Travail pratique: Problème du voageur de commerce par algorithme génétique

1 Préambule

Lisez tout l'énoncé avant de commencer à programmer!!!

Ce travail pratique a pour but la résolution du problème du voyageur de commerce par algorithme génétique.

1.1 Généralités

Dans ce TP, il y a une grande quantité de fonctionnalités différentes qui sont interdépendantes. Afin de garantir un fonctionnement aussi sûr que possible, il est fortement recommandé d'utiliser des tests. Il est dès lors fortement recommandé d'utiliser le **développement piloté par les tests** (**test driven development**).

1.2 Recommandations

Il faut tenter d'utiliser un maximum les concepts que vous avez déjà vu dans les travaux pratiques précédents.

Entre autres:

1. Modularisez votre code: organisez votre code en plusieurs fichiers/librairies lorsque cela s'y prête.
2. Séparez votre code en fonctions aussi "petites" que possible: à chaque fonctionnalité sa fonction (pensez à ne pas exagérer non plus, additionner un à un entier ne nécessite pas une fonction à part).
3. N'oubliez pas d'essayer d'avoir des codes qui sont aussi résistants aux erreurs que possible (utilisez les `Option` et `Result`), mais n'hésitez pas également à avoir des "paniques" (`panic!()`) lorsque des comportement trop "graves" ou totalement interdits pour être corrigés se produisent.
4. Commentez votre code de façon raisonnable. Il est raisonnable de documenter vos librairies par exemple (ce que fait votre librairie).
5. Utilisez autant que possible l'auto-documentation: vos fonctions et variables doivent avoir des noms aussi "documentants" que possible (cela vous évitera de les documenter explicitement). Lorsque l'auto-documentation n'est pas possible et que le code n'est pas suffisamment explicite (ou que vous pensez que vous utilisez une méthode non-triviale n'hésitez pas).

Soyez proactifs. Profitez des séances de travaux pratiques pour poser un maximum de questions et n'attendez pas la veille du rendu pour nous bombarder de mails...

Vous êtes également fortement encouragés à aller au libre service pour obtenir des conseils et de l'aide pour avancer votre TP.

1.3 Rendu

Votre code source doit être déposé sur `https://cyberlearn.hes-so.ch` avant le **19.02.2019 avant 23h55**.

Il doit être rassemblé dans une archive `.zip` (ou `.tar.gz` ou `.tgz` ou `.tar.bz2` ou ...) à votre nom. Plus précisément, l'étudiant `Michel Lazeyras` mettra ses projets cargo, **après avoir fait un `cargo clean` dans chacun d'entre eux** dans un répertoire nommé `michel_lazeyras` qui sera lui même zippé en un fichier nommé `michel_lazeyras.zip` (une des autre extensions de votre choix). Si vous utilisez une librairie externe qui ne se télécharge pas automatiquement (par exemple la librairie de fractions dans polynômes), n'oubliez pas de l'inclure et de spécifier son **chemin relatif** dans le fichier `Cargo.toml` (vous serez pénalisés si vos codes ne compilent pas à cause de chemins mal spécifiés).

Sous peine de sanctions, vous devez respecter toutes ces spécifications. Ce travail pratique est noté. L'évaluation sera faite sur la base de votre code et d'une interrogation orale lors de laquelle vous expliquerez le travail réalisé.

2 traveling_salesman_problem

2.1 Le problème du voyageur de commerce

2.1.1 Introduction

Le problème du voyageur de commerce (traveling salesman problem) consiste en la recherche du trajet minimal permettant de relier N villes qui sont toutes visitées une seule fois et qui revient à sa ville de départ (voir la fig. 1).

FIGURE 1: Le problème consistant à relier quelques villes allemandes. Source: https://upload.wikimedia.org/wikipedia/commons/c/c4/TSP_Deutschland_3.png

Ce problème classique d'optimisation est très simple à énoncer, mais très difficile à résoudre. En effet, l'algorithme le plus naïf consistant à comparer toutes les chemins possibles nécessite un temps de calcul qui est proportionnel à $N!$. Des algorithmes exacts un peu plus efficaces ont été créés, mais ils ne permettent pas d'aller beaucoup plus loin que $20'000$ villes.

