Jules DAROLD
Alexandre LAVAL
Feng ZHIXUAN

22/01/202

Team Orienteering Problem

Algorithme génétique et Particle Swarm Optimization

Introduction:

Le Team Orienteering Problem (TOP) est un problème de planification de tournées qui consiste à planifier une tournée pour un groupe de véhicules ou d'individus qui doivent visiter un certain nombre de sites dans un ordre spécifique, en minimisant la distance totale parcourue et en respectant les contraintes de temps de visite et de capacité. Ce problème est largement utilisé dans les domaines de la logistique, de la planification des tournées de véhicules et de la planification des opérations.

Il existe plusieurs méthodes pour résoudre le TOP, chacune ayant ses avantages et ses inconvénients en fonction des besoins spécifiques de chaque application. Les algorithmes de programmation linéaire, les algorithmes génétiques, ou les algorithmes de recherche tabou sont quelques-unes des méthodes couramment utilisées. Dans ce rapport, nous nous concentrons sur l'utilisation de l'algorithme génétique (AG) pour résoudre le TOP.

Dans ce rapport, nous décrirons en détail la méthodologie utilisée pour résoudre le TOP en utilisant l'algorithme génétique, et mettrons en perspective les résultats obtenus. Nous montrerons également comment cette méthode peut être appliquée à des cas d'utilisation réels pour résoudre des problèmes de planification de tournées similaires.

I Algorithme génétique

L'algorithme génétique (AG) est une heuristique de recherche qui imite le processus naturel de l'évolution tel qu'il se produit pour toutes les espèces d'êtres vivants. Cette méthode utilise des techniques inspirées de la nature telles que la mutation, le croisement, l'héritage et la sélection pour générer des solutions pour des problèmes d'optimisation. Le succès d'un AG dépend du type et de la complexité du problème auquel il est appliqué. Dans un AG, les chromosomes ou individus sont représentés sous forme de chaînes qui codent des solutions candidates pour un problème d'optimisation, qui évoluent ensuite vers des solutions meilleures. Le processus évolutif d'un GA commence par l'initialisation d'une population de solutions (généralement de manière aléatoire), qui évoluent et s'amélioreront au cours de trois étapes principales :

- La sélection: une portion de chaque génération successive est sélectionnée, en fonction de leur “fitness” , afin d'engendrer la génération suivante, avec probablement une meilleure fitness.
- La reproduction: Les individus sélectionnés produisent la génération suivante en passant par des processus de mutation et d'hybridation, cherchant récupérer et transmettre par héritages les traits des individus les plus performants.
- Terminaison: après un certain nombre de générations, l'algorithme s'arrête.

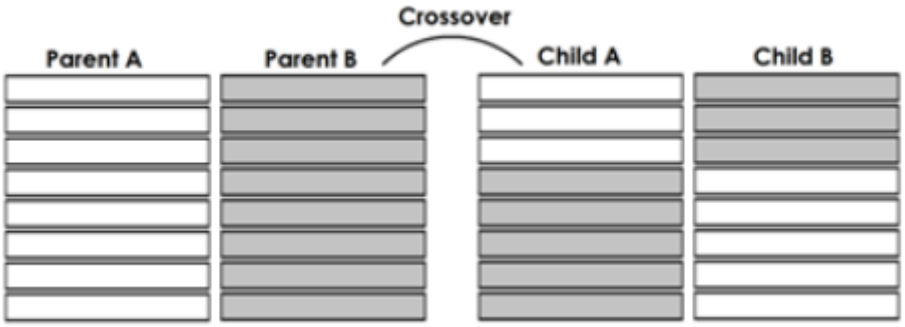
Dans le cadre de notre problème, nous considérons l'espace d'état comme un graphe dont les nœuds correspondent aux balises à récupérer, avec comme valeur leur score, et les arcs les distances euclidiennes entre chaque nœud. Une solution est donc une liste L composée de n liste, où n est le nombre de véhicules. Chaque liste de L correspond à l'ordre successif des balises atteintes par un véhicule.

1. Fonction de fitness

Dans cette fonction de fitness, la performance de la solution est mesurée en calculant la somme des scores de chaque véhicule dans l'agent. La somme des scores est utilisée comme métrique de performance car elle reflète la qualité de la solution en termes de points d'intérêt visités par les véhicules. Plus les scores des points d'intérêt visités sont élevés, plus la performance de la solution est élevée.

