ÉCOLE CENTRALE LYON

Algorithmes collaboratifs et applications

Bureau d'Etude

Problème de Coloriage de graphe et l'algorithme de colonie de fourmis

Auteurs :
M Ilyas DAHAOUI

Encadrants :
M. Alexandre SAIDI

Version du
28 avril 2024

Table des matières

1 Introduction

La coloration de graphes est un problème fondamental en informatique avec de nombreuses applications dans le monde réel, allant de la planification et de l'allocation de ressources à l'optimisation de réseaux et à l'allocation de registres dans les compilateurs. L'objectif est de trouver une façon d'assigner des couleurs aux sommets d'un graphe de telle sorte que deux sommets adjacents n'aient pas la même couleur.

Les approches traditionnelles de coloration de graphes impliquent souvent des algorithmes heuristiques ou des méthodes exactes, qui peuvent avoir du mal avec des graphes volumineux ou des instances NP-difficiles. L'Optimisation par Colonie de Fourmis (OCF) offre une alternative prometteuse inspirée par le comportement de recherche de nourriture des fourmis.

Dans ce projet, nous explorons l'application de l'OCF au problème de coloration de graphes. L'OCF est une technique d'optimisation métaheuristique qui imite le comportement de recherche de nourriture des fourmis pour résoudre efficacement des problèmes d'optimisation combinatoire. En exploitant l'intelligence collective d'une population de fourmis artificielles, les algorithmes OCF recherchent de manière itérative des solutions de haute qualité.

2 Modélisation du problème

Soit $G = (V, E)$ un graphe non orienté, où V est l'ensemble des nœuds et E l'ensemble des arêtes.

L'objectif est de trouver une coloration des nœuds du graphe G en utilisant un nombre minimal de couleurs, tout en s'assurant que deux nœuds adjacents n'aient pas la même couleur.

2.1 Notations

— $N = |V|$: nombre de nœuds dans le graphe
— C : ensemble des couleurs disponibles
— α : paramètre d'importance relative de la phéromone
— β : paramètre d'importance relative de la valeur heuristique
— ρ : taux de décroissance de la phéromone

2.2 Initialisation

1. Initialiser la matrice de phéromones $= (\tau_{ij})$ où $\tau_{ij} = 1$ si i et j ne sont pas adjacents, 0 sinon.
2. Initialiser la coloration des nœuds de manière aléatoire.

3 Algorithme

Répéter jusqu'à ce que le critère d'arrêt soit atteint :

1. Créer une colonie de m fourmis.
2. Pour chaque fourmi k :
   (a) Initialiser la fourmi k avec un nœud de départ et une coloration partielle.
   (b) Tant que tous les nœuds ne sont pas coloriés :
      i. Calculer la valeur heuristique η_{ij} pour chaque nœud j non colorié, en fonction du degré de saturation du nœud.
      ii. Calculer la probabilité de transition p_{ijk} pour chaque nœud j non colorié, en fonction de la phéromone τ_{ij} et de la valeur heuristique η_{ij}.
      iii. Choisir le prochain nœud à colorier selon la probabilité de transition p_{ijk}.

iv. Assigner une couleur disponible au nœud choisi.

(c) Calculer le nombre de couleurs utilisées par la fourmi k.

3. Sélectionner la meilleure solution (coloration) parmi les fourmis.
4. Mettre à jour la matrice de phéromones en fonction de la meilleure solution.
5. Appliquer le taux de décroissance ρ à la matrice de phéromones .

Formulation mathématique

1. Valeur heuristique η_{ij} :

$$\eta_{ij} = \text{dsat}(j)^{\beta}$$

où $\text{dsat}(j)$ est le degré de saturation du nœud j, c'est-à-dire le nombre de couleurs utilisées par les voisins de j.

2. Probabilité de transition p_{ijk} :

$$p_{ijk} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N_k} (\tau_{il}^{\alpha} \cdot \eta_{il}^{\beta})}$$

où N_k est l'ensemble des nœuds non coloriés pour la fourmi k.

3. Mise à jour de la matrice de phéromones :

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}$$

où $\Delta\tau_{ij} = 1$ si les nœuds i et j ont la même couleur dans la meilleure solution, 0 sinon.

Ce processus est répété jusqu'à ce que le critère d'arrêt soit atteint, par exemple un nombre maximal d'itérations ou une solution satisfaisante.

