Optimisation Stochastique

Sujet 3 :Composition des poules pour un championnat

Alexandre ENOUF | Tom GIMENES | Optimisation Stochastique | Mai 2016

Sommaire

Présentation du projet 3

Objectifs et Description 3

Étude et conception 3

Fonctionnement du projet 4

Algorithme génétique 4

Fonctionnement du programme 5

Présentation technique du projet 6

Documentation du projet 9

Difficultés rencontrées 10

Webographie 11

Présentation du projet

Objectifs et Description

L'objectif du sujet est de répartir les équipes dans différents groupes d'un championnat suivant deux critères : la distance parcourue par les équipes et l'équité sportive.

La problématique étant de trouver le meilleur compromis entre la distance totale parcourue par les équipes et l'équilibre des poules par rapport au niveau des équipes.

Pour résoudre cela, nous allons utiliser un algorithme génétique dont on expliquera le fonctionnement par la suite.

Les données à notre disposition sont une matrice des distances entre les diverses équipes et le niveau sportif de celles-ci.

Dans ce projet, nous devrons gérer dans un premier temps un cas simple de dix-huit équipes réparties en deux poules de neuf et jouant une fois chaque adversaire de la poule.

Dans un second temps, on envisagera le cas d'un championnat à douze équipes dont les équipes sont réparties en deux groupes de six et deux sous-groupes de trois au sein de chaque groupe de six. Les équipes se déplacement deux fois chez chaque adversaire du sous-groupe, une fois chez chaque autre adversaires du groupe et une fois chez trois des six adversaires de l'autre groupe.

Nous expliquerons en détails les caractéristiques et différences de chaque cas par la suite et nous verrons ainsi pourquoi un cas est plus complexe que l'autre.

Étude et conception

Pour la réalisation de ce projet, nous avons utilisé le langage de programmation JAVA.

C'est un langage connu par nous deux qui offre une api conséquente.

Fonctionnement du projet

Algorithme génétique

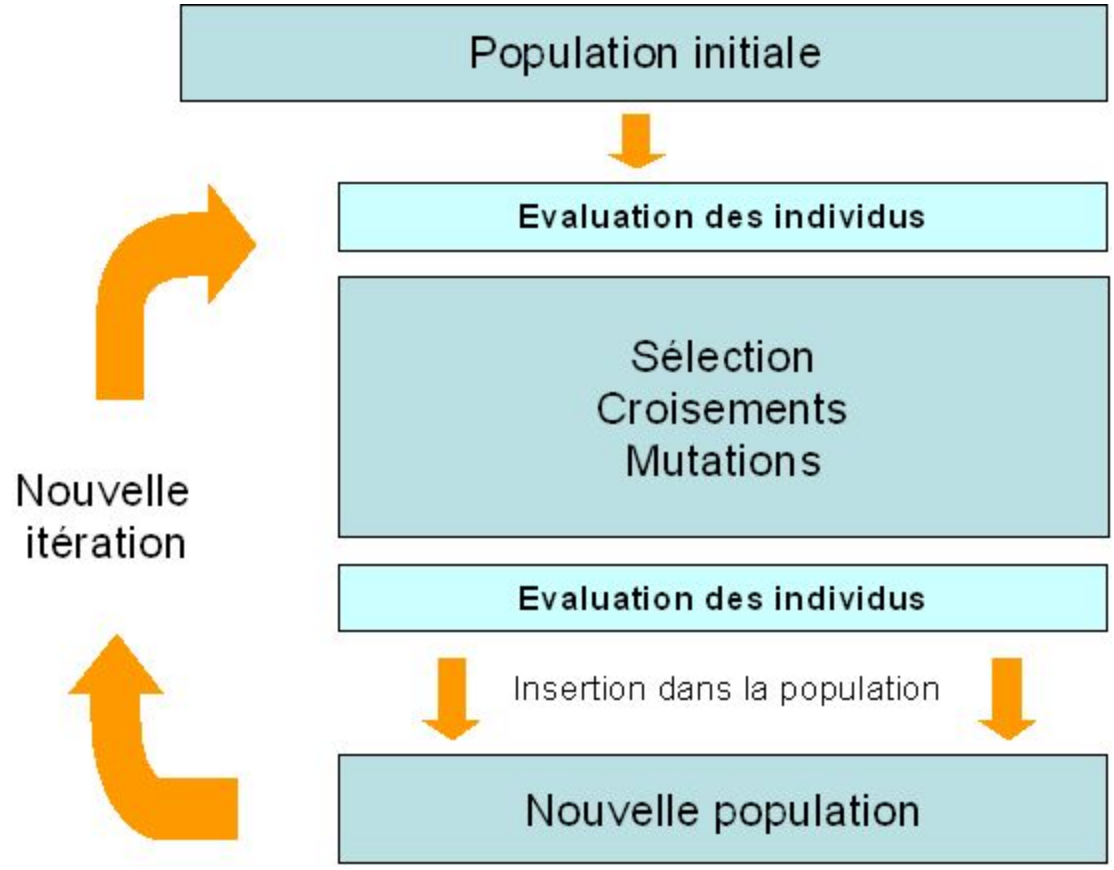
Tout d'abord, nous allons expliquer les spécificités d'un algorithme génétique qui est le cœur du programme.

