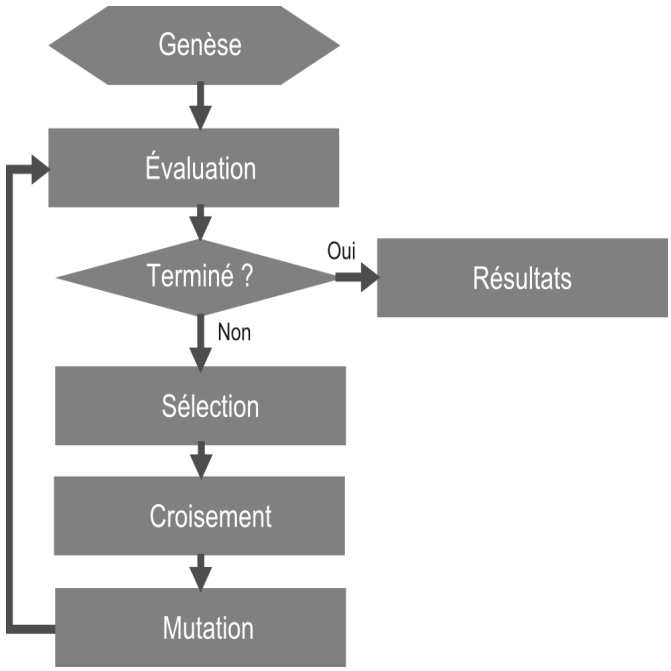
Projet C++
Algorithmes génétiques

Le but de ce projet est de réaliser des outils génériques pour faire des algorithmes génétiques.

I - Informations

I1 - Déroulement d'un algorithme génétique

Le déroulement d'un algorithme génétique est le suivant :



0 - On commence avec une population de N **solutions**, **générées** aléatoirement.

1 - On **évalue** ensuite ces solutions, en leur donnant une note.

2 - Si on atteint un certain **critère d'arret** (note seuil atteinte, nombre d'itérations…) on termine, et on retourne la solution ayant la meilleure note

3 - Sinon, on **sélectionne** un sous ensemble des solutions, on les **croise** entre elles, et on effectue des **mutations** sur les résultats obtenus. On recommence à l'étape 1 avec la nouvelle population.

Vous trouverez de plus amples informations aux adresses suivantes :
http://khayyam.developpez.com/articles/algo/genetic/#L3.1.1
http://www-igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/index.html
https://fr.wikipedia.org/wiki/Algorithme_g%C3%A9n%C3%A9tique

I2 - Parties customisables de l'algorithme génétique

Comme nous voulons des outils génériques, il faut proposer des solutions pour choisir :

- le type **T** des solutions : classiquement, il s'agit de string, vecteur de booléens ou de float, ou d'arbres.
- le générateur aléatoire **G** de solutions : typiquement, un foncteur sans paramètre qui retourne une solution
- un évaluateur **E** : foncteur qui prend en paramètre une solution et retourne une note (un réel)
- un sélécteur **S** : foncteur qui prend en paramètre une population et ses notes, et retourne un sous ensemble de cette population.
- un opérateur de croisement **C** : foncteur qui prend en paramètres deux solutions, et retourne une solution issue de leur croisement
- un opérateur de mutation **M** : foncteur qui prend en paramètres un solution, et retourne une solution obtenue en modifiant celle-ci.
- un critère d'arrêt **F**: qui prend en paramètre les notes d'une population, et retourne un booléen indiquant si on arrête ou non.

Le générateur, l'évaluateur et les opérateurs de croisement et de mutation sont spécifiques au type de solutions, et plus encore, au problème dont on cherche la solution.

Par contre, le sélecteur et le critère d'arrêt sont les même quelque soient le type de solution : on choisira juste les algorithmes que l'on souhaite utiliser, parmi ceux décrits ci-dessous.

I3 - Les sélécteurs

Il existe plusieurs algorithmes de sélections :

- Sélecteur ***par élitisme*** : on garde les N meilleurs solutions
- Sélecteur ***par note*** : on tire aléatoirement les solutions à garder, qui ont une chance d'être choisies proportionnelle à leur note.
- Sélecteur ***par rang*** : on trie les solutions, puis on tire aléatoirement les solutions à garder, qui ont une chance d'être choisies proportionnelle à leur classement.
- Sélecteur ***par tournoi*** : pour sélectionner une solution, on en choisi M aléatoirement, et on choisi celle qui à la meilleure note parmi elles.

I4 - Les critères d'arrêt

Il existe plusieurs critères simples pour décider de l'arrêt d'un algorithme génétique :

- ***seuil de qualité*** : si on obtient une solution suffisamment bonne, on arrête
- ***seuil d'itérations*** : si on itère trop longtemps, on arrête
- ***apparition d'un "plateau"*** : si on n'améliore pas notre solution pendant un certain nombre d'itérations (note reste stable, on parle de plateau)
- combinaison de critères ci-dessus.

II - Ce qui vous est demandé

II1 - Fonction templatisée

Faire une fonction templatisée ***generate***, qui prend en paramètres

- un générateur
- un évaluateur
- un sélectionneur
- un opérateur de croisement
- un operateur de mutation
- un critère d'arrêt
- une taille de population

et qui retourne un élément de type T, résultat de votre algorithme génétique.

II3 - Selecteurs

Créer les classes correspondant aux différents sélecteurs définis en I3.

II4 - Critères d'arrêts

Créer les classes correspondant aux différents critères d'arret définis en I4.

II5 - Résolution de problème

Prenez un problème de votre choix résolvable par algorithme génétique et implémentez les classes correspondant au générateur, à l'évaluateur, et aux opérateurs de croisement et de mutation correspondant à votre problème.

Quelques idées de problème :

- trouver une phrase cachée (l'évaluateur donne le nombre de caractères corrects)
- trouver le point a N dimensions tel que sa distance moyenne à un ensemble de points est minimale
- problème du voyageur de commerce (https://fr.wikipedia.org/wiki/Algorithme_g%C3%A9n%C3%A9tique#Applications_dans_la_recherche)

Si vous faites plusieurs problèmes, ou un problème difficile (utilisant des arbres comme solutions, par exemple) cela sera pris en compte dans votre note.

II6 - Classe templatisée

Faire une classe GeneticAlgorithm, templatisée par :

- un générateur
- un évaluateur
- un sélectionneur
- un opérateur de croisement
- un operateur de mutation

- un critère d’arrêt

Cette classe aura une méthode **run** qui prendra en paramètre une taille de population, et retournera le résultat de l’algorithme génétique (cette méthode peut appeler la méthode generate créée dans la partie II1).

II7 - Spécialisation de template

Faire une spécialisation de GeneticAlgorithm, en considérant que le critère d’arrêt sera toujours un nombre d’itérations (précisé dans le constructeur) et la selection sera toujours faite par tournoi.