DM Graphes

Sudoku
Cartes géographiques
Allocation de fréquences dans les réseaux GSM
2021-2022

Benedictus Kent **RACHMAT**
Hichem **KARFA**

I. SUJET

1. Sudoku

Une grille de Sudoku est composée de 9 × 9 = 81 cases qu'on remplit par des chiffres (de 1 à 9). Il est interdit d'avoir plusieurs fois le même chiffre :

- Sur une même ligne
- Sur une même colonne
- Dans chacun des 9 sous-grilles de 3 × 3 = 9 cases.

Étant donnée une grille avec déjà quelques chiffres dans certaines cases, l'objectif du jeu est de deviner comment remplir les cases vides.

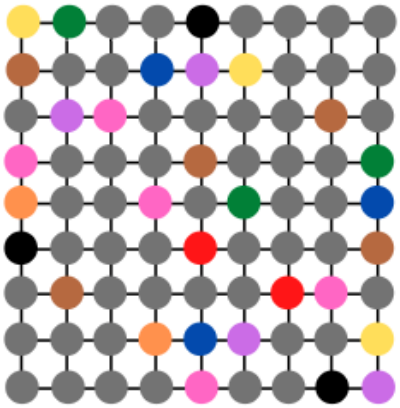
Pour remplir une grille admettant une solution, nous pouvons résoudre le problème en utilisant graphe. Disons que nous avons le graphe G (le sudoku). Nous savons que c'est un **graphe non orienté** et chaque case représente un sommet donc le graphe G a **81 sommets, a un degré de 20, et a un total de 810 arêtes** (81*20/2)**.**

Dans la contrainte, on ne peut pas avoir le même nombre dans la même ligne, dans la même colonne et dans le même sous-grilles. Dans ce cas, il faudrait mettre une arrête reliant le sommet. Et cet arrêt représente que le nombre ne peut pas être le même. Par exemple dans l'image ci-dessous, (X1,Y1) et (X5,Y1) ne peuvent pas avoir la même valeur car ils sont dans la même ligne, (X1,Y1) et (X1,Y5) ne peuvent pas avoir la même valeur car ils sont dans la même colonne, enfin (X1,Y1) et (X3,Y3) ne peuvent pas avoir la même valeur car ils sont dans la même sous-grilles.

	X1	X2	X3	X4	X5	X6	X7	X8	X9
Y1	5	3			7				
Y2	6			1	9	5			
Y3		9	8					6	
Y4	8				6				3
Y5	4			8		3			1
Y6	7				2				6
Y7		6					2	8	
Y8				4	1	9			5
Y9					8			7	9

Après avoir analysé le graphe, l'étape suivante est la colorisation. Pour le nombre 1-9 (les données qui sont données dans le sudoku) nous pouvons attribuer à chacun d'eux une couleur différente. Et pour résoudre le problème il suffit de coloriser le reste du sommet jusqu'à ce que tout soit colorisé. Par exemple :

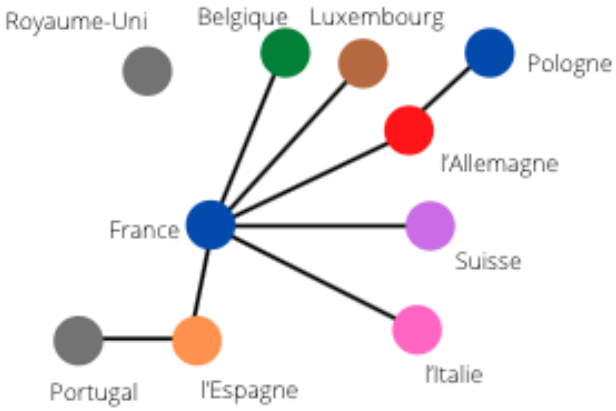
- 1 : BLEUE
- 2 : ROUGE
- 3 : VERT
- 4 : ORANGE
- 5 : JAUNE
- 6 : BRUN
- 7 : NOIR
- 8 : ROSE
- 9 : VIOLET
- Sommet sans données : GRIS



2. Cartes géographiques

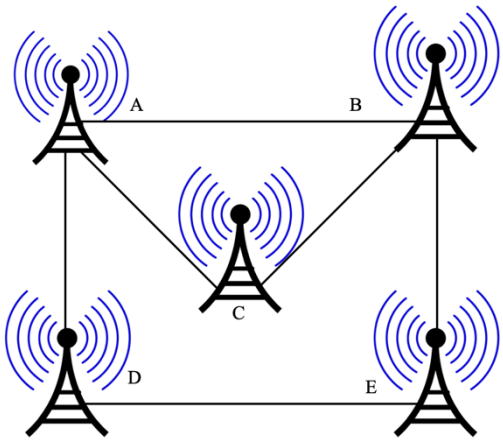
Les cartes géographiques comme celles qu'on peut trouver dans un atlas, ou celles qui sont accrochées dans les classes, ou les livres de géographie, représentent généralement une partie du monde et, afin de mieux délimiter les pays, on colorie ceux-ci de différentes couleurs.

