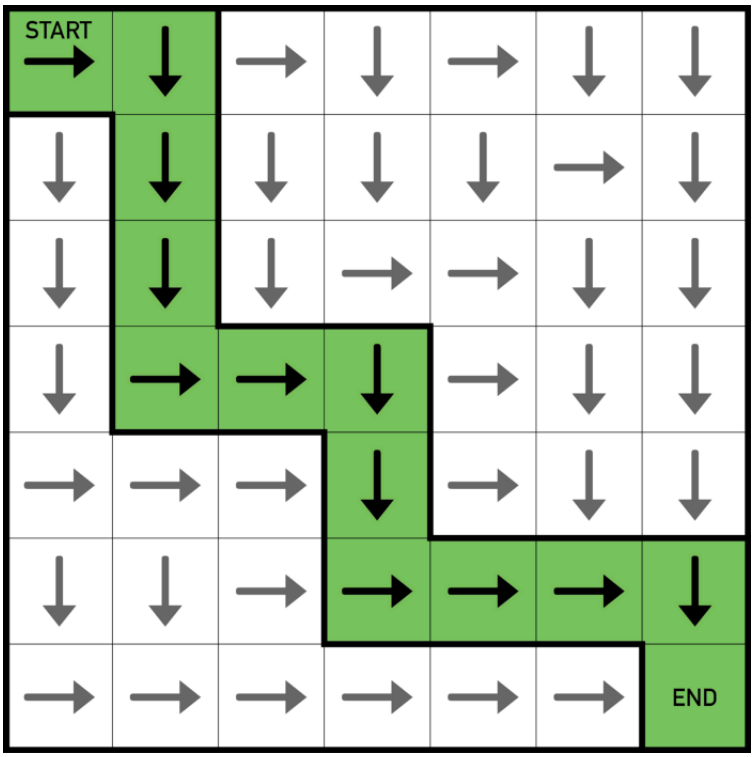
Algorithmes Génétique pour un Jeu de Grille

L'objectif de cette étude de cas est de concevoir et de modéliser le problème de collect de pièces dans une Grille, ainsi qu'une solution basée sur les Algorithmes génétiques. Le but du jeu est de collecter un nombre maximal de pièces en faisant k mouvements seulement.



Un utilisateur a la possibilité de configurer une instance du jeu :
- saisir la dimension de la grille.
- le nombre de pièces.
- saisir le nombre de pas autorisé.
- saisir le point de départ de la grille.

Ainsi que de lancer l'algorithme et de visualiser les résultats.

Question

Donnez un diagramme de cas d'utilisation de ce cas d'étude.

1 Instance du jeu de grille

Une instance du jeu est définie par une grille G de taille $n \times m$, une case de départ c_d et un nombre k de pas autorisés. la grille contient un nombre de pièces t (une case de la grille peut soit contenir une pièce ou rien), les cases de G sont représentées par des coordonnées $(ligne, colonne)$. Soit une séquence de $k+1$ cases $s =< c_d, \ldots, c_k >$ tel que c_i, c_{i+1} sont des cases voisines. Une Solution pour le jeu est une séquence s. Une instance vérifie si une solution proposée est valide ou pas.

Question

Donnez un diagramme de classes comportant les éléments mentionnée ci-dessus.

2 Algorithme Génétique

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnistes. Leur but est d'obtenir une solution approchée à un problème d'optimisation. Dans notre cas c'est de récolter le maximum de pièces possible (idéalement toutes les pièces) en faisant k pas dans une grille. L'algorithme prend en entrée une instance du problème, a partir d'une population initiale, il produit une nouvelle population après des étapes de sélection, de croisement et de mutation. Le traitement se fait de manière itérative et $nbGen$ fois. L'algorithme retourne comme solution la meilleur séquence s trouvée. Les algorithmes génétiques se diffèrent principalement par leurs opérateurs de sélection, de croisement, de mutation et la génération des nouvelles populations.

Sélection

La sélection permet de sélectionner des individus candidat pour calculer la prochaine génération, ces candidats seront de nouveau présent dans la nouvelle génération.

Croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des individus. Les croisements sont envisagés avec deux individus préalablement sélectionnés de la population courante et ils génèrent un enfant qui est à ajouter dans la prochaine population.

Mutation

La mutation d'un individu sélectionné revient à modifier une partie de sa structure pour produire un individu muté qui est à ajouter dans la prochaine population.

