TP1 Algorithme génétique appliqué au Knapsack Problem

Objectif

Ce TP vise à résoudre le problème du sac à dos connu sous le nom KP (Knapsack Problem) à l'aide des algorithmes génétiques. Voici une description simple du problème :

Un voleur entre dans une maison en portant un sac à dos qui peut porter un certain poids (30 kg). La maison dispose de 10 objets de valeurs différentes, chacun ayant un poids. Le dilemme du voleur est alors de faire une sélection d'objets qui maximise la valeur totale des objets volés sans dépasser le poids du sac à dos.

Il existe plusieurs applications du problème du sac à dos. Il est particulièrement utilisé dans la gestion du portefeuille. Par exemple, avec un budget limité (capacité du sac), le problème consiste à acheter les titres qui maximisent les gains (valeurs des objets) avec une contrainte de diversification, afin de réduire les risques. Il est appliqué au chargement des moyens de transport mais aussi aux problèmes de découpe.

Description

Considérons le problème dans sa version originale. Nous souhaitons calculer la composition des objets qui permet de maximiser un gain donné dans un sac à capacité limitée. Pour ce faire, un algorithme génétique a été développé (voir le fichier GAexemple_Knapsack.py[1]).

Il est défini de la manière suivante :

1- Population : La population est une matrice binaire où chaque ligne est considérée comme solution au problème (individu). Lorsqu'un élément de la matrice dans la ligne i et dans la colonne j est égal à 1, ceci signifie que la $i^{ème}$ solution pose le $j^{ème}$ objet dans le sac.

2- La fitness (cal_fitness) : Le codage des solutions ne garantit pas la faisabilité de la solution, à savoir que la somme des poids des objets sélectionnés par une solution donnée peut être supérieure au poids maximal. Ainsi, l'évaluation d'une solution (fitness) est égale la somme des valeurs des objets considérés par la solution, seulement si le poids total sera inférieur à la limite physique du sac. Dans le cas contraire, la valeur fitness de la solution est considérée nulle, puisque le sac se déchire.

3- Sélection : Elle consiste à prendre seulement les meilleures solutions.

4- Croisement : Il consiste à choisir parmi les solutions sélectionnées deux parents pour générer une solution. Les deux parents peuvent engendrer un enfant avec une certaine probabilité (taux_croisement). L'enfant hérite de la moitié droite du parent 1 et de la moitié gauche du parent2.

5- Mutation : Elle est appliquée aux enfants pour générer un mutant. Chaque enfant a une chance d'être muté (taux_mutation). La mutation consiste à inverser un bit de la solution.

En plus de l'algorithme génétique, le fichier GAexemple_Knapsack.py génère automatiquement des instances aléatoires du problème de sac à dos, à savoir des données relatives aux poids et aux valeurs des objets.

Questions

1- Faites varier le nombre d'objets, leurs poids, la capacité maximale du sac et le nombre de

[1] Le programme proposé dans ce TP est à titre indicatif pour pouvoir entamer le problème rapidement. Ce programme peut donc être optimisé.

générations. Que constatez-vous ? Expliquez les causes.

2- Apporter les modifications nécessaires pour aboutir seulement à des solutions faisables, si elles existent. Voici les approches qu'il faut coder et comparer :

   a. Mettre une fitness négative proportionnelle au dépassement de la capacité du sac.

   b. Vérifier lors de la génération des solutions, du croisement et de la mutation que ces solutions respectent la capacité du sac.

   Pour comparer les deux approches, choisir intelligemment 4 types d'instances du problème et stocker les dans des fichiers csv. Comparer les deux approches en termes de qualité de résultats et de temps de calcul.

3- Nous souhaitons déployer cet algorithme pour la gestion du portefeuille[2]. Pour des raisons de minimisation du risque, nous considérons que le client peut acheter plusieurs parts du même titre dans la mesure où il respecte la limite des 20% du budget total. Par exemple, si la part du titre « 0 » vaut 20€ et que le budget total à disposition est de 1000€, la solution ne doit pas contenir plus de 10 parts du titre « 0 ». Chaque titre dispose d'une estimation du gain par part. A la suite des estimations des gains escomptés de chaque titre, l'objectif est d'augmenter les gains en respectant les limites de son budget et la contrainte de minimisation de risque.

   a. A la suite de la question 2, choisir la version la plus performante en termes de résultats, et adapter le programme à la gestion du portefeuille.

4- Choisir au moins 4 instances des données du problème et stocker les dans des fichiers csv. Améliorer les performances de l'algorithme génétique sur ces quatre instances. Décrivez dans le rapport les améliorations réalisées et les motivations.

[2] Le problème de gestion de portefeuille est nettement plus complexe dans la réalité que la description donnée dans le sujet. Par exemple, il peut résolu en multicritères, pour la minimisation des risques avec prise en compte de la volatilité des titres. Actuellement, l'apprentissage machine appliqué à la gestion de portefeuille est un sujet de recherche en pleine croissance.