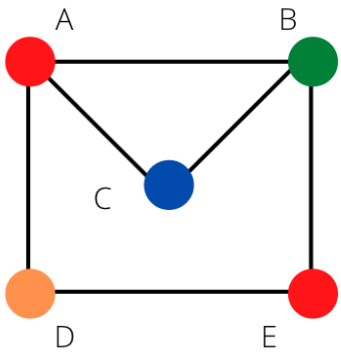
Pour colorier les pays de sorte que deux pays frontaliers ne soient pas de la même couleur, on va résoudre ce problème comme le précédent qui est en colorisant le graphe mais maintenant nous nommons le graphe J. Le graphe J est aussi un **graphe non orienté** et chaque pays représente un sommet et chaque arrêt relie aux les pays voisins. Donc dans ce cas 2 sommet voisins ne peuvent pas avoir la même couleur. Par exemple la France et la Pologne peuvent avoir la même couleur, mais la France, l'Espagne, l'Italie, La suisse, l'Allemagne, Luxembourg et la Belgique ne peuvent pas avoir la même couleur.



3. Allocation de fréquences dans les réseaux GSM

Dans un réseau d'antennes, on souhaite éviter que deux antennes voisines se voient affectée la même fréquence pour ne pas avoir d'interférences. Pour résoudre ce problème nous allons faire la même méthode que la 2 réponse précédente qui est la coloration de graphe. Maintenant nous nommons le graphe H. Le graphe H est aussi un **graphe non orienté** et chaque antenne représente un sommet et chaque arrêt relie aux les antennes voisins. Donc dans ce cas 2 sommet voisins ne peuvent pas avoir la même couleur. Par exemple l'antenne A et l'antenne E peuvent avoir la même couleur, mais l'antenne A, l'antenne B et l'antenne C ne peuvent pas avoir la même couleur.





4. Conclusion

En conclusion, ces trois problèmes ont une manière similaire à résoudre qui est de coloriser le sommet avec une note que le sommet voisin doit avoir une couleur différente.

II. ALGORITHMES

1) Qu'est-ce qu'une heuristique ?

Une heuristique est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale (une méthode approximative). Donc on peut dire que c'est le contraire d'un algorithme exact qui trouve une solution optimale pour un problème donné. Il est généralement plus judicieux de faire appel à des méthodes heuristiques pour des problèmes difficiles parce que les algorithmes de résolution exacte sont exponentiellement complexes.

2) Proposer un premier algorithme naïf, qui permettra de résoudre le problème lorsqu'on ne met pas de contraintes sur le nombre d'éléments permettant de résoudre le problème (typiquement le nombre de fréquences pour le problème des antennes, ou le nombre de couleurs pour le problème des cartes).

Algorithme 1 : glouton_naif (G)

Entrées	:	un arbre G
Résultat	:	associer le sommet de G par un code couleur
		DÉBUT
1	:	couleur ← 0
2	:	pour chaque sommet ∈ sommet_de_G() faire
3	:	| sommet['couleur'] ← couleur
4	:	| couleur ← couleur + 1
		FIN

3) Proposer un second algorithme qui mettra en œuvre une heuristique dont l'objectif sera de tenter d'envisager les sommets dans un ordre plus habile.

Pour mettre en œuvre une heuristique, on a choisi un algorithme appelé algorithme glouton (*Greddy algorithms*). L'algorithme glouton visent à faire le choix optimal à un moment donné. Ils ne regardent pas vers l'avenir pour décider de la solution optimale globale.