Composantes de l'algorithme génétique

Nous vous proposons l'architecture suivante avec la liberté de faire des choix qui peuvent impacter considérablement l'efficacité de la solution :

Individu

Un individu est un chemin dans la grille qui commence depuis un point de départ c_d et termine après k pas. Il y a plusieurs façons d'encoder un individu, on vous propose les deux encodages suivants :

Individu GDHB (Gauche,Droite,Haut,Bas)

Depuis une case de départ c_d, cet encodage consiste a créer k mouvements aléatoire ou chaque pas peut prendre une des valeurs G, D, H, B (ex : $DHDBGGBDH$). On a trois types d'individus GDHB :

— **IndividuGDHBSimple :** cet individu a une fonctionnalité de croisement et mutation basiques.
Croisement : soit deux individus de ce type, pour faire un croisement entre eux on a les étapes suivantes :
— on tire aléatoirement un nombre de pas $p \in [0, k-1]$.
— on avance p pas dans le premier individu depuis le point de départ c_d, on avance $k-p$ pas dans le deuxième individu depuis c_p et on récupère les parcours $s_{i1},\ s_{i2}$.
— on colle $s_{i1},\ s_{i2}$ pour avoir un fils.
— le fils est normalisé afin qu'il reste dans la grille.
Exemple : Soit deux individus $indiv_1 = (BDDHDD)$,$indiv_2 = (DDDBGB)$, une valeur de $p = 3$. Après le parcours des deux individus on aura $indiv_1 = (BDD)$ et $indiv_2 = (BGB)$ et le fils du croisement sera $fils = (BDDBGB)$.
Mutation : Pour muter un individu GDHBSimple on a les étapes suivantes :
— on tire aléatoirement deux valeurs $p_1, p_2 \in [0, k-1]$.
— on fait une permutation entre c_{p1} et c_{p2}
— l'individu muté doit être normalisé afin qu'il reste dans la grille.
Exemple : Soit le fils généré précédemment $fils = (BDDBGB)$ et $(p_1 = 1, p_2 = 3)$ la mutation du fils donne l'individu $fils = (BBDDGB)$.
Normalisation d'un individu : Lors du croisement et la mutation il y a la possibilité de générer des individus fils qui ne sont pas valident c-a-d. ils dépassent la grille. Pour cela chaque individu non valide dois être normalisé en suivant ces étapes :
— on supprime tout les mouvements qui sortent de grille.
— ensuite on tire aléatoirement des mouvement qui sont valide pour compléter les k mouvements.

— **IndividuGDHBSmartCrossing :** ce type individu a une fonctionnalité de croisement intelligente.
Croisement : soit deux individus $indiv_1$ et $indiv_2$ de ce type, si on veut faire un croisement entre eux on a les étapes suivantes :
— on tire aléatoirement un nombre de pas $p \in [0, k/2]$.
— on avance p pas dans les deux individus et on récupère les points d'arrivé $c_{indiv_1},\ c_{indiv_2}$.
— on avance de c_{indiv_1} vers c_{indiv_2} en prenant le plus cours chemin et en faisant $k-p$ pas ou moins.
— si il en reste des pas on ajoute des points de $indiv_2$ jusqu'à avoir fini les pas.
— le fils produit commence par les p premiers pas de $indiv_1$ et fini avec le chemin généré entre c_{indiv_1} vers c_{indiv_2}.
Exemple : Soit un point de départ $c_d = (0, 0)$, deux individus $indiv_1 = (BDDHDDBBDH)$,$indiv_2 = (DBDBGBBBDB)$ et une valeur de $p = 4$. Après le parcours des deux individus on aura $indiv_1 = (BDDH)$ et $indiv_2 = (DBDB)$ les deux derniers pas des deux individus sont $indiv_1 = H, indiv_2 = B$ c-a-d les points d'arrivé sont $c_{indiv_1} = (2, 2),\ c_{indiv_2} = (0, 2)$ le plus cours chemin de $(2, 2)$ a $(0, 2)$ c'est $\{(2, 2) \to (1, 2) \to (0, 2)\} \leftrightarrow (H, H)$ alors pour le croisement en prend les p premier pas de $indiv_1$ plus le chemin entre $(2, 2)$ et $(0, 2)$: $fils = (BDDHHH)$ et la il nous restent 4 pas alors en complète avec les pas de $indiv_2$ ce qui fait $fils = (BDDHHHGBBB)$
Mutation : La mutation de ce type d'individu est la même que celle de IndividuGDHBSimple.

— **IndividuGDHBSmartCrossingSmartMutation :** ce type individu a la même fonctionnalité de croisement que IndividuGDHBSmartCrossing, mais il a une fonctionnalité de mutation intelligente.
Mutation Pour muter cet individu on a les étapes suivantes :
— On tire une aléatoirement une valeur $p \in [1, k-2]$, et on prend trois points (c_{p-1}, c_p, c_{p+1})
— si (c_{p-1}, c_p, c_{p+1}) sont des mouvements identique alors on créer un "crochet".
Exemple : si on a DDD alors on aura $DHDBD$, pareil pour tout les différents mouvement (GGG, BBB, HHH).

— sinon on prend deux mouvements consécutives différentes et on les inversent.
Exemple : si on a DDB alors on aura DBD, si on a BDH alors on aura DBH

Note : il faut gérer les cas ou il y a des chemins non valides.

