chaque méthode, on considère le problème du voyageur de commerce pour 50 villes placées aléatoirement de manière uniforme dans un carré de côté 100 km. Les graphes retournés ne relient pas la première ville du chemin (correspondant au numéro 0) avec la dernière. Cela permet de mieux visualiser le chemin. Néanmoins, la distance totale renvoyée comprend toutes les distances entre chaque ville.

Dans le code, la position des villes est stockée la liste *P*. Les distances entre chaque ville sont stockées dans le tableau *villes*.



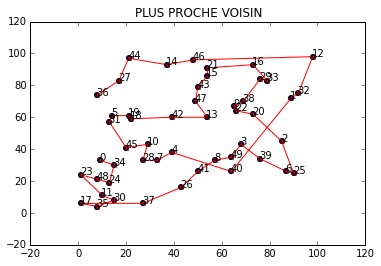
**Méthode combinatoire (exacte)** : La manière la plus intuitive de résoudre ce problème est de considérer tous les chemins possibles et de calculer pour chacun la longueur totale du parcours. Il ne reste plus qu’à déterminer la distance minimale parmi toutes celles calculées. Cependant, pour un ensemble de n points, il existe (n-1)!/2 (ville de départ fixée puis chemins symétriques) chemins possibles différents, l’algorithme n’est donc pas résoluble en un temps raisonnable, d’où la nécessité de trouver de nouvelles méthodes.

Le graphe de l’algorithme n’est pas représenté dans ce cas de figure car pour 50 villes, le coût temporel est très élevé.

**Algorithme du plus proche voisin :**

On impose à l’algorithme la ville de départ. Dans nos codes, tous les chemins débutent par la ville numéro 0. Puis on sélectionne la prochaine ville telle que le poids entre la ville courante et la prochaine ville soit minimal. On réitère cette opération jusqu’à avoir visité toutes les villes.

Cette méthode est peu précise, et possède une complexité O(n^2).



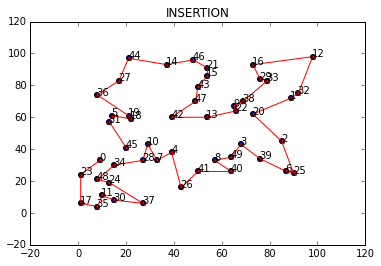
La longueur totale du chemin est : 670.037025 km.

**Méthode par insertion:**

Le parcours du voyageur de commerce est construit pas à pas en y insérant de nouvelles villes. On introduit une nouvelle ville de manière optimale, c'est-à-dire qu’elle augmente au minimum la longueur totale du cycle. À l’étape initiale de l’algorithme, le parcours de voyageur est composé de deux villes, la ville de départ, 0 et celle qui en est la plus proche. L’algorithme se termine lorsque toutes les villes à visiter ont été insérées.

Cependant, même si l’insertion d’une ville dans le cycle est optimale et rapide à calculer, la solution finale n’est pas nécessairement optimale, mais elle est obtenue rapidement (heuristique).

Cette méthode est assez précise, et l’algorithme a une complexité en O(n^3).



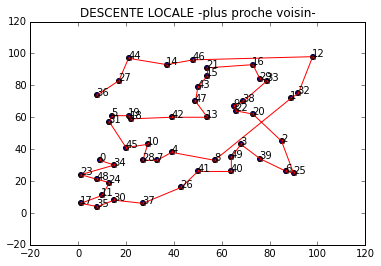
La longueur totale du chemin est : 585.104129 km.

**Méthode de la descente locale:** Après la construction d’une solution initiale par l’utilisation d’un algorithme glouton, un opérateur de voisinage est appliqué à cette solution pour obtenir plusieurs nouvelles solutions voisines. Cet opérateur de voisinage consiste à modifier une solution selon une règle bien définie. Parmi cet ensemble de nouvelles solutions, l’algorithme va choisir la meilleure, celle qui a le cycle de plus petite longueur. Enfin, soit cette solution sélectionnée est meilleure que la précédente, et l’algorithme recommence à appliquer l’opérateur de voisinage sur cette nouvelle solution sélectionnée, soit la solution est moins bonne que la précédente et l’algorithme se termine.

