TD d'algorithme génétique
Contrôle d'un espace entre des unités

Encadrants :

- Thomas Devogele

Langage de programmation :
Java ou C++

Contexte

Un espace a été découpé en **cellules** formant une **matrice** de taille N × M. Pour contrôler cette espace nous disposons de NB_U **unités**.
Chaque cellule est caractérisée par la difficulté à la contrôler.
Chaque unité est caractérisée par sa **capacité** de contrôle. Elle est localisée sur une **cellule (point de départ)** et contrôle un groupe de cellules (**la zone de contrôle**). Chaque cellule fait partie du groupe dont le point de départ est le plus proche. En cas d'égalité de deux distances, l'unité dont la capacité est la plus importante contrôle la cellule. Une unité ne peut pas contrôler un groupe de cellules dont la somme des difficultés est supérieure à sa capacité de contrôle.

Exemple

Pour un espace décomposé en une matrice 4 × 4 (coordonnées ligne colonne) et 3 unités.

Unité 0 (blanche) :	capacité 21	départ en 1 0	(taux de charge calculé 13/21 = 62%)
Unité 1 (grise foncée) :	capacité 40	départ en 2 2	(taux de charge calculé 33/40 = 82%)
Unité 2 (grise claire) :	capacité 45	départ en 3 1	(taux de charge calculé 19/45 = 42%)

	[0]	[1]	[2]	[3]
[0]	3	2	5	8
[1]	5	2	1	9
[2]	1	2	3	5
[3]	4	5	8	2

Difficultés des cellules [i] [j]

	[0]	[1]	[2]	[3]
[0]	0	0	1	1
[1]	0	0	1	1
[2]	0	2	1	1
[3]	2	2	2	1

Numéro de l'unité contrôlant la cellule [i] [j]

Le tableau de droite fournit une solution possible à ce problème. Les cellules encadrées représentent les points de départ.

Evaluation

Pour répartir au mieux les cellules entre les unités, nous allons tenir compte de **deux critères** :

- **Des zones de contrôle de taille homogène.** Chaque zone de contrôle regroupe N_u cellules. Les zones doivent être de taille homogène. La variance de la taille des groupes est calculée pour évaluer l'homogénéité des tailles.
- **La répartition de la charge homogène** entre les unités en fonction de leur capacité. Il vaut mieux que chaque unité soit à 80 % de ses capacités plutôt que des unités à 100% et une unité à 20 %. Ce pourcentage est appelé **taux de charge**. La variance des taux de charge est calculée pour évaluer la répartition.

Objectif

Le but de ce TD est d'obtenir la "meilleure" solution à l'aide d'un algorithme génétique. Pour cela, un programme en Java ou en C++ sera écrit et il sera testé (rapidité, convergence) en fonction de différents paramètres. Ces derniers sont la taille de la matrice, le nombre d'unités, la taille de la population, le taux de mutation, la fonction d'évaluation et le nombre de générations.

Modélisation

Nous allons commencer par définir ensemble les étapes soulignées en bleu. Les autres étapes sont imposées dans un premier temps.

1. *Choix d'une fonction d'évaluation*

La première étape consiste à **définir une fonction d'évaluation** FE. Elle doit prendre en compte les deux critères : taille homogène et charge homogène.

Solution retenue :

La valeur de la **fonction d'évaluation (FE)** est obtenue de la manière suivante :

fe =(int) (1./critere_taille+ 1./critere_charge)

Remarque : Si une des unités est en surcharge, la fonction d'évaluation (FE) aura arbitrairement la valeur 1. Si deux unités ont le même point de départ, la fonction d'évaluation (FE) aura arbitrairement la valeur 2.

2. *Codage*

La deuxième étape est de **définir un codage adapté** pour représenter les solutions et définir une méthode de croisement simple.

Solution retenue :

Une solution est codée sous la forme d'une matrice de deux colonnes (pt[NB_U][2]). La ligne i correspond aux coordonnées (ligne, colonne) du point de départ de l'unité u.

Unité 0 blanche
Unité 1 grise foncée
Unité 2 grise claire

pt [3] [2]

1	0
2	2
3	1

A partir de cette matrice est calculée une matrice N × M u_ctrl ou la cellule i,j de cette matrice contient le numéro de l'unité contrôlant la cellule i,j de l'espace.

0	0	1	1
0	0	1	1
0	2	1	1
2	2	2	1

u_ctrl [i] [j]

3. *Génération de la population initiale*

P solutions sont générées aléatoirement. P est un nombre pair. Pour chaque solution, la matrice définissant quelle unité contrôle quelles cellules est alors calculée.