4 Simulation :

Classe	Explication
`Ant`	Cette classe représente une fourmi dans l'algorithme de colonie de fourmis. Elle contient les attributs et méthodes nécessaires pour effectuer le coloriage du graphe : — `initialize(self, g, colors, start=None)` : initialise la fourmi avec le graphe `g`, l'ensemble des couleurs `colors` et un nœud de départ `start` (choisi aléatoirement si non spécifié). — `assign_color(self, node, color)` : assigne une couleur `color` au nœud `node`. — `colorize(self)` : effectue le coloriage du graphe en choisissant les prochains nœuds à colorier de manière probabiliste. — `dsat(self, node=None)` : calcule le degré de saturation du nœud `node` (ou du nœud de départ si non spécifié). — `si(self, node, adj_node)` : calcule la valeur de phéromone entre le nœud `node` et le nœud adjacent `adj_node`. — `next_candidate(self)` : choisit le prochain nœud à colorier de manière probabiliste. — `pheromone_trail(self)` : calcule la matrice de phéromones en fonction de la coloration obtenue. — `colisions(self)` : calcule le nombre de conflits (nœuds adjacents de même couleur) dans la coloration.
`create_graph(path)`	Cette fonction lit un fichier de graphe et crée un objet `networkx.Graph` correspondant.
`draw_graph(g, col_val)`	Cette fonction dessine le graphe `g` en utilisant les couleurs spécifiées dans `col_val`.
`init_colors(g)`	Cette fonction initialise l'ensemble des couleurs disponibles pour le coloriage du graphe `g`.
`init_pheromones(g)`	Cette fonction initialise la matrice de phéromones pour le graphe `g`.
`adjacency_matrix(g)`	Cette fonction calcule la matrice d'adjacence du graphe `g`.
`create_colony()`	Cette fonction crée une colonie de fourmis pour résoudre le problème de coloriage.
`apply_decay()`	Cette fonction applique le taux de décroissance des phéromones à la matrice de phéromones.
`update_elite()`	Cette fonction sélectionne la meilleure solution (coloration) parmi les fourmis et met à jour la matrice de phéromones en conséquence.
`solve(input_graph, num_ants=10, iter=10, a=1, b=3, decay=0.8)`	Cette fonction résout le problème de coloriage de graphe en utilisant l'algorithme de colonie de fourmis. Elle prend en entrée le graphe à colorier, le nombre de fourmis, le nombre d'itérations, les paramètres `alpha` et `beta`, ainsi que le taux de décroissance des phéromones. Elle retourne le coût de la solution optimale, la solution optimale (coloration des nœuds) et le nombre d'itérations nécessaires.

TABLE 1 – Description des classes et fonctions de l'algorithme de colonie de fourmis

5 Résultat

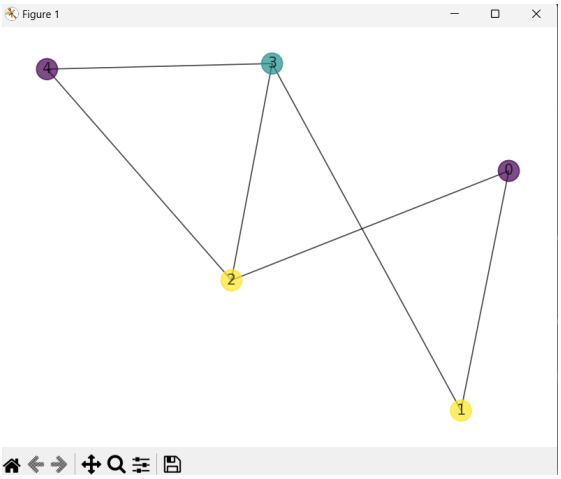

FIGURE 1 – exemple de coloriage du graphe par l'ACO

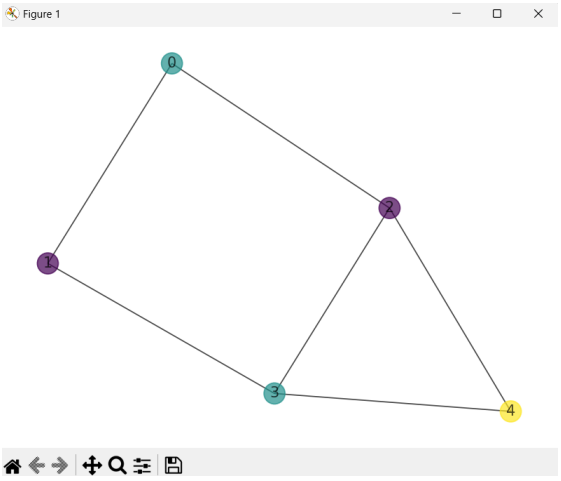

FIGURE 2 – exemple de coloriage du graphe par l'ACO

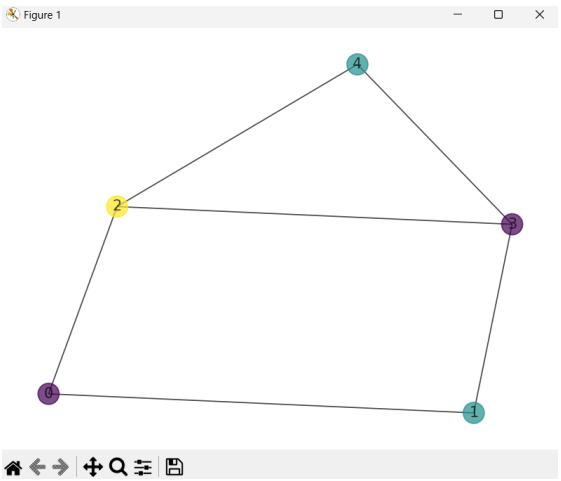

FIGURE 3 – exemple de coloriage du graphe par l'ACO

Pour tester la robustesse de l'algorithme, on va ajouter des arêtes supplémentaires au graphe afin d'accroître sa complexité. Cela nous permettra d'évaluer comment l'algorithme se comporte face à un graphe plus dense et plus interconnecté.