Algorithme 2 : glouton (G)

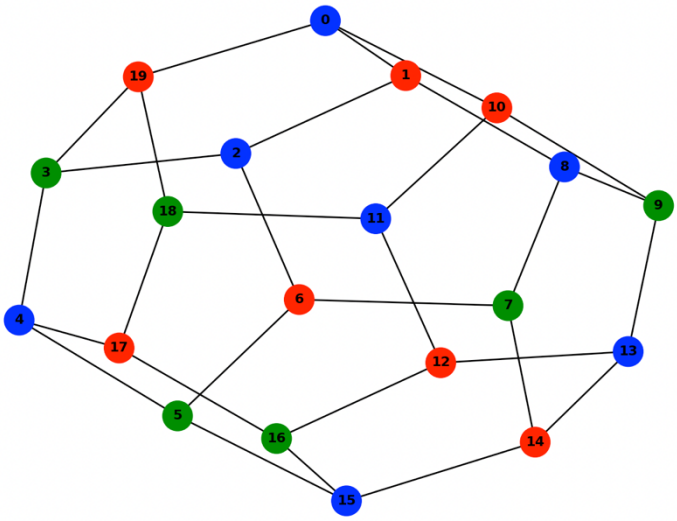
Entrées	:	un arbre G
Résultat	:	associer le sommet de G par un code couleur
		DÉBUT
1	:	sommet ← liste ordonnée des sommets de G avec la valeur décroissante de leur degré
2	:	pour chaque x ∈ sommet faire
3	:	| clr ← première couleur non utilisée par les voisins de x
4	:	| colorer le sommet x avec clr
		FIN

4) Discuter des limites de cette heuristique.

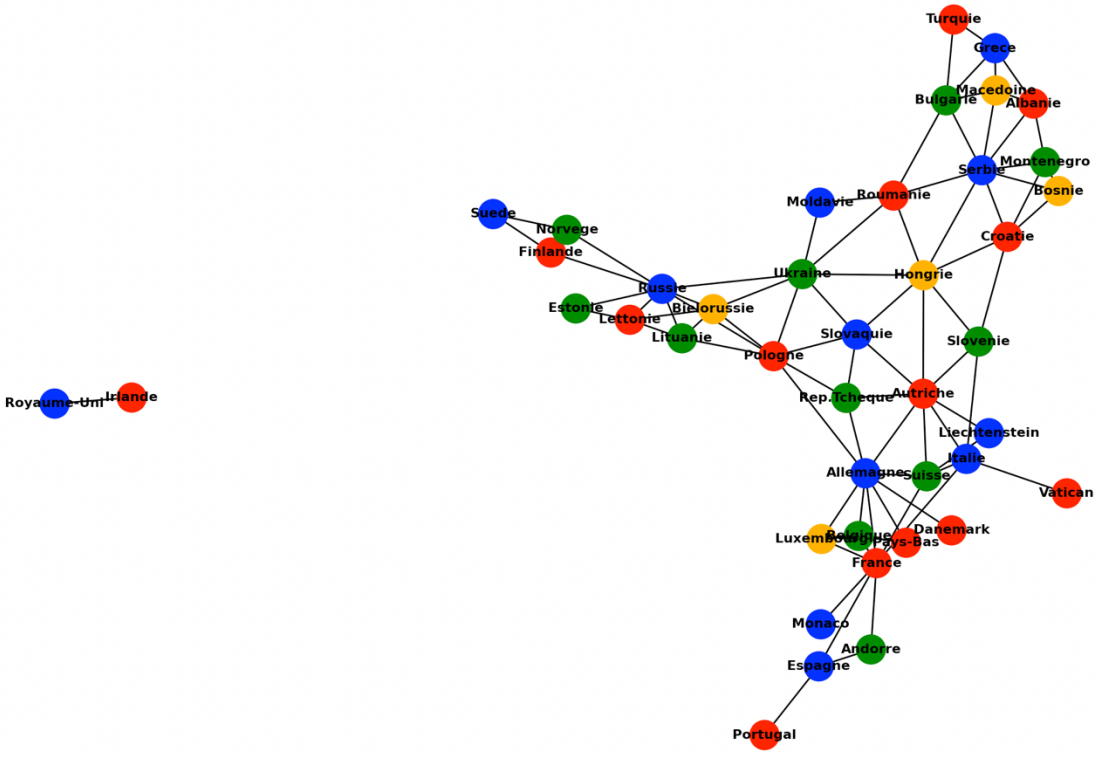
L'algorithme ne s'intéressent qu'à la solution optimale localement. Cela signifie **que la solution optimale globale peut différer de la solution choisie par l'algorithme**. Et aussi ils ne garantissent pas de solutions, mais ils sont très efficaces en termes de temps.

