INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Croisement de chemins optimaux dans un graphe avec un algorithme génétique

BE Graphes - Problème ouvert

26 avril 2023

Auteurs

Marti JIMENEZ

Rémi JACQUEMIN

Problème d’échange de colis :

On travaille sur un graphe avec des arcs pondérés représentant des plans de villes ou pays avec des routes. Chaque arc représente une route et sa valeur représente sa longueur ou son temps de parcours en fonction du mode (Temps|Distance).

Ce problème consiste en deux robots R1D1 et R2D2 qui ont chacun une origine et une destination. Ces deux robots veulent s'échanger un colis. On doit trouver le chemin le plus court pour les deux robots afin que l’échange puisse avoir lieu.

On cherche ici à minimiser la somme des temps de trajets des deux robots. La solution obtenue doit permettre de déterminer le point de rencontre des deux robots et de reconstituer les trajets effectués.

Partie I : Algorithme proposé

Explication de notre algorithme :

Notre solution pour ce problème est de faire un algorithme génétique qui est une méthode d'optimisation inspirée de l'évolution biologique.

Notre approche du problème est de trouver le point de croisement optimal. Avec celui-ci nous pourrons retrouver les chemins à suivre en effectuant 4 algorithmes de A*, chacun partant du point solution trouvé et allant respectivement vers l'origine 1, origine 2, destination 1 et destination 2.

Tout d'abord on va avoir une population initiale (génération 0) constituée de solutions aléatoires (nœuds aléatoires). On va ensuite évaluer la qualité de chaque solution et créer une nouvelle génération en utilisant les meilleurs de la génération précédente et de nouvelles solutions issues de la mutation des meilleures solutions.

Ce processus de sélection et mutation répété sur plusieurs générations va permettre à la population de converger vers une solution optimale.

Cet algorithme peut être adapté en ajustant les paramètres tels que la taille de la population, le taux de mutation, la fonction d'évaluation, la sélection des survivants, etc. Afin d'obtenir de meilleures performances.

Quelques outils pratiques :

Tirage aléatoire : Afin de ne pas tirer aléatoirement des points déjà visités nous allons créer au début de l'algo un tableau contenant des int allant de 1 à n-1. Puis une variable int taille pour stocker la taille du tableau. A chaque fois que nous allons tirer un nœud aléatoirement il suffira de faire un pop du tableau à un indice aléatoire compris entre 0 et taille. Une fois le pop réalisé nous pourrons diminuer la taille. On récupérera ainsi un id de noeud pas déjà tiré.

Mémoire de visite : Afin de ne pas évaluer deux fois le même nœud nous allons ajouter un boolean (boolean vue) à la classe label qui deviendra true quand le nœud sera évalué avec la fonction fit.

Comme tous les noeuds ne sont pas tirés au hasard, pour éviter de mettre à jour le tableau de noeuds tirés à chaque ajout de noeud dans une population nous allons, en cas de tirage aléatoire d'un noeud déjà vu (vue==true), effectuer un nouveau tirage et jeter ce noeud. On aura donc mis à jour le tableau en coût O(1) pour le nœud tiré, au lieu de coût O(log(taille)) si on avait dû le chercher pour le supprimer au moment de l'inclusion dans une population.

Classement de la population : Nous allons utiliser un tas binaire pour classer la population en fonction de leur score (sortie de la fonction fit). Ceci nous permet d'optimiser le coût des insertions.

Comme nous ne gardons que les 20 meilleurs de chaque génération nous allons garder en mémoire le score du 20ème nœud présent dans le tas binaire. Si lors de l'évaluation d'un nœud son score est inférieur à celui-ci on pourra directement le jeter sans avoir à l'ajouter dans le tas.

Etape 1 : Population initiale

Pour notre population initiale nous allons choisir 100 nœuds du graphe tirés aléatoirement.

Etape 2 : Itération de génération : Sélection, mutation (fonction fit, repeuplement)

Fonction fit : Notre méthode d'évaluation (attribution d'un score) pour un noeud P est la suivante; nous allons effectuer 4 algorithmes de A* partant chacun de P et allant respectivement vers l'origine 1, origine 2, destination 1 et destination 2, puis ajouter les distances (ou temps) trouvés pour les 4 chemins. Ceci nous donnera la distance minimale à parcourir (ou le temps minimal) afin que les deux robots se croisent au nœud P.

Sélection : Après avoir évalué et classé toute la population. Nous allons garder les 20 nœuds avec les scores les plus élevés. Grâce à notre implémentation du tas nous allons juste avoir à prendre les nœuds présents dans le tas.