2. Hybridation et mutation

La procédure de d'hybridation est réalisée en échangeant des trajets entre deux chromosomes, ce qui entraîne la création de deux nouveaux chromosomes. Les trajets à échanger sont sélectionnés aléatoirement, mais des blocs entiers de trajets consécutifs sont copiés sur les nouveaux chromosomes. Ci dessous voici un exemple d'hybridation entre 2 parents, résultant en 2 enfants



Plusieurs processus de mutation différents sont mis en oeuvre:

- La modification d'un des véhicules de l'agent choisit aléatoirement en échangeant l'ordre de deux nœuds consécutifs.
- La modification un des véhicules de l'agent choisit aléatoirement en déplaçant un nœud à une autre position dans ce véhicule.
- La modification d'un des véhicules de l'agent choisit aléatoirement, en retirant puis en insérant des nœuds aléatoires dans ce véhicule tant que c'est possible sans dépasser la distance parcourue maximale.

II Exploration de la méthode Particle Swarm Optimization

Particle Swarm Optimization (PSO) est un algorithme d'intelligence d'essaim proposé par Kennedy et Eberhart avec l'idée de base de simuler le comportement collectif des animaux sauvages dans la nature.

Le PSO a d'abord été utilisé pour des problèmes d'optimisation dans l'espace continu comme suit. Un ensemble connu sous le nom d'essaim de solutions candidates, appelé particules, est composé de positions dans l'espace de recherche. L'essaim explore l'espace de recherche selon les équations (1) et (2). Dans ces équations, x_i^t et v_i^t sont respectivement les vecteurs de position et de vitesse de la particule i à l'instant t. Trois valeurs w, c1 et c2, appelées respectivement inertie, facteur cognitif et facteur social, sont des paramètres de l'algorithme. Deux valeurs r1 et r2 sont des nombres aléatoires générés dans l'intervalle [0,1]. Chaque particule i mémorise sa position la plus connue jusqu'à l'instantané t comme x_i^{lbest}, et la meilleure (1) (2) position connue jusqu'à l'instantané t pour l'essaim est notée comme x_i^{gbest}.

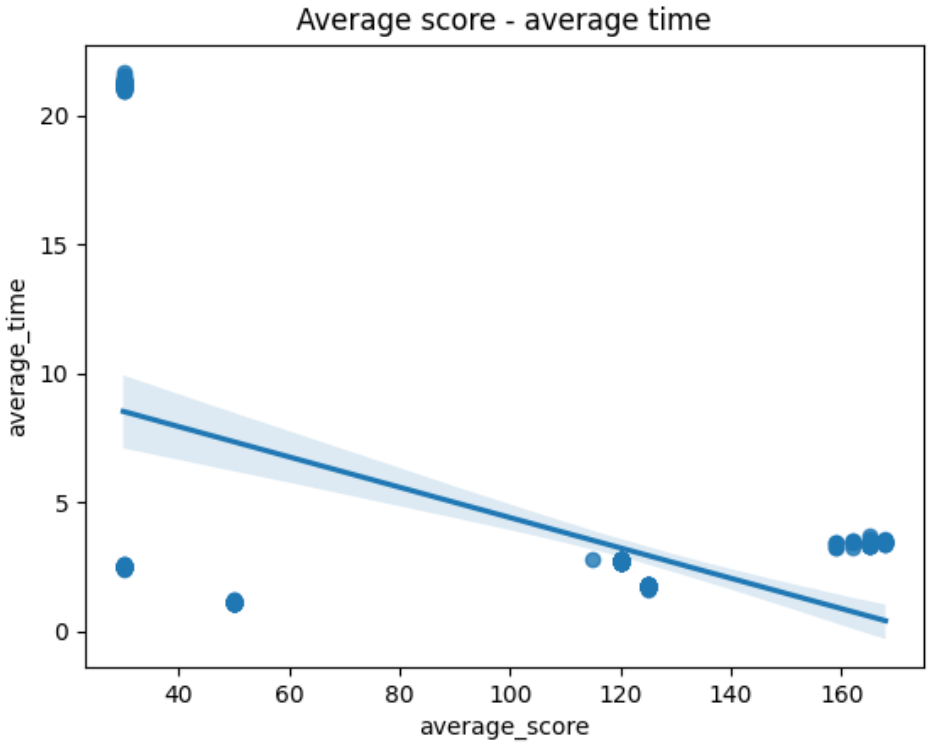
$$v_i^{t+1} = w \cdot v_i^t + c_1 \cdot r_1 \cdot (x_i^{lbest} - x_i^t) + c_2 \cdot r_2 \cdot (x^{gbest} - x_i^t) \quad (1)$$
$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2)$$