Pour lancer le code, utilisez simplement notre fichier **Makefile** pour faciliter les choses, comme nous pouvons le voir pour le Cartes géographiques et GSM, nous pouvons utiliser l'algorithme glouton pour les résoudre. En analysant le résultat, nous pouvons voir que l'algorithme glouton est plus rapide que glouton naïf. Cela s'est produit parce que nous avons ajouté une mise à niveau dans l'algorithme glouton, par exemple en ajoutant une break dans l'instruction if et en triant la liste par ordre décroissant selon leur degré, ce genre de chose peut accélérer l'exécution de l'algorithme. Le fichier de sortie sera enregistré dans le fichier ***_output.txt** dont la première colonne est le nom des sommets et la deuxième colonne représente le code couleur, par exemple 1 est bleu, 2 est rouge, 3 est vert, etc. et ces données peuvent être représentées graphiquement grâce à networkx. (comme vous pouvez le voir chaque voisin d'antenne a des couleurs différentes)

ce graph est produit par > **make gsm**

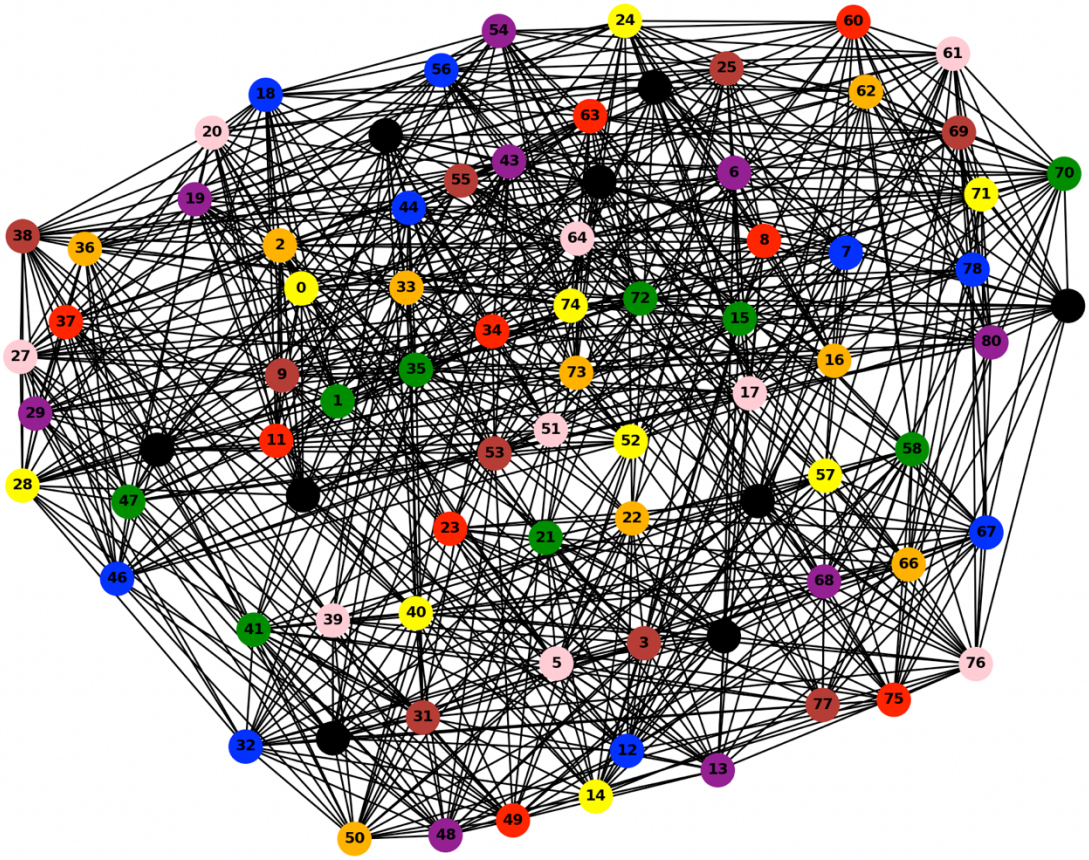


ce graph est produit par > **make map**



Cependant la démarche est différentes avec sudoku, quand on utilise l'algorithme glouton sur sudoku on peut voir le code couleur plus de 9 (10 11 12), ce qui n'est pas bon car on ne peut avoir que 9 code couleur, c'est pourquoi il faut faire un autre algorithme (*backtracking*).

ce graph est produit par **> make sudoku**



Les autres explications peuvent être trouvées dans le commentaire du code,
merci d'avoir lu ce rapport.