Création de la nouvelle population : Après la sélection nous allons créer la nouvelle génération de nœuds. Celle-ci consistera en:

- Les 20 meilleurs noeuds de la génération précédente
- Tous les successeurs de ces 20 noeuds (on remarque que le nombre de noeuds d'une population peut varier et être plus grand que 100, ceci ne pose pas de problème car seul les 20 meilleurs noeuds seront sélectionnés et donc notre population ne va pas exploser exponentiellement)
- 10 noeuds tirés aléatoirement (pour sortir des possibles minima locaux)

Si après ces ajouts notre population est inférieure à 100. On comble la différence en tirant des nœuds aléatoires. Pour ne pas avoir de générations trop peu peuplées.

Etape 3 : Critères d'arrêt

Nous allons implémenter plusieurs critères d'arrêts et ceux-ci seront à tester et vérifier expérimentalement afin de les ajuster au mieux.

Critère 1 : Tous les nœuds ont été vus (tableau vide) dans ce cas on retourne le meilleur nœud du tas. Le nœud trouvé est l'optimal.

Critère 2 : Le meilleur nœud du tas n'a pas changé depuis X (exemple X=100) générations. Dans ce cas là on retourne ce nœud. On considère qu'on à trouvé la meilleure solution avec une certaine probabilité.
Critère 3 : La distance (ou temps) totale du chemin du meilleur nœud n'a pas varié de plus de X% depuis Y générations (exemple X=0.1% Y=100). Dans ce cas là, on retourne le premier nœud du tas. On considère alors que la solution trouvée est "assez proche" de l'optimale.

Partie II : Optimalité et améliorations

Optimalité de l'algorithme génétique :

Utiliser un algorithme génétique pour résoudre ce problème offre plusieurs avantages. Grâce à son approche basée sur l'évolution et l'ajout d'aléatoire dans chaque génération, il permet d'explorer une grande surface très rapidement et de trouver des solutions optimales potentielles en quelques itérations.

Cependant, à moins de visiter tous les nœuds, cet algorithme ne garantit pas l'optimalité de la solution. On aura une solution optimale avec une certaine probabilité mais à moins de parcourir tous les nœuds, nous ne pouvons pas être sûrs de l'optimalité de la solution trouvée.

Améliorations possibles de l'algorithme génétique :

Bien que l'algorithme génétique puisse être efficace pour résoudre le problème d'échange de colis, certaines améliorations peuvent être envisagées pour optimiser encore davantage ses performances. Voici quelques idées d'améliorations potentielles :

Optimisation des paramètres de l'algorithme : Les performances de l'algorithme génétique dépendent fortement des paramètres tels que la taille de la population, le nombre de sélectionnés, la génération de la nouvelle population, les critères d'arrêt. Une recherche des meilleures valeurs de ces paramètres peut permettre d'améliorer la qualité des solutions obtenues.

Stratégies de sélection : Notre algorithme utilise une sélection assez simple qui consiste à prendre les meilleurs selon la fonction fitness de chaque génération. Cependant il existe d'autres méthodes de sélection qui pourraient être plus efficaces comme par exemple la sélection par rang (attribution d'un rang en fonction du score fitness et attribution de probabilité de reproduction plus ou moins élevée selon le rang) ou la sélection par domination (une solution B est dominée par A si A est meilleure que B dans au moins un critère et qu'elle n'est pas pire que B dans aucun des autres critères, on choisit de reproduire les solutions non dominées).

Ces différents types de sélections peuvent éviter la convergence prématurée vers un optimum local.

Méthodes d'initialisation avancées : L'utilisation de méthodes d'initialisation avancées, comme l'initialisation aléatoire guidée par les informations du problème, peut permettre de générer une population initiale de meilleure qualité et d'accélérer la convergence de l'algorithme.

En conclusion on peut donc dire que notre algorithme génétique apporte une approche qui semble efficace pour résoudre ce problème mais que de nombreuses améliorations peuvent être envisagées pour optimiser ses performances.

Partie III : Code et amplification à n robots

L'implémentation du code ce trouve sur le repo Github suivant :
https://github.com/Marti2405/BE_graphes
(accès accordé aux comptes : anlentz et carlajuvin)

Amplification à n robots :
Notre algorithme peut se généraliser à n robots devant tous se rencontrer en même temps en un point de leur trajet. Il suffit d'ajouter dans le calcul du score d'un noeud de la fonction fit la distance vers les autres origines et destinations possibles (aussi calculées avec A*). Comme ce qu'on cherche à minimiser est la distance (ou temps) globale, notre algorithme peut se généraliser à n robots juste en rajoutant les origines et destinations supplémentaires.