2.1.2 Énoncé mathématique du problème

Soit une liste de N villes toutes reliées par des chemins de longueur connue. Nous numérotons les villes, i, de 0 à $N-1$. Les distances entre les villes i et j sont notées d_{ij} (on a que $d_{ij} = d_{ji}$). Le chemin reliant une ville à elle-même ne fait pas de sens et ne doit pas être représenté (à vous de réfléchir comment encoder cette information).

On peut représenter le problème sous la forme d'un graphe dont les sommets sont les villes et les arêtes représentent les distances entre les villes. Sur la fig. 2, on a un exemple à cinq villes reliées entre-elles

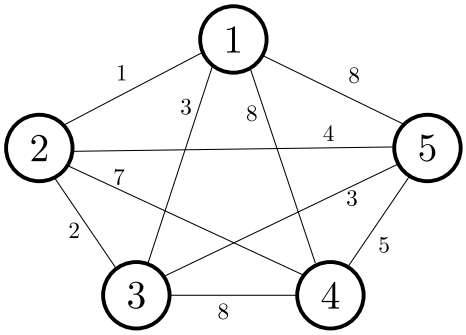

FIGURE 2: Graphe représentant cinq villes reliées entre elles.

Le contenu du graphe peut se représenter sous la forme d'une matrice (un tableau à deux dimensions). Pour le problème de la figure ci-dessus la matrice s'écrit

$$\underline{\underline{d}} = \begin{pmatrix} \infty & 1 & 3 & 8 & 8 \\ 1 & \infty & 2 & 7 & 4 \\ 3 & 2 & \infty & 8 & 3 \\ 8 & 7 & 8 & \infty & 5 \\ 8 & 4 & 3 & 5 & \infty \end{pmatrix} \tag{1}$$

Les distances parcourues étant identiques de i à j et de j à i, on a que la matrice est symétrique. Les ∞ dénotent les parcours impossibles.

Un trajet du voyageur de commerce peut se représenter comme la séquence ordonnée des villes parcourues qu'on peut stocker dans un tableau $\vec{v}$, de longueur N, et où chaque ville n'apparaît qu'une seule fois. Dans le cas à 5 villes, deux chemins possibles seraient par exemple

$$\vec{v}_1 = (0, 1, 2, 3, 4), \quad \vec{v}_2 = (1, 0, 2, 4, 3). \tag{2}$$

On constate que le dernier trajet est sous-entendu.

2.1.3 Remarque

1. Soit une séquence de villes $i = 0, .., N-1$ et leurs coordonnées (x_i, y_i). La distance entre deux villes i et j est donnée par d_{ij}

$$d_{ij} = \begin{cases} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, & \text{si } i \neq j, \\ \infty, & \text{sinon.} \end{cases} \tag{3}$$

2. Chaque trajet est défini à une permutation cyclique près. Ainsi le trajet $\vec{v}_1$ est le même que

$$\vec{v}_1' = (4, 0, 1, 2, 3), \quad \vec{v}_1'' = (3, 4, 0, 1, 2). \tag{4}$$

   De plus comme les chemins sont symétriques, on peut également parcourir les chemins dans le sens inverse le chemin sera identique. Ainsi le trajet $\vec{v}_1$ est équivalent à

$$\vec{v}_1^i = (4, 3, 2, 1, 0). \tag{5}$$

2.1.4 Cahier des charges

Il s'agit de créer:

1. une structure de données représentant le graphe (la matrice des distances);
2. une méthode permettant de générer la matrice des distances à partir des coordonnées des villes;
3. une structure de données représentant le chemin parcouru et de calculer la distance à partir de la matrice des distances;
4. des tests permettant de vérifier le bon fonctionnement de vos fonctionnalités.

