RAPPORT DE PROJET
Coloration de graphes

Recherche opérationnelle
L3 SDN - 2024/2025

M. WEINBERG

Jade Delebecque
Clément Szewczyk

Sommaire

Sommaire **2**

I - Introduction **3**

Problème, contraintes et objectifs 3

Équipe et Mode de Travail 3

II - Cadre théorique **3**

III - Implémentation de la structure de donnée **3**

IV - Implémentation des algorithmes **4**

Welsh Powell - Réalisé par Clément 4

Hill Climbing - Réalisé par Jade 5

V - Méthodologie de test **7**

Observations 7

VI - Discussion et compétition **7**

Graphe anna.col 8

Graphe queen9_9.col 8

Graphe myciel7.col 8

Graphe DSJC1000.9.col 8

VII - Conclusion **8**

I - Introduction

Problème, contraintes et objectifs

L'objectif du problème de coloration de graphe est de trouver une affectation des sommets d'un graphe de sorte que deux sommets reliés par un arc soient affectés dans des groupes différents et que le nombre de groupes soit le plus petit possible.

Pour ce projet, l'objectif était d'implémenter deux algorithmes de colorations. Nous avons choisi de développer le Welsh Powell et Hill Climbing.

Équipe et Mode de Travail

Pour ce projet, notre équipe est composée de Jade Delebecque et de Clément Szewczyk. Nous nous sommes réparti le travail de sorte à ce que chacun développe un algorithme. La gestion des graphes et des structures de données a été réalisée en équipe.

Le travail a été partagé régulièrement sur un dépôt GitHub, avec un markdown de suivi mis à jour à chaque commit. Des réunions hebdomadaires ont permis de faire état de l'avancée du projet et de coordonner les sprints suivants.

II - Cadre théorique

Un graphe simple *G* est un couple formé de deux ensembles :

- un ensemble $X = \{x1, x2, \dots, xn\}$ dont les éléments sont appelés **sommets**, représentés par des points
- un ensemble $A = \{a1, a2, \dots, am\}$, partie de l'ensemble $P2(X)$ des parties à deux éléments de X, dont les éléments sont appelés **arêtes,** représentées par des lignes reliant deux sommets.[1]

Nous avons choisi deux approches plus ou moins complémentaires pour aborder ce problème :

- Une méthode gloutonne rapide : le Welsh Powell
- Une méthode de recherche local : le Hill Climbing

III - Implémentation de la structure de donnée

Nos différentes recherches théoriques nous ont amenés à plusieurs types de structures possibles, notamment : matrice d'incidence, deux tableaux, matrice d'adjacence, liste de successeurs…

Nous avons fait le choix de cette dernière structure : la liste de successeur, ou de voisins dans le cas d'un graphe orienté. C'est une structure composée d'un tableau adressant une

[1] https://www.imo.universite-paris-saclay.fr/~ruette/mathsdiscretes/polygraph-Sigward.pdf

liste de successeurs noté TS d’un graphe $G = (X, U)$ possédant n sommets et m arcs. Ainsi l’élément $TS[i]$ du tableau contient la liste des successeurs du sommet xi.[2]

Nous avons déduit de nos différentes recherches que l’accès aux successeurs ou aux voisins d’un sommet est plus rapide avec le dictionnaire car il n’est pas nécessaire de parcourir toute la ligne de la matrice. L’utilisation d’un dictionnaire permet également de nommer les sommets sans ambiguïté et ne le limite pas à des entiers.[3]

Nous avons donc créé la fonction `“lire_graphe_col”` qui vient lire un graphe à partir d’un fichier au format DIMACS (.col) pour ensuite le représenter sous forme de liste de successeurs.

La fonction retourne un dictionnaire où
- **Les clés** représentent les sommets du graphes
- **Les valeurs** représentent des listes contenant les voisins de chaque sommet.

Exemple :

Graphe	**Liste de successeurs**

`c Commentaire`
`p edge 4 4`
`e 1 2`
`e 2 3`
`e 3 4`
`e 4 1`
	`{`
	`  1: [2, 4],`
	`  2: [1, 3],`
	`  3: [2, 4],`
	`  4: [3, 1]`
	`}`

IV - Implémentation des algorithmes

Welsh Powell - Réalisé par Clément

L'algorithme de Welsh-Powell est un algorithme glouton qui consiste à colorer séquentiellement le graphe en visitant les sommets par de degré décroissant. L’idée est que les sommets ayant le plus de voisins seront plus difficiles à colorer, et donc il faut les colorer en premier[4]. Son fonctionnement est le suivant :

1. **Tri des sommets** : Les sommets du graphe sont triés par ordre décroissant de degré (nombre de voisins).
2. **Coloration itérative** : L'algorithme parcourt la liste des sommets triés et attribue la première couleur disponible à chaque sommet, en veillant à ce qu'aucun de ses voisins n'ait déjà reçu cette couleur. Si aucune couleur n’est disponible, il en ajoute une.

