TME2 : Algorithmes génétiques et Recherche locale

Exercice 1 – Mise en œuvre d'un algorithme génétique

Dans cet exercice, vous allez mettre en œuvre un algorithme génétique pour la résolution du problème du voyageur de commerce.
Un voyageur de commerce doit visiter n villes $1, 2, \ldots, n$ situées sur un plan et revenir à son point de départ. Le coût du trajet entre les villes i et j correspond à la distance euclidienne entre ces deux villes. Dans cet exercice, on cherche à déterminer la tournée minimisant le coût total du voyage (*circuit hamiltonien* dans le graphe correspondant). Un exemple de tournée optimale pour un problème à 100 villes vous est donné à la Figure 1.

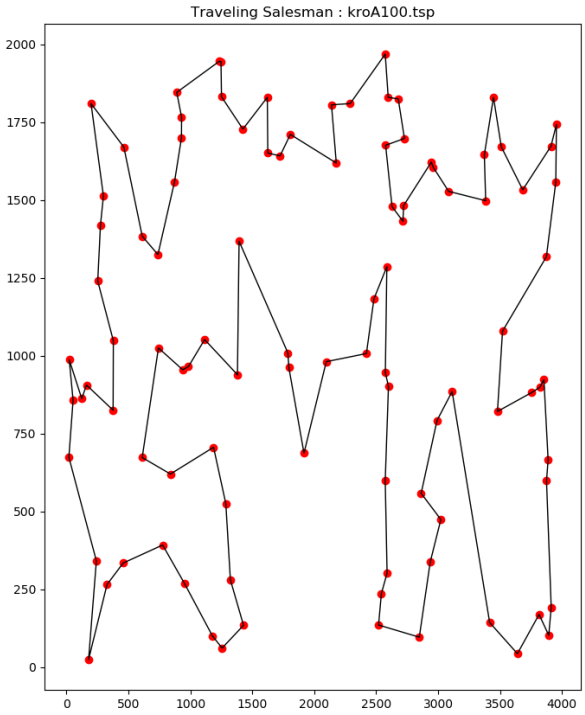

FIGURE 1 – Exemple de tournée optimale

Dans un premier temps, vous allez coder l'algorithme génétique telle que vu au cours, dont le pseudo-code est rappelé ci-dessous.

Algorithm 1 `Algorithme génétique`

Parameter $\downarrow$: N (taille de la population), NbG (nombre de générations), p_m (probabilité de mutation)
Parameter $\uparrow$: Individu de fitness maximum
Générer une population initiale P de N individus
for $k = 1$ to NbG **do**
  for $i = 1$ to N **do**
    f_i =*Evaluation*(Individu(i))
  $P' \leftarrow \{\}$
  for $i = 1$ to $\lfloor N/2 \rfloor$ **do**
    Selection : Choisir deux parents dans P en utilisant les évaluations f_i
    Croisement : Croiser les deux parents pour obtenir deux nouveaux individus
    Mutation : Avec une probabilité p_m, modifier les nouveaux individus
    Insertion : Insérer les nouveaux individus dans P'
  $P \leftarrow P'$
Retourner l'individu de P de fitness maximum

Vous devez ainsi définir :
- Le codage d'une solution
- Une fonction d'évaluation ("fitness")
- Un opérateur de sélection des parents
- Un opérateur de croisement
- Un opérateur de mutation

Pour le codage de cet algorithme, vous pouvez vous aider du fichier `TSP.py` disponible sur Moodle. Ce fichier contient des fonctions permettant de lire des instances au format .tsp et d'afficher les tournées obtenues.

Question 1 — En vue de mettre en œuvre un algorithme génétique, plusieurs codages d'une tournée sont envisageables, parmi lesquels :

• *le codage direct*, correspondant à la donnée directe de la permutation des villes parcourues dans la tournée ;

• *le codage ordinal*, dans lequel la tournée (1, 2, 4, 3, 8, 5, 9, 6, 7) est codée sous la forme de la liste (1, 1, 2, 1, 4, 1, 3, 1, 1), où chaque numéro indique la position de la ville suivante dans la liste des villes non encore visitées ;

Par ailleurs, on envisage deux opérateurs de croisement possibles :

• le *croisement à un point* entre deux tournées x et y, consistant à choisir aléatoirement une position π dans la liste puis à définir le i^{eme} élément du fils z comme étant $z_i = x_i$ pour $i \leq \pi$ et $z_i = y_i$ pour $i > \pi$,

• le *croisement à un point fondé sur le rang*, dans lequel les éléments du fils figurant à droite du point de croisement p correspondent à la liste ordonnée des éléments du second parent ne figurant pas déjà dans la partie gauche. Par exemple, (1, 2, 4, 3, 8, 5, 9, 6, 7) et (3, 1, 7, 4, 5, 6, 2, 9, 8) ont pour fils (1, 2, 4, 3, 8, 7, 5, 6, 9) pour $p = 5$.

Parmi les associations possibles d'un codage et d'un opérateur de croisement, indiquer lesquelles ne peuvent pas convenir et pourquoi.

On considère par la suite un codage direct des tournées.

Question 2 — Proposer un opérateur de mutation transformant une solution réalisable en une autre solution réalisable proche.
Question 3 — Proposer un opérateur de sélection permettant de définir les parents à croiser.
Question 4 — Implémenter ensuite l'algorithme génétique et tester ses performances pour les instances données sur Moodle : vous représenterez l'évolution de la qualité de la meilleure solution de la population en fonction du nombre de générations.
Voici les évaluations optimales pour les instances "Kro" :

KroA100	21282
KroB100	22141
KroC100	20749
KroD100	21294
KroE100	22068

Vous pourrez trouver d'autres instances et évaluations optimales sur le site suivant :
`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/`

Exercice 2 – Amélioration grâce à la recherche locale

Question 1 — Proposer une fonction de voisinage et appliquer une recherche locale à partir de la meilleure solution obtenue par l'algorithme génétique.
Question 2 — Remplacer maintenant l'opérateur de mutation de l'algorithme génétique par la méthode de recherche locale : vous obtenez ainsi un algorithme mémétique (combinaison entre algorithme génétique et recherche locale).
Question 3 — Comparer les trois algorithmes pour différentes instances en termes de temps de calcul et qualité de la solution obtenue.