Recherche arborescente Monte-Carlo pour la coloration de graphe pondéré

Cyril Grelier[1] Olivier Goudet[1] Jin-Kao Hao[1]

LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France
{cyril.grelier,olivier.goudet,jin-kao.hao}@univ-angers.fr

Mots-clés : *Coloration de graphe pondéré, recherche arborescente Monte-Carlo, recherche locale*

1 Introduction

Le problème de coloration pondérée (*Weighted Vertex Coloring Problem* - WVCP) est une extension du problème de coloration de graphes standard (GCP). Étant donné un graphe pondéré $G = (V, E, W)$ où V désigne l'ensemble des sommets, E l'ensemble des arcs et W l'ensemble des poids $w(v)$ associés à chaque sommet $v \in V$, le WVCP consiste à trouver une partition de l'ensemble des sommets V en k différents groupes de couleurs, $S = \{V_1, \ldots, V_k\}$, telle que deux sommets adjacents soient dans des groupes de couleurs différents et que la somme $\sum_{i=1}^{k} max_{v \in V_i} w(v)$ soit minimisée.

Comme le GCP, le WVCP est NP-difficile. Il existe des applications pratiques du WVCP dans différents domaines tels que des problèmes de décomposition de matrices, de planification de tâches ou bien de partitionnement d'items en lots.

Les algorithmes obtenant les meilleurs résultats pour ce problème sont RedLS, un algorithme qui réduit l'espace de recherche avant d'utiliser une recherche locale [3], ILS-TS, un algorithme de recherche locale itérée avec un opérateur grenade [2], ainsi que DLMCOL, un algorithme mémétique couplé avec un réseau de neurones utilisé pour guider le choix des croisements à effectuer dans la population [1].

2 Recherche arborescente Monte Carlo pour le WVCP

Dans ce travail, nous proposons d'étudier l'apport de la recherche arborescente Monte Carlo (MCTS) pour résoudre le WVCP. Le MCTS explore un arbre de recherche et répète itérativement *4 phases*. Lors de la première phase, l'algorithme *sélectionne* quel noeud explorer en partant de la racine de l'arbre jusqu'à une feuille et en considérant un compromis entre exploitation et exploration. Ensuite, lors de la phase d'*expansion*, un noeud fils est ajouté au noeud sélectionné avant de lancer une phase de *simulation* qui complète la solution courante. Enfin, la phase de *mise à jour* incrémente un compteur de visite pour chaque noeud choisi et réactualise la moyenne courante des scores obtenus sur la branche à la fin de la simulation. L'algorithme MCTS utilisé dans ce travail est adapté spécifiquement au problème de coloration pondérée : chaque noeud dans l'arbre correspond à un sommet coloré dans le graphe, l'ordre des sommets colorés à chaque niveau dans l'arbre est prédéfini en amont pour limiter le nombre de choix possibles à chaque itération, les sommets les plus lourds et de degrés les plus élevés sont placés en premier.

Une fois le cadre général de l'algorithme défini, notre travail s'est concentré notamment sur l'étude de l'impact de la phase de *simulation*. Nous avons étudié plusieurs manières de compléter la solution, tout d'abord avec des approches aléatoires puis gloutonnes avant de s'orienter vers une recherche locale itérative inspirée de [2].

Dans une première version, la procédure de simulation de l'algorithme est complètement aléatoire : à chaque étape, pour chaque sommet, une couleur est attribuée parmi les couleurs déjà ouvertes ou bien un nouveau groupe de couleur est créé. Cette méthode conduit à ouvrir beaucoup de nouveaux groupes de couleurs ce qui augmente la valeur de la fonction objectif rapidement. Nous avons ensuite intégré deux versions d'algorithmes gloutons différents, l'un consistant à colorer chaque sommet avec une couleur aléatoire déjà ouverte et n'ouvrant de nouvelles couleurs que lorsque qu'un sommet ne peut être affecté dans aucun groupe de couleurs déjà ouvert sans créer de conflits, l'autre consistant à colorer chaque sommet avec la première couleur disponible.

Enfin, nous avons abordé une manière d'améliorer la phase de simulation avec une recherche locale, l'idée étant d'utiliser l'arbre de recherche comme un outil pour apprendre à placer les sommets les plus lourds du graphe dans les bons groupes de couleurs puis à utiliser une recherche locale uniquement pour colorer les sommets restants. La procédure de recherche locale implémentée est une recherche locale itérative basée sur un opérateur grenade [2].

3 Expérimentation

Nous avons appliqué les différentes variantes de notre algorithme sur 4 benchmarks d'instances utilisés dans l'état de l'art soit 162 instances provenant du challenge DIMACS ou de problèmes de décomposition de matrices. Lors de cette étude, nous avons pu identifier différents patterns qui dépendent du type d'instance traitée.

Lorsque l'instance est grande (nombre de sommets/nombre d'arcs), le MCTS n'arrive pas à trouver des zones prometteuses de l'espace de recherche en un temps limité et il semble bénéfique de favoriser plus d'exploitation, ce qui peut être fait de trois façons différentes : (i) lors de la phase de sélection, en diminuant le coefficient exploration/exploitation, (ii) en utilisant une heuristique déterministe adaptée au problème lors de la phase de simulation (placer les sommets les plus lourds et de degré le plus élevé dans les premières couleurs disponibles), (iii) en utilisant une recherche locale pour améliorer les solutions complétées à la fin de la phase de simulation.

À l'inverse, pour les plus petites instances, il semble plus utile de favoriser l'exploration pour éviter une stagnation dans des minima locaux. Ce qui peut être fait en augmentant le coefficient exploration/exploitation ou en utilisant une stratégie de simulation plus aléatoire qui permet d'obtenir une meilleure évaluation des branches les plus prometteuses de l'arbre de recherche. Il est intéressant de noter que pour les petites instances, il est possible d'explorer tout l'arbre de recherche assez rapidement en élaguant les branches dont le score minimal est supérieur au meilleur score obtenu jusqu'ici ce qui permet de prouver l'optimalité.

Pour les instances de taille moyenne, il est important de trouver un bon compromis entre exploration et exploitation. Pour ces instances, le couplage entre le MCTS et la recherche locale peut être profitable car il permet de trouver dans certains cas de meilleures solutions que le MCTS ou la recherche locale seule.

Références

[1] Olivier Goudet, Cyril Grelier, and Jin-Kao Hao. A deep learning guided memetic framework for graph coloring problems. *arXiv :2109.05948 [cs]*, Sep 2021. arXiv : 2109.05948.

[2] Bruno Nogueira, Eduardo Tavares, and Paulo Maciel. Iterated local search with tabu search for the weighted vertex coloring problem. *Computers & Operations Research*, 125 :105087, Jan 2021.

[3] Yiyuan Wang, Shaowei Cai, Shiwei Pan, Ximing Li, and Monghao Yin. Reduction and local search for weighted graph coloring problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(0303) :2433–2441, Apr 2020.