Un algorithme génétique s'inspire de l'évolution des espèces dans leur cadre naturel.

L'évolution, l'adaptation, la reproduction ou encore l'amélioration, sont le genre de concepts qu'applique cet algorithme.

Dans notre cas, nous ferons évoluer une population dans le but d'en améliorer les individus qui sont des championnats.

Cette évolution passera par une sélection et la mutation d'un nombre d'individus qui fourniront à chaque fois une nouvelle solution qui sera meilleure ou égale.



Shéma du fonctionnement d'un algorithme génétique

Il existe différents types de selection telles que la sélection par roulette, par rang, par différents types tournois et par élitisme. Notre programme implémente une sélection par élitisme en gardant d'une génération sur l'autre les meilleurs individus.

Nous n'avons pas implémenté des mutations, car croiser deux championnats ne mene à rien. En effet il faut que les poules des deux championnats soient identiques pour effectuer un croisement afin qu'une équipe ne soit pas dans les deux poules et le nouveau championnat est une copie de ses parents.

Les mutations que nous avons implémentées sont aléatoires.

Dans notre programme le taux de sélection et de mutation des individus d'une génération sur l'autre sont fixe. Le taux de sélection est de trente-cinq pourcent de la population, celui de mutation est de trente-trois pourcent, les derniers trente-deux pourcent sont des championnats aléatoires.

Fonctionnement du programme

Pour obtenir et voir les résultats, voici la marche à suivre :

→ Dans le fichier"Parametrage.csv", renseigner les paramètres que vous souhaitez :

- typeChampionnat : 0 pour les championnats simples, 1 pour les championnats compelexes
- nbGen : le nombre de génération souhaitées.
- nbIndiv : le nombre d'individus par population.
- Le type d'évaluation : 0 pour une évaluation uniquement sur la distance, 1 pour l'équité sportive et 2 pour une évaluation mixte (moyenne pondérée)
- pourcentageDistance : sert de coefficient lorsque l'évaluation porte sur la moyenne pondérée. Le pourcentage de l'équité sportive est donné par la formule 100 - pourcentageDistance

→ Cliquer sur l'exécutable"Championnat.jar" pour lancer le traitement.

→ Les résultats correspondant sont notifiés dans un fichier texte créé dans le dossier "Resultats".

Java étant une technologie multi-plateforme, le .jar devrait nécessairement fonctionner sur votre machine, sinon n'hésitez pas à le recompiler en ajoutant nos sources dans votre IDE préféré en créant un jar utilisant le seul main disponible (MainPrincipal.java).

Présentation technique du projet

Notre programme est composé de plusieurs packages :

Mains

Ce package contient notre classe main qui créée une matrice, récupères les options du fichier de paramétrage, fait un algorithme génétique et écrit les résultats dans le fichier texte.

Matrice

Matrice.java : permet de transformer le fichier csv contenant la matrice en objet

FonctionsTransverses.java : Qui contient une méthode permettant de tirer aléatoirement un nombre d'équipe différentes passé en paramètre.

Paramétrage

CSVFile.java : Ce fichier contient les noms du ficher de paramétrage et du fichier de matrice.

Options.java : Ce fichier permet de lire le fichier de paramétrage et creer un objet regroupant toutes ces options.

Championnats

Championnat.java : Interface d'un championnat, impose les getter de toutes les informations.

ChampionnatComplexe.java : Un championnat à deux groupes de six équipes, eux memes séparés en groupes de trois. En plus des méthodes de l'interface *Championnat.java* cette classe propose deux constructeurs, un constructeur aléatoire et un constructeur à partir d'un tableau d'entiers.