Individu Permutation

Cet encodage fournit une information sur le positionnement des pièces dans la grille, il est généré a base de l'ordre avec le quel on récolte les pièces.
Exemple : si on a 10 pièces et 5 mouvements autorisé alors un individu permute peut être $(1, 10, 5, 2, 4)$ c-à-d. il va récolter la première pièce en premier en suite la dixième après ainsi de suite.

— **Croisement :** Pour croiser deux individu ind_1, ind_2 on a les étapes suivantes :
  — tirer au hasard deux entier $d, f \in [0, k-1]$.
  — prendre les éléments de ind_1 qui sont dans l'intervalle $[d, f]$.
  — compléter avec les éléments de ind_2 qui sont différents des éléments pris depuis de ind_1.
  Exemple : $ind_1 = (0425136)$, $ind_2 = (2316450)$ et $[d, f] = [3, 5]$, on a $ind_1[3, 5] = 513$ qui est mis en premier est on ajoute les valeurs de ind_2 différentes de $ind_1[3, 5]$, le fils généré après le croisement est $fils = (5132640)$.
— **Mutation :**
  — on tire aléatoirement deux valeurs $p_1, p_2 \in [0, k-1]$.
  — on fait une permutation entre c_{p1} et c_{p2}
  Exemple : Soit le $fils = (5132640)$ et les valeurs $p_1 = 0, p_2 = 5$, le résultat de la mutation sera donc $fils = (4132650)$

Fitness

La fitness d'un individu est une fonction d'évaluation qui prend en compte le nombre de pièces récolté après le parcours $f(individu) = 1 + 10 * score(individu)$.

Solution

Un individu génère une solution depuis son encodage.

Population Initial

Une population est un ensemble d'individus ou le nombre d'individus est fixé par l'utilisateur, la population initial est celle avec laquelle l'algorithme commence sa résolution.

Sélection Par Roulette

Nous allons utiliser un type de sélection nommé sélection par roulette, un individu est sélectionné d'une population par une probabilité de $fitness(individu)/S$ ou S est la somme des fitness de tout les individus dans la population, pour faire cela nous avons les étapes suivantes :

— on tire aléatoirement une valeur $r \in [0, S-1]$.
— on parcours d'une manière itérative les individus $indiv_i$ de la population et on fait la somme des fitness a chaque fois $s = s + fitness(indiv_i)$ (au début $s = 0$), si on trouve que la somme partiel des individus parcourus est supérieur ou égal a r ($s \geq r$) on s'arrête et on retourne l'individu $indiv_i$.
  Exemple : Soit une population de trois individus $p = \{indiv_1, indiv_2, indiv_3\}$ avec les valeurs fitness $f(indiv_1) = 2$, $f(indiv_2) = 5$, $f(indiv_3) = 3$ et une valeur $r = 6$. On parcours la population on faisant la somme des valeurs de fitness et quand on arrive a l'individu $indiv_2$ on voit que $s = 2 + 5 = 7$ ce qui fait on retourne $indiv_2$ car $s > r$.

Calcule des nouvelles générations

Le calcule de la nouvelle génération ce fait depuis la population initial et en suivant trois étapes principales (Sélection,Croisement,Mutation) :

Croisement Mutation V1

Dans cette version on sélectionne (par roulette) a chaque fois deux individu de la population ensuite on fait un croisement entre eux pour générer un fils qui est par la suite muté avec une probabilité de mutation p et ajouté a la nouvelle génération. En ajoute a la nouvelle génération les $n = 2$ meilleurs individus (parents) de la population précédente aussi.

Croisement Mutation V2

Dans cette version on sélectionne (par roulette) a chaque fois deux individu de la population ensuite on leurs fait une mutation avec une probabilité de mutation p, ensuite on les croise pour générer un fils qui est ajouté a la nouvelle génération. En ajoute a la nouvelle génération les $n = 2$ meilleurs individus (parents) de la population précédente aussi.

3 Un Solveur Glouton

Afin de comparer votre algorithme génétique, on met en œuvre un solveur d'instances basé sur la stratégie glouton(greedy). L'algorithme du solveur suit les étapes suivantes :

— depuis un point de départ c_d il prend le chemin vers la pièce la plus proche.
— il répète l'étape précédente jusqu'à avoir fini le nombre de pas autorisé k.

Chaque Instance a sont solveur glouton.

Questions

— Complétez le diagramme de classes avec les éléments mentionnée ci-dessus.
— En suivant le patron **FactoryPattern**, créez les créateurs des différents types d'individus :
  — $GDHBSimpleCreator$
  — $GDHBSmartCrossingCreator$
  — $GDHBSmartCrossingSmartMutationCreator$
  — $IndividuPermutCreator$
— Donnez les diagrammes de séquences des méthodes :
  — CréerPopulationInitialGDHBSimple(instance).
  — CréerPopulationInitialPermut(instance).
  — CalculerProchaineGénerationV1(pop).
  — CalculerProchaineGénerationV2(pop).