[2] Définition “Les manuels visuels pour la licence Informatique” - Editeur Dunod - Chapitre 12. p 237
[3] https://info-mounier.fr/terminale_nsi/structures_donnees/graphes
[4] https://members.loria.fr/JDumas/files/tipe/rapport.pdf

Nous avons donc créé la fonction `“welsh_Powell”` qui prend en paramètre d’entrée “graphe” qui est un dictionnaire représentant le graphe sous forme de liste adjacente. La fonction retourne une coloration sous forme de dictionnaire où chaque sommet est associé à une couleur et une durée en secondes

Pour trier les sommets du graphe, nous utilisons la fonction `sorted`, en définissant une clé basée sur une fonction anonyme qui calcule la taille de la liste des voisins de chaque sommet. Cela nous permet de classer les sommets en fonction du degré (nombre de voisin) de chacun.

Une fois les sommets triés, nous les parcourons un par un pour leur attribuer une couleur. Lors de cette attribution, nous vérifions les couleurs déjà assignées aux voisins afin d’éviter tout conflit, c'est-à-dire que deux sommets adjacents ne partagent jamais la même couleur. Ce processus garantit une coloration valide du graphe

À la fin, on retourne une coloration qui pourra être associée au graphe afin de retourner un graphe coloré et une durée qui est le temps que notre algorithme à mis pour réaliser la coloration.

Hill Climbing - Réalisé par Jade

L’algorithme de Hill Climbing consiste à utiliser une fonction heuristique pour choisir à chaque étape le nœud à générer. Une solution initiale est améliorée de manière itérative en apportant de petits changements progressifs.[5] C’est la stratégie de l’alpiniste (soi-disant) qui choisit la direction de la pente la plus forte pour arriver au sommet le plus vite.[6]

Le Hill Climbing est donc un algorithme de recherche locale qui fonctionne en avançant petit à petit vers de meilleures solutions. Son fonctionnement est le suivant :

1. **Initialisation** : On commence avec une coloration du graphe.
2. **Recherche de voisins** : On regarde les solutions proches en changeant la couleur d’un ou plusieurs sommets.
3. **Choix du meilleur voisin** : Parmi les voisins, on prend celui qui réduit le plus le nombre de conflits (c’est-à-dire les arêtes entre sommets de même couleur).
4. **Itération** : On répète le processus en partant de la nouvelle solution, jusqu’à ce qu’on soit bloqué dans un optimum local, où aucune amélioration n’est possible.

Concernant notre initialisation : Il est normalement conseillé d’avoir une première coloration avec une heuristique qui permette d’avoir une solution “acceptable” avant d’appliquer le Hill Climbing. Nous avons pourtant fait le choix de mettre à disposition deux fonctions différentes avec deux procédés différents.

[5] https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/
[6] https://www.irit.fr/~Philippe.Muller/Cours/RO_si/trsp_cours_ia.html

La première fonction `hill_climbing_pasopti` est une version du Hill Climbing dites “moins optimisée” car l’initialisation est faite avec une coloration aléatoire. Le nombre initial de conflits peut donc fluctuer selon la coloration générée. Cette version a initialement été mise en place car la condition d’arrêt (tous les conflits sont résolus) empêchait le Hill Climbing d’effectuer la moindre itération.

Cette approche nous a amené à une constatation intéressante : dans certains cas, l’aléatoire génère parfois des configurations initiales inattendues qui nous amène à obtenir des solutions plus optimisées que les heuristiques classiques. C’est pourquoi nous avons gardé cette version au lieu de simplement la “corriger”.

La seconde fonction `hill_climbing_opti` est la version du Hill Climbing dites “plus optimisée” parce que l’initialisation est effectuée grâce au Welsh Powell décrit précédemment. Cette coloration permet de commencer la recherche avec une solution initiale satisfaisante.

Cette fonction a en revanche demandé l’implémentation d’un autre outil : la fonction `forcer_conflits_progressif` qui vient modifier un pourcentage de sommets pour introduire des conflits. Cet outil consiste à créer une “pente” à améliorer pour rechercher de meilleures solutions. Sans cet outil, la condition d’arrêt (tous les conflits sont résolus) empêchait le Hill Climbing d’effectuer la moindre itération.

