Rapport Graphes

Yann TROU et Wassim DJELLAT

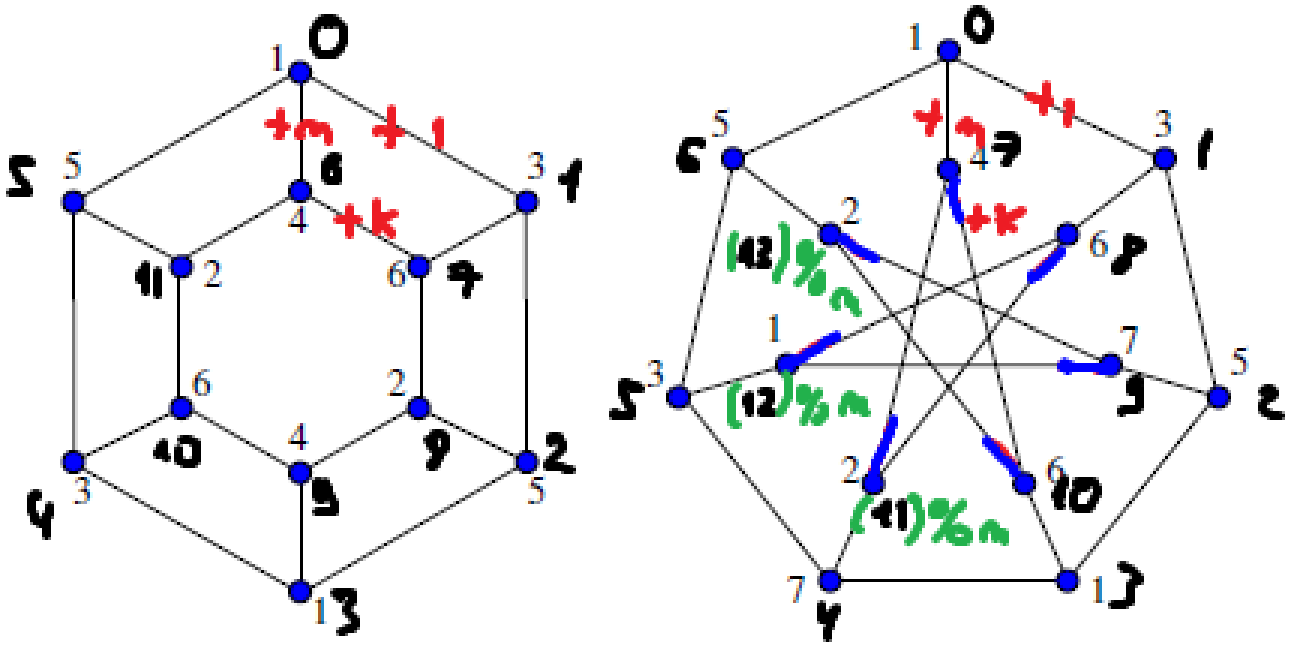
Avril 2021

Table des matières

0.1 Introduction

Dans ce projet, nous devons faire une L2,1 coloration d'un graphe généralisé de Petersen. Un graphe de Petersen généralisé est un graphe particulier composé d'un polygone extérieur et d'un graphe en étoile interne, ces deux graphes étant reliés entre eux par leurs extrémités. Une L2,1 coloration correspond à une coloration avec prise en compte de voisins communs entre deux points. On peut la définir et l'appliquer de la façon suivante : pour donner une couleur a un point, si le point à un voisin, l'écart entre les couleurs des deux points doit être au moins de 2. Aussi, si ces deux points ont un voisin commun, l'écart entre les couleurs doit être d'au moins 1.

Ci-dessous, un schéma représentant deux graphes de petersen avec des notations pour le générateur de graphes. En noir, l'indice des points, en rouge la différence entre les indices pour valide l'arête et en vert l'application d'un modulo pour ne pas dépasser les limites du tableau.