2.2 Algorithme génétique pour la résolution du problème du voyageur de commerce

2.2.1 Introduction

Un algorithme génétique est un algorithme évolutionniste utilisé pour résoudre des problèmes d'optimisation de manière approchée, lorsque le calcul exact est trop coûteux. Il utilise la notion de sélection naturelle (inspiré de la théorie de l'évolution de Darwin) et l'applique à un ensemble (une population) de solutions possibles du problème d'optimisation.

De façon générale un algorithme génétique pour la résolution d'un problème d'optimisation fonctionne sur le principe suivant.

0. On construit un ensemble initial d'individus (aussi appelés chromosomes) qui sont constitué de gènes. Ces individus représentent chacun une solution possible du problème d'optimisation.

Puis ont itère jusqu'à ce qu'un critère d'arrêt soit atteint (si aucune modification bénéfique n'apparaît pendant un certain temps par exemple)

1. On choisit parmi ces individus un ensemble de parents qui vont se reproduire.

2. Les couples de parents se reproduisent et donnent naissance à deux enfants qui sont un mélange de leurs gènes (crossover).
3. Les enfants peuvent subir des mutations aléatoires de leurs gènes (mutation).
4. Puis on sélectionne parmi la population globale (parents et enfants) un ensemble d'individus qui survivront selon un critère d'adaptabilité (fitness).

Chacune des étapes ci-dessus peut se réaliser de différentes manières. Nous allons en proposer une dans ce qui suit.

2.2.2 Les individus

Un individu dans le problème du voyageur de commerce est la séquence dans laquelle les villes sont parcourues. Si nous numérotons les villes de 0 à $N-1$, il s'agit donc d'une liste de longueur N qui contient les entiers 0 à $N-1$ apparaissant chacun une seule fois. S'il y a 8 villes, un individu possible est

7	6	2	5	1	3	4	0

En revanche, tout individu ayant un nombre qui se répète ou possédant un nombre plus grand que 7 est non valide. On peut donc noter mathématiquement la liste de ville, $\vec{v}$ comme:

$$\vec{v} = \{v_i\}_{i=0}^{N-1},\ v_i \in [0, N-1],\ \text{où chaque } v_i \text{ apparaît une fois.} \tag{6}$$

2.2.3 L'adaptabilité

L'adaptabilité est la distance totale parcourue lors d'un tour. Plus la distance est faible, plus l'adaptabilité est grande. En utilisant les distances d_{ij} séparant les villes numérotées i et j définies plus haut, on peut calculer la distance totale, s, du parcours $\vec{v}$ comme

$$s = \sum_{i=0}^{N-1} d_{v_i, v_{(i+1)\%N}}. \tag{7}$$

2.2.4 La population

La population est simplement un ensemble de M individus. Nous initialiserons chaque individu aléatoirement au début de l'algorithme. N'oubliez pas que chaque ville ne peut apparaître qu'une seule fois.

2.2.5 La sélection des parents

La sélection des parents se fait sous la forme d'un tournoi. Si nous avons une population de M individus, nous sélectionnons aléatoirement une sous-population de K individus. Le premier parent est l'individu de ce groupe avec l'adaptabilité la plus élevée. En recommençant le processus nous obtenons un deuxième parent que nous faisons se reproduire avec le premier parent. Nous recommençons cette sélection $M/2$ fois afin d'avoir des reproductions de $M/2$ couples.

2.2.6 Le crossover

La reproduction de deux parents $\vec{p}_1$ et $\vec{p}_2$ donne deux enfants $\vec{e}_1$ et $\vec{e}_2$. Ils sont obtenus en mélangeant les génomes de chacun des parents. Comme chaque gène (chaque ville) ne doit apparaître qu'une fois dans chaque individu, ce mélange ne peut pas se faire n'importe comment. Pour simplifier ici, nous choisissons aléatoirement K gènes du parent 1 et les copions à la même position dans l'enfant 1. Puis nous copions les gènes restants du parent 2 dans l'ordre dans lequel ils apparaissent. Ensuite, nous recommençons de même pour l'enfant 2 en commençant par le parent 2.