Dans tous les cas, les algorithmes s'arrêtent si tous les conflits sont résolus, ou si le nombre maximal d'itérations est atteint (arbitrairement fixé à 1000 itérations).

`hill_climbing_opti`	**`hill_climbing_pasopti`**
Initialisation avec Welsh Powell	Initialisation aléatoire
Utilisation de la fonction `forcer_conflits_progressifs`	

V - Méthodologie de test

Nous avons choisi plusieurs graphes à tester via les sources suivantes :

- https://cedric.cnam.fr/~porumbed/graphs/
- https://mat.tepper.cmu.edu/COLOR/instances.html

Propriétés du graphe testé			Résultats		
Nom du graphe	**Nombre de sommets**	**Nombre d'arêtes**	**Welsh Powell**	**Hill Climbing opti**	**Hill Climbing pas opti**
anna.col[7]	138	986	0.0005 secondes 11 couleurs	8 itérations 15.951 secondes 11 couleurs	6 itérations 20.086 secondes 11 couleurs
queen9_9.col[8]	81	2112	0.0005 secondes 15 couleurs	8 itérations 11.832 secondes 13 couleurs	4 itérations 8.046 secondes 15 couleurs
myciel7.col[9]	191	2360	0.0007 secondes 8 couleurs	16 itérations 139.206 secondes 8 couleurs	10 itérations 135.994 secondes 9 couleurs
DSJC1000.9.col[10]	1000	449449	0.202 secondes 313 couleurs	Temps d'exécution trop élevé pour le test	Temps d'exécution trop élevé pour le test

Observations

Le Welsh-Powell est extrêmement rapide avec pour tous les graphes, un temps d'exécution inférieur à une seconde, et ce même pour le graphe à plus de 1000 sommets.

Les deux versions de Hill Climbing aboutissent généralement au même nombre de couleurs que Welsh-Powell, sauf pour queen9_9.col, où la version optimisée obtient une meilleure solution (13 couleurs contre 15 pour les autres).

VI - Discussion et compétition

Il a été mis en place une compétition dans la promotion grâce à un excel partagé entre les camarades :
https://docs.google.com/spreadsheets/d/1O1IdGuce7CsgjRNVPwF1shR9gCF4UZAA-Enj4YRAjHI/edit?usp=sharing

[7] https://mat.tepper.cmu.edu/COLOR/instances.html
[8] https://mat.tepper.cmu.edu/COLOR/instances.html
[9] https://mat.tepper.cmu.edu/COLOR/instances.html
[10] https://cedric.cnam.fr/~porumbed/graphs/

Les résultats ont été évalués selon plusieurs critères : nombre de couleurs utilisées, temps d'exécution, nombre de conflits résiduels, et nombre d'itérations si applicable.

Graphe anna.col

Avec **Welsh-Powell**, notre groupe a obtenu des résultats parmi les plus rapides avec 11 couleurs et un temps quasi-instantané de 0.00057 s.

En revanche, notre **Hill Climbing** a été plus lent (1.7 s) mais a produit une solution optimale avec 11 couleurs.

Graphe queen9_9.col

Avec **Welsh-Powell**, notre groupe a produit un résultat rapide mais légèrement moins optimisé (15 couleurs) par rapport à d'autres groupes, mais avec un temps quasi-instantané de 0.00052 s.

En revanche, en **Hill Climbing**, nous avons réussi à réduire le nombre de couleurs à 13, ce qui est la solution la plus optimisée, bien que le temps d'exécution soit plus long (10 s).

Graphe myciel7.col

Notre implémentation **Welsh-Powell** a été l’une des plus rapides avec 8 couleurs en 0.00071 s.

En revanche, notre **Hill Climbing** a été plus lent (154 s) mais a produit une solution optimale avec 8 couleurs.

Graphe DSJC1000.9.col

Avec **Welsh-Powell**, notre groupe a obtenu l'un des meilleurs résultats en termes de nombre de couleurs (313), tout en maintenant un temps d'exécution compétitif (0.202 secondes).

VII - Conclusion

Welsh-Powell s'est révélé extrêmement rapide, avec des temps d'exécution quasi instantanés, même pour des graphes complexes comme DSJC1000.9.col. Cependant, son efficacité en termes de minimisation du nombre de couleurs dépend fortement de l'ordre initial des sommets.

Hill Climbing a montré une meilleure capacité d'optimisation sur certains graphes (par exemple, queen9_9.col), en réduisant le nombre de couleurs par rapport à Welsh-Powell. Cependant, cette méthode entraîne un temps d'exécution bien plus élevé, plus particulièrement sur des graphes de grande taille.