TME 10 : stratégies d'évolution

Objectifs pédagogiques : implantation du pattern Strategy

Nous allons appliquer le *Design Pattern* Strategy au projet que nous avons développé jusqu'ici. Créez un projet « tme10 » et importez l'archive de votre TME9.

1. Stratégies d'évolution

Votre projet contient les opérations élémentaires qui permettent de créer un individu à partir de 2 individus sélectionnés, puis de l'évaluer avant d'être inséré dans la population. Sur cette base, nous avons mis en place un algorithme qui permet de modifier une population de génération en génération.

Vous allez à présent mettre en place deux mécanismes d'évolution distincts :

- le premier, de type générationnel, est celui que nous avons utilisé jusqu'ici.
- le second, plus progressif, consiste à sélectionner deux parents en fonction de leur fitness, à engendrer un rejeton à partir de ces parents et de remplacer un individu choisi en fonction de sa fitness par ce rejeton. De même que dans le cas générationnel, on effectue cette opération un nombre de fois fixé a priori ou bien jusqu'à ce que l'un des individus engendrés soit totalement satisfaisant.

1. Stratégies de sélection d'un parent

Vous allez faire appel au pattern *Strategy* pour permettre le choix entre la sélection uniforme existante et une sélection dépendante de la fitness. Toutes les opérations réalisées dans le cadre de ce TME se situent dans le *package algogen*.

Créez une interface *IndivSelecteur* qui définit une méthode *getRandom(Population pop)* qui renvoie un *Individu* de la population passée en paramètres.

Créez une classe *SelecteurUniforme* qui implémente *IndivSelecteur* et dont la méthode *getRandom(Population pop)* renvoie un individu tiré au hasard dans la population à laquelle ce sélecteur est associé.

Dans la classe *Population*, ajoutez une méthode *getSommeFitnesses()* qui renvoie la somme des fitnesses de tous les individus présents dans la population.

Créez une classe *SelecteurParFitness* qui implémente *IndivSelecteur.* La méthode *getRandom(Population pop)* de cette classe associe à chaque individu de la population passée en paramètres une probabilité de sélection proportionelle à sa fitness selon une méthode de sélection appelée « roue de la fortune » décrite par la figure ci-dessous. L'idée est que chaque individu se voit attribuer une proportion de la roue proportionnelle à sa fitness, on lance la roue et on choisit l'individu sur lequel est le pointeur quand elle s'arrête.

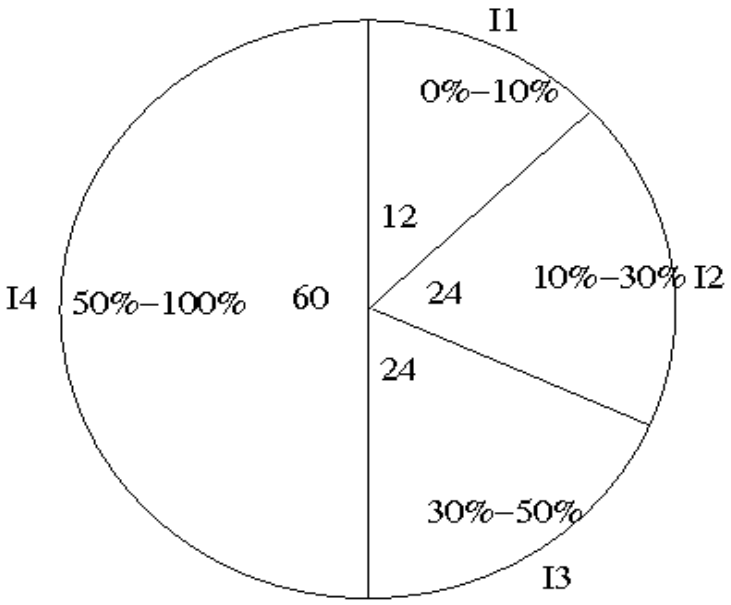

Illustration du principe de la roue de la fortune. Chaque individu de la population a une probabilité d'être sélectionné comme « parent » proportionnelle à sa fitness ou à son rang. La population contient les individus I1 à I4 de fitness respectives 12, 24, 24 et 60. On calcule leur fitness relative en divisant par la somme des fitness (ici, 120). Si on tire 0.2814, c'est I2 qui sera choisi (car le nombre tiré est entre 10% et 30%).

Pour réaliser la sélection par la roue de la fortune, on découpe l'intervalle [0,1[en sous-intervalles dont la taille est proportionnelle à la fitness de chacun des individus, puis on tire un nombre au hasard dans [0,1[et on renvoie l'individu dont l'intervalle est pointé par ce nombre. En pratique, l'agorithme est le suivant. Soit X le produit du nombre tiré aléatoirement par la somme des fitnesses de tous les individus. On se dote d'un compteur de fitness initialisé à 0 et on lui ajoute la fitness de chaque individu au fur et à mesure qu'on les parcourt. Quand ce compteur dépasse la valeur X, l'individu pointé est renvoyé.

2. Implantation des stratégies d'évolution

Pour permettre le choix entre les deux mécanismes d'évolution décrits en introduction, vous allez faire appel au pattern *Strategy.*

*A*u sein de la classe *Population*, assurez-vous que la méthode de production d'une nouvelle génération *Population reproduire(Population pop)* intègre aussi l'opération de mutation des individus nouvellement créés. La nouvelle signature de la méthode sera *Population reproduire(Population pop, double ratio).*

Créez une interface *IEvolution* qui définit la méthode *Population reproduire(Population pop, double ratio)*. Insérer cette interface en tant qu'attribut de la classe *Population* et déléguez-lui l'appel de la méthode *Population reproduire(Population pop, double ratio)* qui doit se trouver déjà au sein de la classe *Population*.

Associez à l'interface *IEvolution* deux implémentations : les classes *EvolutionGenerationnelle* et *EvolutionProgressive*. Ces classes contiennent un attribut de type *IndivSelecteur* qui leur est passé lors de la construction et qui permet, via un pattern Strategy, de choisir les individus soit de façon uniforme, soit en fonction de leur fitness.

La méthode *Population reproduire(Population pop, double ratio)* de la classe *EvolutionGenerationnelle* se comporte comme celle que vous aviez définie précédemment dans la classe *Population*.

Celle de la classe *EvolutionProgressive* choisit les parents de façon proportionnelle à leur fitness et installe leur rejeton à la place de l'individu qui a la plus mauvaise fitness au sein de la population. Pour implémenter ce dernier point, ajoutez une méthode *void removeLast()* au sein de la classe *Population.*

Pour décider si vous faites appel à une évolution générationnelle ou à une évolution par niche, vous ajouterez au fichier de configuration créé lors du TME précédent une nouvelle entrée *« evolutionGenerationelle »* qui vaut *true* par défaut. De même, pour le choix uniforme ou par fitness des individus, vous ajouterez une entrée *« selectionUniforme »* qui vaut *true* par défaut. Vous ferez en sorte que la classe *Population* crée les mécanismes d'évolution et de sélection du bon type en fonction de la valeur donnée à ces paramètres.

Faites le nécessaire pour tester le comportement de ces deux stratégies sur les deux types d'individus sur lesquels nous avons travaillé jusqu'à présent (les expressions arithmétiques et les agents).

Pour tester les différentes configurations, vous noterez qu'il faut appeler l'évolution par niche sur un bien plus grand nombre de génération, chaque génération de remplaçant qu'un individu à la fois.

3. Remise du TME

Question : fournissez les performances des deux stratégies d'évolution codées en précisant les paramètres que vous avez utilisés dans chaque cas. Que pouvez-vous conclure de la comparaison ?