4. *Evaluations des solutions*

Chaque solution doit alors être évaluée à l'aide de la fonction définie en 1.

5. *Sélectionner des couples de parents*

La sélection des parents est réalisée à l'aide d'une **roue de la fortune**. Chaque solution ayant une chance de se reproduire proportionnelle à sa fonction d'évaluation. P solutions sont sélectionnées. Une solution peut être sélectionnée plusieurs fois. Les 2 premières vont se reproduire, puis les 2 suivantes et ainsi de suite.

1. *Croiser (reproduction des parents)*

Pour croiser 2 solutions parents, le principe retenu est, pour chaque unité, un aléa de valeur 0 ou 1, est tiré au sort. Si l'aléa est égal à 1 alors les coordonnées de l'unité de l'enfant 1 est égale au coordonnées du parent 1 et les coordonnées de l'unité de l'enfant 2 est égale au coordonnées du parent 2. Si l'aléa est égal à 0, c'est l'inverse.

2. Réaliser une mutation

La mutation consiste simplement à :

- choisir aléatoirement une solution,
- choisir aléatoirement un point de départ d'une unité de cette solution,
- modifier aléatoirement les coordonnées de ce point.

3. Evaluer les enfants et les mutants

Chaque enfant et mutant doit alors être évalué.

4. Définir la nouvelle population

Pour définir la nouvelle génération, les individus de l'ancienne génération, les enfants et les mutants sont :

- Triés en fonction de la valeur de la fonction d'évaluation de chaque solution,
- Épurés ; Les doublons s'il existe sont identifiés, la FE d'un des doublons est mise à zéro,
- Trié à nouveau,
- Réduite ; seulles les P meilleures solutions sont gardées.

Critère d'arrêt

Le programme est interrompu après un nombre de générations (NB_GEN) défini préalablement.

Programmation, test et évaluation

Une fois la modélisation terminée, La TD consiste à programmer et tester un algorithme génétique pour répondre à ce problème. Des classes et une partie des méthodes écrites en C++ vous sont fournies.

Manipulation de solutions

1. Compilez le projet, un espace et les unités ont été générés aléatoirement et les valeurs doivent être affichées.
2. Complétez le programme principal, en déclarant une solution s0 à l'aide du constructeur solution(unites). Puis, calculez la fonction d'évaluation, avant d'employer la méthode afficher de solution.
3. Codez la fonction de **croisement**. Pour générer un aléa vous pouvez employer la fonction : (rand() % 2). N'oubliez pas de calculer la matrice de calcul puis la fonction d'évaluation pour les enfants.
4. Après avoir créé une deuxième solution s1 appelé la méthode de **croisement**. Affichez les 2 solutions enfants.

Manipulation d'une population de solutions

Maintenant que vous savez manipuler des solutions, nous allons manipuler des populations de solutions.

5. Programmez une fonction de **reproduction.** Cette fonction sélectionne P parents à l'aide d'une **roue** de la fortune. Elle les met dans un vecteur parents. Elle **croise** les parents. Les enfants sont alors ajoutés à la population.
6. Appelez cette méthode dans la fonction main(), puis affichez les 2P solutions de la population à l'aide de affiche().
7. Ajoutez un mutant à la population puis affichez les 2P+1 solutions de la population
8. Triez les solutions de la population puis affichez les FE des 2P+1 solutions à l'aide de affiche_fe().
9. Employez la méthode doublon() et triez à nouveau la population.
10. Affichez les 2P+1 fe des solutions de la population à l'aide de la fonction affiche_fe de population.
11. Réduisez la population à l'aide de la méthode reduire() ;

12. Vous avez réussi à créer une nouvelle génération, répétez NB_GEN fois cette séquence afin d'obtenir les générations successives.

Optimisation

Il est possible d'optimiser et de tester des variantes de ce programme

13. Faites varier les constantes (nombres d'individus de la population, nombres de générations) pour observer le comportement de l'algorithme.)
14. Ajoutez un test d'arrêt qui permettra d'arrêter la boucle, si trois populations consécutives (après avoir réduit le nombre de solutions) ont une somme des fe de leurs solutions, identiques.
15. Remplacez la sélection à l'aide d'une roue de la fortune par une sélection par un tournoi entre X solutions (X>2). Le tournoi permet de retenir comme parents, les 2 meilleurs parmi les X solutions tirées au sort. Comparez les résultats obtenus avec ceux obtenu à l'aide de la roue de la fortune.