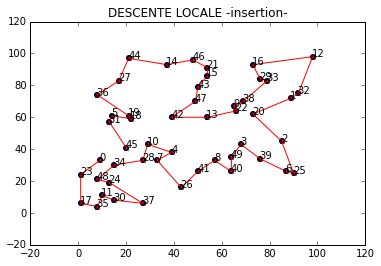
Il existe de nombreux opérateurs de voisinage d’efficacités variées. Un exemple simple est l’échange de deux villes dans le cycle. Pour obtenir l’ensemble de nouvelles solutions à partir d’une solution, il suffit de tester l’échange de chaque ville avec toutes les autres dans le cycle (= permutation)

A partir d’un cycle déjà réalisé par un autre algorithme, un opérateur local modifie l’ordre du cycle. L’algorithme intervertit deux villes au hasard en échangeant leur position. Le nouveau chemin ainsi établi est conservé uniquement si sa longueur totale est inférieure au chemin précédent. Il est ensuite de nouveau modifié. Cette opération est réitérée jusqu’à l’obtention d’une solution précise.

Plus le chemin initialement considéré est précis, meilleure sera la solution renvoyée.



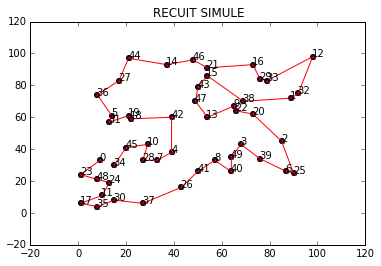
La longueur totale du chemin est : 650.8892436 km.



La longueur totale du chemin est : 580.495927 km.

**La méthode du recuit simulé**

Le recuit simulé est un algorithme itératif considérant les minimums locaux. Il échange aléatoirement les positions des villes entre elles à partir d’une solution initiale. Si le chemin obtenu est meilleur que l’ancien, on le garde et on recommence. Sinon, s’il est moins bon que celui actuel, il peut être accepté ou non selon une probabilité dépendant de la loi de Boltzmann.

Graphique représentant le chemin renvoyé par la méthode du recuit simulé.

La longueur totale du chemin est : 571.883794 km.

**II-Initiative personnelle : les algorithmes génétiques**

1. Introduction aux algorithmes génétiques

Les algorithmes génétiques sont des algorithmes **itératifs** (basés sur la répétition) et **stochastiques** (en référence au hasard).

Ils reposent sur les principes du néodarwinisme : la **sélection naturelle** et **la recombinaison génétique**. D’une part, la sélection naturelle, selon Charles Darwin, énonce que les individus les mieux adaptés sont plus aptes à survivre et à devenir parents de la prochaine génération. D’autre part, Gregor Mendel, père de la génétique, explique que des mutations au niveau des gènes permettent l’évolution des espèces. Ainsi, les mutations les plus adaptées à l’environnement sont les plus transmises et grâce à cet héritage génétique, les espèces actuelles deviennent des “versions optimisées” de leurs ancêtres.

Les phénomènes biologiques ont été une grande source d’inspiration pour les informaticiens. Le parfait exemple est donné par les algorithmes génétiques. Ceux-ci ont été initiés dans les années 1970 par le psychologue et scientifique John Holland. Ils imitent au sein d’un programme les mécanismes d’évolution dans la nature : croisement, mutation, sélection. En effet, dans un algorithme génétique, on part d’une population de solutions potentielles au problème, initialement choisies aléatoirement. Les solutions sont ensuite évaluées grâce à la « fonction d’adaptation » ou *fitness* qui leur assigne une valeur, afin de déterminer leur capacité de survie et de désigner celles qui transmettront leurs gènes à la génération suivante. Les solutions changent donc constamment, les plus adaptées survivent et se reproduisent, les autres disparaissent. On recommence ce cycle jusqu’à obtenir une solution satisfaisante au problème.