2.2.6.1 Exemple On peut voir un exemple ci-dessous. Soit $\vec{p}_1$ et $\vec{p}_2$ les deux parents et $K = 3$.

$\vec{p}_1$	7	*6*	2	*5*	1	*3*	4	0
$\vec{p}_2$	**1**	5	**4**	6	**7**	3	**0**	**2**

On a sélectionné au hasard les trois gènes en italique et on les a copiés dans l'enfant 1, $\vec{e}_1$, puis on a copié les gènes restants (en gras ci-dessus) dans l'ordre dans lequel ils apparaissent

$\vec{e}_1$	**1**	*6*	**4**	*5*	**7**	*3*	**0**	**2**

2.2.7 La mutation

La mutation est simplement une permutation de deux gènes, tirés aléatoirement, avec une probabilité p. En d'autres termes, on tire un nombre aléatoire entre 0 et 1: s'il est plus petit que p, on échange deux gènes tirés aléatoirement.

2.2.8 La population finale

La population finale est uniquement composée des enfants et les parents sont tous éliminés de la population.

2.2.9 Cahier des charges

Il s'agit d'écrire une librairie qui permet d'appliquer un algorithme génétique au problème du voyageur de commerce.

Il s'agit de créer:

1. une structure individu représentant un chemin;
2. une structure population représentant un ensemble d'individus;
3. les fonctionnalités de sélection, de reproduction, de mutation, et d'adaptabilité;
4. toutes les fonctionnalités permettant d'importer une population à partir d'une liste de villes et de positions.

Ensuite, il faudra créer un programme cherchant le plus court chemin entre 52

villes se trouvant dans le fichier `berlin52.tsp.raw` (il s'agit des coordonnées de 52 villes). Le chemin optimal se trouve dans le fichier `berlin52.opt.tour.raw`.

2.3 Remarques

2.3.1 Généralités

Un algorithme génétique n'a pas de garantie de convergence. Il n'est donc pas certain que vous trouverez le chemin le plus court. Vous devriez idéalement trouver quelque chose de similaire.

2.3.2 Aides diverses

Afin de vous aider à visualiser vos résultats vous pouvez produire des fichiers `.svg` à l'aide de la librairie `plotlib` que nous avons déjà utilisée dans un TP précédent. Pour ce faire, vous pouvez vous inspirer du code suivant:

```
let points = Vec<(f64,f64)>:new();

// add a lot of (f64, f64) tuples
// which represent the points coordinates
// and use them below

let l1 = plotlib::line::Line::new(&points)
         .style(plotlib::line::Style::new().colour("black"));
let v = plotlib::view::ContinuousView::new().add(&l1);
plotlib::page::Page::single(&v)
    .save("tmp/line".to_owned()
            +&format!("{:0>8}", i)
            +&".svg".to_owned())
    .expect("saving svg");

// this saves a file in ./tmp/line0000XXXX.svg
// where XXXX is a number paved with
// at most 8 zeroes
```

2.3.3 Les fichiers fournis

1. Le fichier 'berlin52.tsp.raw a la structure suivante. Il s'agit de deux colonnes contenant la position, x, y, de chaque ville.
2. Le fichier `berlin52.opt.tour.raw` contient le tour optimal. Le numéro de chaque ville correspond à l'ordre spécifié dans le fichier `berlin52.tour.raw`.
3. Si cela vous intéresse tout un tas de parcours peuvent être trouvés sur le lien suivant: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/

2.4 Bibliographie

Algorithmes génétiques:

0. https://fr.wikipedia.org/wiki/Algorithme_g%C3%A9n%C3%A9tique

1. https://skyduino.wordpress.com/2015/07/16/tutorielpython-les-algorithmes-genetiques-garantis-sans-ogm/

Problème du voyageur de commerce

0. https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_voyageur_de_commerce
1. https://interstices.info/le-probleme-du-voyageur-de-commerce/

Problème du voyageur de commerce par algorithme génétique:

1. http://perso.ensta-paristech.fr/~lunevill/sim_numerique/projets/vdc.pdf
2. https://antoinevastel.com/algorithme/python/algorithmes%20g%C3%A9n%C3%A9tiques/2016/04/30/probleme-voyageur-commerce.html