0.2 Fonctionnement

L'application a été conçue sous Visual Studio 2019 en C++ (norme C++17)

Vous trouverez les fichiers sources dans le dossier Projet_Graphes L'application s'utilise de manière simple. Elle vous demandera de saisir n et k puis de choisir si l'on doit donner le résultat pour ces valeurs (mode "simple") ou alors itérer pour toutes les valeurs de k possibles (mode "multiple").

Les résultats seront affichés sous la forme :

- n : <valeur> k : <valeur>
- résultat DSAT : <coloration trouvée par DSAT> | <temps de calcul> ms
- <liste des couleurs calculées par DSATUR pour chaque point du graphe>
- résultat exact : <coloration exacte>
- <liste des couleurs calculées par colorExact pour chaque point du graphe>
- obtenu en <temps de calcul> ms

Le graphe de petersen est représenté sous la forme d'une matrice d'adjacence de taille 2*n par 2*n. Si deux

points sont liés par un arc, on met à 1 la valeur de la matrice aux positions des deux points. Les voisins sont stockés eux aussi dans un tableau à deux dimensions de taille 2*n par 2*n. Le fonctionnement est similaire : si deux points ont au moins un voisin commun, on met la valeur du tableau à un pour aux positions des deux points.

La principale modification apportée a DSATUR et colorExact pour le projet est la fonction convient. elle est la même pour les deux algorithmes.

```
bool convient(int pt, int couleur){
    for (int i = 0; i < 2 * n; i++){
        if (couleur1[i] != 0){
            //si pt et i sont voisins
            if (at(adj, pt, i) == 1){
                if (abs(couleur - couleur1[i]) < 2){
                    return false;
                }
            }
            //si pt et i ont un voisin commun
            if (at(voisins, pt, i) == 1){
                if (abs(couleur - couleur1[i]) < 1){
                    return false;
                }
            }
        }
    }
    return true;
}
```

les autres modifications incluent l'ajout d'une sauvegarde intermédiaire de la coloration pour colorExact afin d'éviter l'arrêt de l'algorithme lorsqu'il ne trouve pas de solution. DSATUR a été aussi modifié pour qu'il boucle sur 2*n au lieu de n afin de parcourir tous les points.

0.3 Résultats

n	k	nbChromatique DSATUR	ms DSATUR	nbChromatique Exact	ms Exact
6	1	8	0	6	1
7	3	10	0	7	5
8	2	10	0	7	28
9	3	9	0	7	34
10	4	9	0	7	90
24	5	8	0	6	300
36	8	8	0	6	32107

Coloration GP(6,1) :
- DSATUR : 1 - 3 - 5 - 1 - 6 - 4 - 5 - 7 - 2 - 4 - 8 - 2
- Exact : 1 - 3 - 5 - 1 - 3 - 5 - 4 - 6 - 2 - 4 - 6 - 2

Coloration GP(7, 3) :
- DSATUR : 1 - 3 - 5 - 1 - 3 - 5 - 7 - 4 - 6 - 2 - 6 - 8 - 10 - 9
- Exact : 1 - 3 - 5 - 1 - 7 - 3 - 5 - 4 - 6 - 7 - 6 - 2 - 1 - 2

Les résultats ci-contres ont étés obtenus en faisant des tests en mode simple.

Il apparait clairement que DSATUR écrase la méthode exacte en termes de performance, surtout pour des valeurs de n élevées même si il ne retourne pas la coloration du graphe optimal.

0.4 Limites et extensions

Comment optimiser colorExact pour obtenir des résultats pour des n plus grands en temps raisonnable ? Ce problème me fait penser à la résolution d'un sudoku. Au lieu de tester chaque valeur naivement l'une après l'autre, on pourrait leur donner une priorité.

0.5 Conclusion

Pour conclure, tout ce qui a été demandé dans ce projet a été réalisé. On a bien une méthode DSATUR et colorExact qui nous retourne une L2,1 coloration valide d'un graphe de Petersen généralisé. On peut calculer la coloration d'un graphe ou alors la coloration d'un graphe pour toutes les valeurs de k possibles.