C’est en assurant le brassage génétique, par mutation spontanée et croisement sélectif, qu’on peut s’approcher de la solution optimale.

On cherche à déterminer le plus court chemin. On utilise pour cela les algorithmes génétiques. A partir d’un ensemble de solutions initiales, la population, on applique les principes de la théorie de l’évolution : la sélection naturelle et la recombinaison génétique. On sélectionne d’abord les meilleurs individus (meilleures solutions dans la population). Pour cela on emploie la fonction d’adaptation : ici, elle représente la distance totale parcourue par le voyageur. Les solutions, ou individus, sont représentés en tant que liste des villes par lesquelles passent le voyageur. La population est alors une liste de liste (ensemble d’individus). Après sélection, les individus sont croisés de manière à former un individu fils, qui peut subir des mutations et est ensuite réintroduit au sein de la population. Au cours des générations créées, de nouveaux individus apparaissent et remplacent les moins bons individus. Ainsi la qualité des individus s’améliore (la fonction d’adaptation est minimisée) et des chemins de plus en plus courts sont déterminés. La force des algorithmes génétiques est qu’ils font appel au hasard : tous les individus peuvent être modifiés au cours du temps et être sélectionnés.

GENERATION DE LA POPULATION INITIALE

Pour générer la population initiale, on peut soit créer des chemins totalement aléatoires soit prendre des solutions déjà optimales générées par des algorithmes de résolution classiques tels que le plus proche voisin ou la méthode de l’insertion.

SELECTION DES MEILLEURS INDIVIDUS

Comme dans la nature où seuls les meilleurs individus sont aptes à survivre et à se reproduire, on sélectionne deux meilleurs individus, les parents à l’aide d’algorithmes spécialisés : Sélection par roulette, sélection par rang, sélection élitiste, sélection par tournoi.

Intérêt de la sélection : elle permet de sélectionner les meilleurs individus tout en laissant une part au hasard (il est possible que de moins bons individus soient sélectionnées) et donc à la variabilité génétique.

CROISEMENT/HYBRIDATION DES PARENTS

Un individu fils est généré à partir de deux individus sélectionnés. Pour cela, on croise les parents : croisement simple (un point de jonction), croisement double (deux points de jonctions). On obtient ainsi un nouvel individu, qui a des chances d’être une bonne solution puisque issu de solutions optimales.

MUTATION DES INDIVIDUS

Pour éviter la convergence prématurée de l’algorithme vers un minimum local (solution non optimale), il est nécessaire d’introduire de la variabilité génétique, c’est-à-dire une diversité dans les solutions. Pour cela, on échange certaines villes dans les solutions (mutation par substitution) ou on peut inverser le parcours sur une portion de chemin (mutation par inversion de parcours). D’autres types de mutation existent, qui ont pour but d’augmenter la diversité tout en gardant les solutions optimales (certaines mutations peuvent au contraire créer des instabilités et empêcher l’algorithme de trouver des solutions optimales). Ces mutations peuvent être appliquées à l’ensemble des individus ou seulement au fils.

REINSERTION DU FILS DANS LA POPULATION

On supprime le pire individu (le plus long chemin), et on le remplace par le fils nouvellement obtenu. La population possède ainsi une taille constante au cours du temps.

**Etapes de l’algorithme génétique** :

1)Génération de la population initiale

2)Sélection des meilleurs individus

3)Croisement des parents

4)Mutation appliquée au fils

5)Remplacement du fils dans la population

**Vocabulaire** :

* *Individu* : solution du problème (liste de villes par lesquelles passent le voyageur)
* *Population* : ensemble d’individus (liste de liste)
* *Génération* : population à un moment donné du processus
* *Fonction d’adaptation* : mesure de la qualité de l’individu (distance totale parcourue par le voyageur
* *Opérateurs de reproduction* : croisement et mutation