TP PLC n^{o}7 et 8 :Algorithmes génétiques et voyageur de commerce

Ce projet TP6−8 est à rendre sous la forme :
− > code Prolog
− > compte rendu (PDF)
via Moodle, rubrique 'Rendu du projet'
Date limite de rendu : mercredi 31 mars 2021.
Indiquer les noms d'étudiants en entête des fichiers.
Le compte-rendu devra contenir *a minima* :
- l'indication des différentes fonctions de sélection, mutation, croisement, génération de population initiale implémentées ;
- un exemple de lancement du programme ;
- un bilan des tests effectués (paramètres, résultats).
Au-delà de ce minimum, chaque binôme poussera ses explorations et orientra ses commentaires suivant sa propre inspiration.

Le projet TP6-8 consiste à résoudre le problème du voyageur de commerce par la méthode des algorithmes génétiques. Il faut pour cela écrire au moins une version de chacune des fonctions nécessaires pour faire tourner l'algorithme génétique : les fonctions d'évaluation et de sélection (pour reproduction, pour remplacement) vont venir s'ajouter aux fonctions de mutation et de croisement du TP6. Il faudra ensuite faire fonctionner l'ensemble de la boucle générationnelle, permettant de passer d'une génération à l'autre. Le fonctionnement complet nécessite de plus de constituer une population initiale.
Vous disposez sur Moodle, dans le dossier TP7-8, d'un canevas de programme, `tp7_AG_TROUS.pl`. Ce fichier contient des matrices de distances, dont une pour 29 villes !

1 Fonction d'arrêt : rappel du fonctionnement

Différentes fonctions d'arrêt sont envisageables. L'agorithme peut par exemple s'arrêter au bout d'un certain nombre d'itérations, ou bien quand une certaine performance est atteinte, ou bien quand le meilleur individu est le même depuis un certain nombre d'itérations, etc.
Dans le programme fourni, une fonction d'arrêt par comptage du nombre d'itérations est déjà réalisée, comme cela a été expliqué au TD7. Voici un rappel de sa mise en oeuvre.

Ajout/retrait dynamique de clauses : détails techniques

Techniquement, le compteur d'itérations a été réalisé en utilisant l'ajout/retrait dynamique de clauses, pour éviter de passer un argument supplémentaire aux prédicats. Ainsi, le compte à rebours est initialisé par un fait tel que `decompteurArret(20)`. A chaque tour de boucle, la valeur du compteur est consultée (par unification avec le fait `decompteurArret(V).`). Si le compteur est à 0, arrêt du programme `algoGen`, sinon relance d'un tour de boucle générationnelle et le compteur est décrémenté. Pour cela, le fait `decompteurArret(Val)` doit être remplacé par le fait "`decompteurArret(Val_moins_1)`",

ce qui s'effectue au moyen des prédicats prédéfinis `retract` et `assert`. L'exécution d'un but tel que `retract(decompteurArret(V))` n'est autorisée que si le prédicat `decompteurArret` a été déclaré "dynamique", par ajout dans le programme de

```
:- dynamic  decompteurArret/1.
```

2 Fonction d'évaluation

Ecrire la fonction d'évaluation d'un individu, calculant le coût du parcours à partir de la matrice de distances. Rappel : ce genre de fonction a déjà été écrit au TP PLC n°4 "Voyageur de commerce".

3 Fonctions de sélection

Deux sélections s'appliquent au cours de l'algorithme génétique : la sélection pour reproduction et la sélection pour remplacement.

A- Sélection pour reproduction : quels individus vont se reproduire ? La sélection pour reproduction consiste à choisir `k` individus (les `k` parents) dans la population, destinés à produire `k` enfants (par croisements et mutations). Plus précisément, les `k` parents sélectionnés donneront d'abord `k` enfants, par croisement. Parmi ceux-là seul un petit nombre (par exemple 2) sera muté.

B- Sélection pour remplacement : quels individus vont disparaître ? Les `k` enfants résultants seront alors évalués, afin de subir la sélection pour remplacement. Ici, les `n` individus de départ + les `k` enfants seront réduits en un nouvel ensemble de `n` individus, qui constituera la "génération suivante".

Il existe de nombreuses façons de mettre en oeuvre ces sélections. Une méthode simple et relativement efficace pour réaliser la *sélection pour remplacement* consiste à insérer les `k` nouveaux individus dans la population de départ, et à ne conserver que les `n` meilleurs individus parmi les `n+k` (i.e. principe de selection déterministe).
Le choix de la méthode de *sélection pour reproduction* peut quant à lui être plus déterminant pour les performances de l'algorithme.

Sélection déterministe La sélection déterministe consiste tout simplement à choisir les `k` individus les meilleurs dans la population.

Sélection par tournois La sélection par tournois consiste à choisir aléatoirement `m` individus dans la population et à sélectionner pour la reproduction celui qui a la meilleure performance. Au cours d'une étape de sélection, il y a autant de tournois que d'individus à s'électionner (`k`). Parmi les variantes possibles, les individus qui participent à un tournois peuvent être ou non remis dans la population pour le tournois suivant.

4 Population de départ

La population de départ peut dans un premier temps être produite "à la main". Dans une version plus complète, elle peut par exemple être générée aléatoirement.

5 Organisation du programme

Le programme peut gagner en lisibilité si les fonctions sont mises dans un fichier séparé :
`:-use_module('malibrairie').`