Grâce à cette conception, PSO réussit très bien à effectuer des optimisations dans un espace continu. Basé sur cette idée, on a proposé un algorithme prototype.

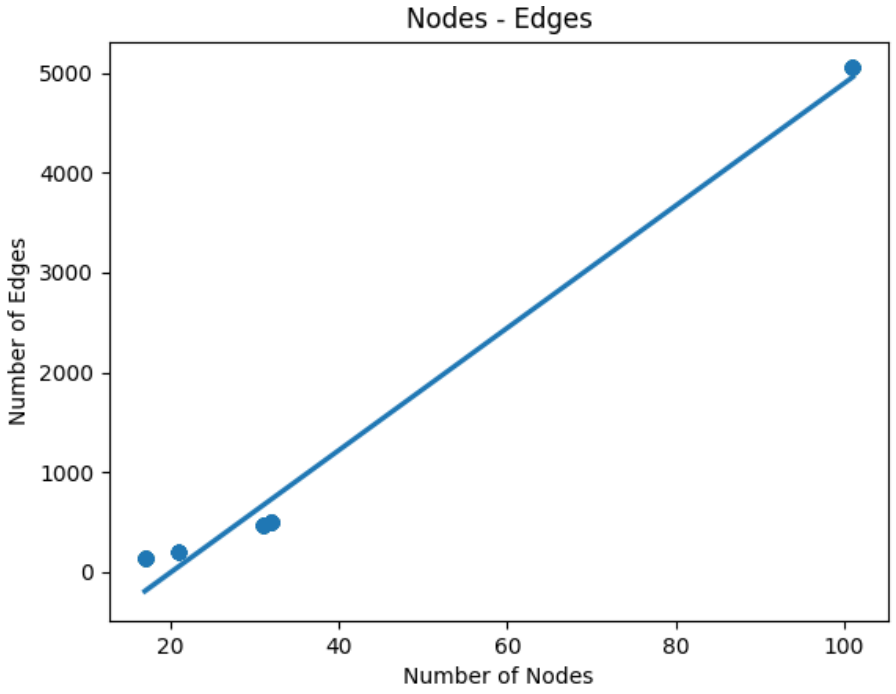
III Analyse des résultats de l'algorithme génétique

Les résultats ont été obtenus pour 327 instances sur les 387 que nous avions récupérées. Chaque instance a été résolue 3 fois puis les mesures faites ont été moyennées afin de couvrir un minimum le bruit statistique des mesures et de l'aléatoire. En effet l'algorithme génétique reposant sur de l'aléatoire ne rend pas les mêmes résultats à chaque itération. Ainsi le score de la solution optimale trouvée peut varier d'une itération à l'autre. L'algorithme utilisé était une implémentation en python de celui présenté précédemment et avait comme paramètres une population initiale de 30 agents qui sera sélectionnée et reproduite sur 80 générations.