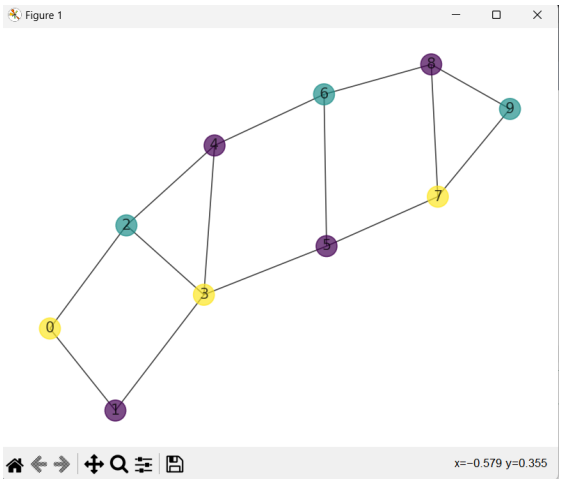

FIGURE 4 – exemple de coloriage du graphe par l'ACO

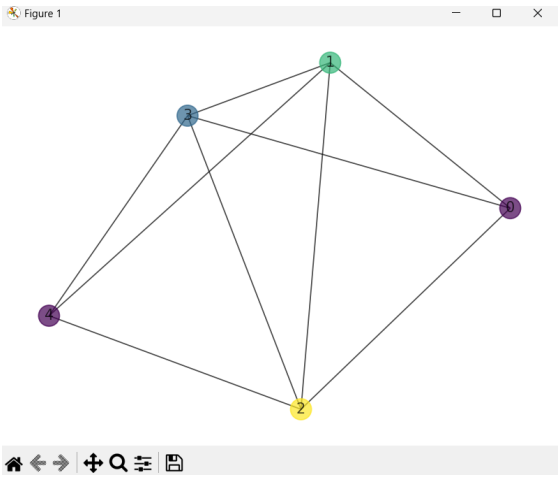

FIGURE 5 – exemple de coloriage du graphe par l'ACO

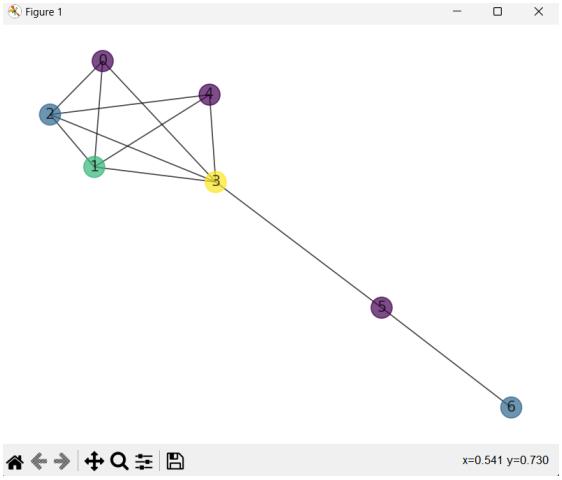

FIGURE 6 – exemple de coloriage du graphe par l'ACO

6 Conclusion

ce projet m'a permis de mettre en pratique de manière concrète les connaissances et les compétences acquises tout au long de ce cours, L'algorithme de colonie de fourmis s'est avéré être une approche efficace pour résoudre le problème de coloriage de graphe. Les résultats obtenus montrent que l'algorithme est capable de trouver des solutions optimales ou proches de l'optimal, en utilisant un nombre minimal de couleurs. Les principaux avantages de cette méthode

sont : Adaptabilité : L'algorithme s'adapte bien à différentes instances du problème de coloriage de graphe, en explorant efficacement l'espace de recherche. Robustesse : Grâce au mécanisme de phéromones, l'algorithme est capable d'éviter les optimums locaux et de converger vers des solutions de bonne qualité. Simplicité d'implémentation : Le code fourni illustre la relative simplicité de mise en œuvre de l'algorithme de colonie de fourmis pour le problème de coloriage de graphe. Cependant, certains aspects peuvent être améliorés, comme l'ajustement fin des paramètres (α, β, ρ) ou l'hybridation avec d'autres techniques d'optimisation.

Mots clés : Coloriage de graphe , Algorithme de colonie de fourmis.

École centrale de Lyon
36, Avenue Guy de Collongue
69134 Écully

7 Réferences

Shtovba, S. D. (2005). Ant algorithms : Theory and applications. Programming and Computer Software, 31(4), 167-178.

Cui, G., Qin, L., Liu, S., Wang, Y., Zhang, X., & Cao, X. (2022). Modified PSO algorithm for solving planar graph coloring problem. Applied Soft Computing, 120, 108677.

Dorigo, M., & Stützle, T. (2004). Ant Colony Optimization. MIT Press.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system : optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(1), 29-41.