ChampionnatSimple.java : Un championnat simple a deux poules de six équipes implémentant les méthodes de l'intreface et deux constructeurs, un aléatoire et un à partir d'un tableau d'entiers.

Les championnats sont composés de tableaux contenant le classement des équipes et d'un objet option.

Chaque championnat comporte aussi son équilibre des poules et la distance totale parcourue par toutes les équipes.

Pour comparer deux championnats, l'équilibre des poules et la distances sont ramenés en pourcentage, une moyenne pondérée, dont vous pouvez choisir les coefficients, est calculée. Les championnats implémentent l'interface *Comparable* et une méthode *compareTo()*. Cette

méthode effectue la comparaison entre deux championnat en fonction de votre choix dans le fichier de paramétrage. (Au choix : la distance, l'équité sportive ou une moyenne pondérée.)

Populations

Population.java : Interface d'un championnat, impose les méthodes : évaluation, SelectionAndMutation et un getter pour obtenir un championnat particulier.

PopulationSimple.java : Une classe implémentant l'interface *Population.java* pour le cas simple.

PopulationComplexe.java : Une classe implémentant l'interface *Population.java* pour le cas complexe.

Une population est une arrayListe de championnats tirés aléatoirement dont la taille a été définie dans le fichier de paramétrage.

Les populations implémentant l'interface *Comparable* et notre algorithme génétique ayant une sélection élitiste, la méthode *evaluation()* classe la population en fonction du choix de comparaison entre deux championnats (la distance, l'équité sportive ou une moyenne pondérée).

La méthode *SelectionAndMutation()* créée une nouvelle population a partir de celle donnée en paramètre. Pour cela elle garde les trente-cinq pourcent des meilleurs individus de la population, fais muter les trente-trois pourcent d'individus suivant de façon aléatoire et tire les trente-deux derniers pourcent de championnats manquant.

Mutations

MutationsComplexe.java : Cette classe gère trois différentes méthodes de mutations pour les championnats complexes :

- Une mutation simple qui échange aléatoirement une équipe de chaque groupe de six équipes
- Une mutation simple qui échange aléatoirement une équipe entre deux groupes de trois du même groupe de six équipes
- Une mutation aléatoire qui est utilisée dans la méthode SélectionAndMutation pour les championnats complexes. Cette mutation effectue aléatoirement entre une et six mutations qui sont aléatoirement une des deux mutations décrites perécèdemment.

MutationSimples.java : Cette classe gère trois différentes méthodes de mutations pour les championnats simples :

- Une mutation simple qui échange aléatoirement une équipe de chaque poule
- Une mutation multiple qui fait un nombre donné de mutations simples aléatoires
- Une mutation aléatoire qui est utilisée dans la méthode SélectionAndMutation pour les championnats complexes. Cette mutation aléatoire effectue aléatoirement entre une et

six mutations qui sont aléatoirement une des deux mutations décrite précèdement, dans le cas d'une mutation multiple le nombre de mutation est le même que le numéros de la mutations appliquée.

Documentation du projet

Notre projet comporte divers dossiers et fichiers :

- Deux fichiers .csv : *Parametrage.csv* contenant simplement les paramètres pris en compte dans le programme et *MatriceDeDistance.csv* contenant les distances entre les villes.
- Un dossier Resultats pour stocker les fichiers de résultats.
- Un exécutable *Championnat.jar* qui lancera le programme.
- Un dossier doc regroupant la Javadoc du programme. (Cliquer sur index.html pour voir la documentation des classes et interfaces sous forme d'arborescence.)
- Un dossier src où sont placés les sources du projet. (Le contenu de ce dossier est décrit précédement)

Difficultés rencontrées

D'un point de vue technique nous n’avons pas rencontré de difficultés techniques, nous avions déja des bases en java et nous nous sommes appuyés sur l’API pour l’implémentation du programme.

D'un point de vue relationnel, nous avons eu quelques soucis pour s'organiser, trouver un rythme de travail similaire, se tenir au courant du travail de chacun, dû notamment à nos emplois du temps respectifs assez différents.

Malgré tout cela, nous avons réussi à nous coordonner suffisamment pour appréhender au mieux le projet et ainsi produire un résultat qui satisfait les objectifs demandés.

Webographie

- *Les algorithmes génétiques. (* http://khayyam.developpez.com/articles/algo/genetic/ *)*