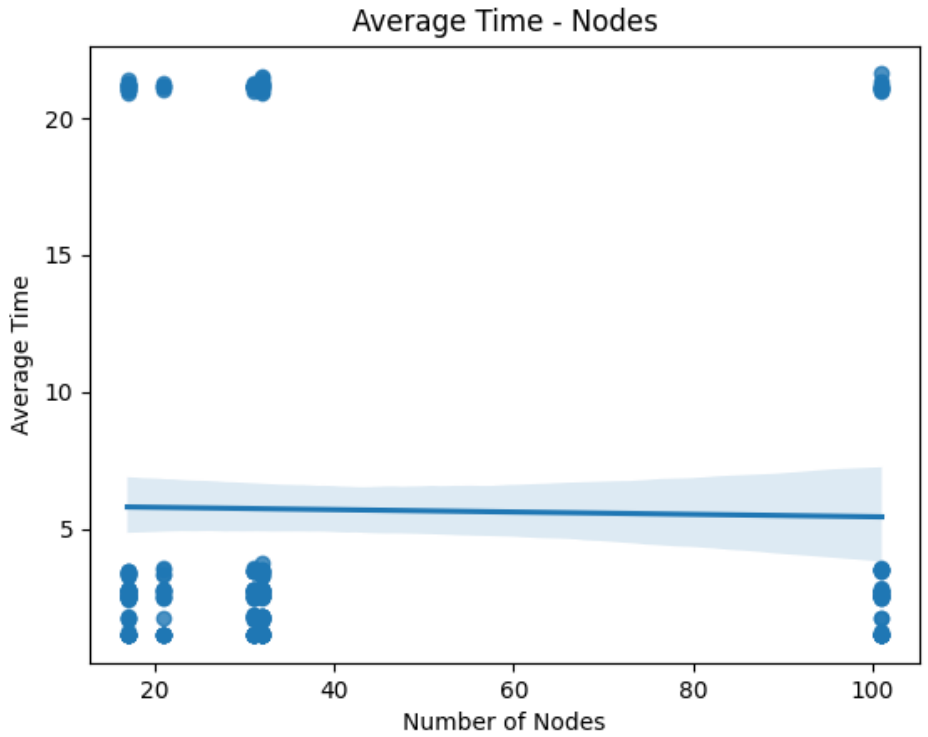
Le temps moyen de résolution est de 5.69 secondes,avec 75% des instances en dessous de 3.5 secondes et un petit nombre bien au-dessus. La standard deviation est à 7.4 secondes et le maximum à 21.6 secondes.



Nous pouvons voir ici que le score moyen et plutôt bien corrélé au temps moyen, avec en haut à gauche du graphique les instances pour lesquelles l'algorithme a clairement rencontré des problèmes, passant bien trop de temps dessus pour ne jamais trouver de solutions satisfaisantes.



Remarque importante: le nombre de nœuds et de liens est très fortement corrélé, cela nous permet d'étudier la corrélation d'autres paramètres avec ces deux là en même temps.



On s'aperçoit ici de l'un des avantages de l'algorithme génétique. Le temps de résolution ne dépend pas du nombre de nœuds (ou de liens). Mais cela montre aussi que l'algorithme génétique peut ne pas réussir à résoudre des instances apparemment assez simples en termes de nombre de nœuds par exemple, avec une performance faible quasi équitablement répartie entre les instances quel que soit leur nombre de nœuds ou de liens.

En conclusion l'algorithme génétique semble être bien adapté dans la plupart des cas. Il aurait été intéressant d'étudier plus en détail les instances pour lesquelles celui-ci ne fonctionne pas bien afin de déterminer plus précisément quel paramètre influence ses performances.

Source:

I Algorithme génétique:

- Ferreira, J., Quintas, A., Oliveira, J.A., Pereira, G.A.B., Dias, L. (2014). Solving the Team Orienteering Problem: Developing a Solution Tool Using a Genetic Algorithm Approach. In: Snášel, V., Krömer, P., Köppen, M., Schaefer, G. (eds) Soft Computing in Industrial Applications. Advances in Intelligent Systems and Computing, vol 223. Springer, Cham. https://doi.org/10.1007/978-3-319-00930-8_32
- Ferreira J., Oliveira J., Pereira G., Dias L., Vieira F., Macedo J., Carção T., Leite T. and Murta D.. Developing Tools for the Team Orienteering Problem - A Simple Genetic Algorithm. DOI: https://doi.org/10.5220/0004273801340140 In Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems (ICORES-2013), pages 134-140 ISBN: 978-989-8565-40-2 Copyright c 2013 SCITEPRESS (Science and Technology Publications, Lda.)

II PSO:

- Dang, Duc-Cuong & Guibadj, Rym & Moukrim, Aziz. (2013). An effective PSO-inspired algorithm for the team orienteering problem. European Journal of Operational Research. 229. 332–344. 10.1016/j.ejor.2013.02.049.