TP série N°3: allocation de fréquences d'un réseau mobile

Christoph Samuel – Jankowiak Matthias

I-Position du problème:

1- L'objectif de ce TP est celui de **l'allocation de fréquences d'un réseau mobile.** En effet, dans le cadre d'un projet de déploiement de réseau mobile, une agence d'opérateurs régionaux ont installé plusieurs transmetteurs pour couvrir l'intégralité d'un territoire sur une zone ciblée, en effet, les **transmetteurs sont répartis** géographiquement en fonction de la densité urbaine et de l'activité économique. Le **problème réel posé** est que sur le plan technique, deux **transmetteurs trop «proches»** géographiquement risquent d'engendrer un phénomène **d'interférence** sauf à les faire opérer sur des fréquences «éloignées» (fréquences compatibles). Ainsi, une estimation préalable de l'investissement à réaliser est un facteur déterminant dans la **prise de décision des opérateurs.**

2- Le problème est de déterminer le **nombre minimum de fréquences** suffisamment éloignées que l'agence doit allouer aux opérateurs afin de garantir le fonctionnement, sans interférence, du réseau mobile. En effet, une allocation de fréquence **minimale** de déploiement est **primordiale** pour la rentabilité du projet. Le problème posé peut donc se ramener à un **problème de recherche de coloration** au sein **d'un graphe non orienté** représentatif du réseau mobile à équiper. La recherche du nombre minimum de fréquences compatibles se ramène donc au calcul du **nombre minimum de couleurs nécessaires** : c'est le **nombre chromatique** du graphe.

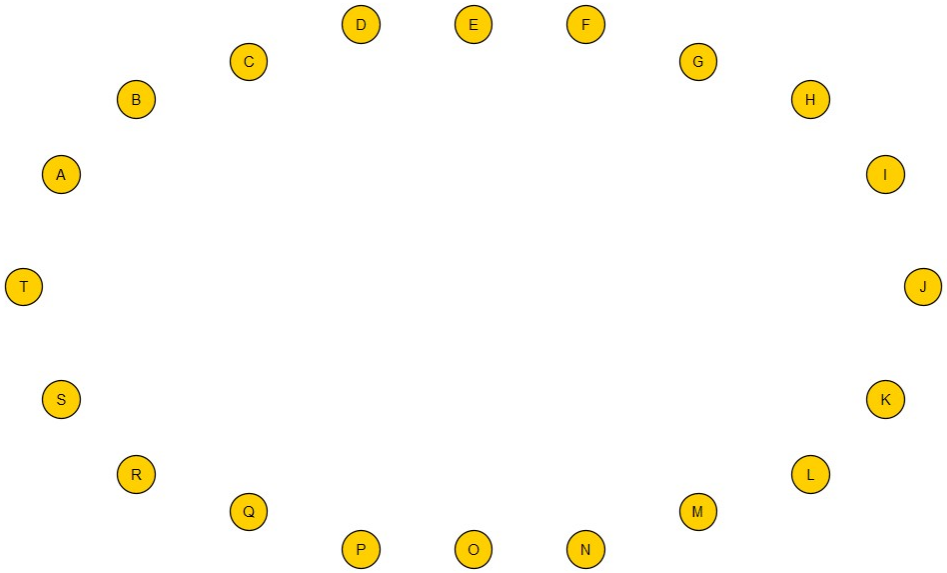
Le tableau de répartition géographique des transmetteurs installés est représenté ci-dessous, le tableau indique, pour **chaque transmetteur,** l'ensemble des transmetteurs avec lesquels il y a **risque d'interférence.**

Le transmetteur	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
...est «proche» des transmetteurs	D E H K	C G M S	B R T	A M S	A F G O	E G Q T	B E F I	A L Q	G K S	N O	A I P R	H O	B D	J T	E J L	K T	F H	C K	B D I	C F N P

3- Le problème réel dans le cadre de la théorie des graphes consiste à **colorer les nœuds représentant les fréquences utilisables** et donc à déterminer le nombre chromatique de celui-ci. Cependant, le problème de calcul du nombre chromatique d'un graphe est **NP-complet,** l'ingénieur se contentera alors d'un **algorithme polynomial** qui fournit une **solution approchée.** Nous avons choisi dans le cadre du TP, de nous appuyer sur l'algorithme de **Welsh-Powell** qui permet d'obtenir une coloration de sommets d'un graphe en utilisant un nombre k «pas trop grand» de couleurs.

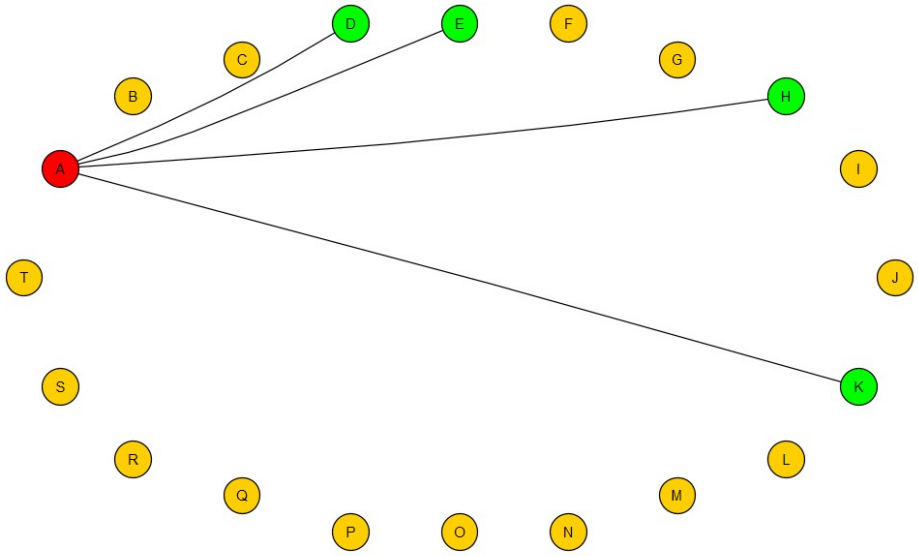
II-Réalisation:

1- Une façon de résoudre ce problème est de **modéliser le réseau mobile** à l'aide d'un **graphe non orienté G = (S,A) définit tel que** chaque sommet S appartenant à S de G représente un nœud et part conséquent une **fréquence de transmetteur** et chaque arête A appartenant à A de G formalise la **relation d'incompatibilité** entre deux fréquences de transmetteur. Nous pouvons par exemple observer les différents cas en partant du graphe vide suivant:



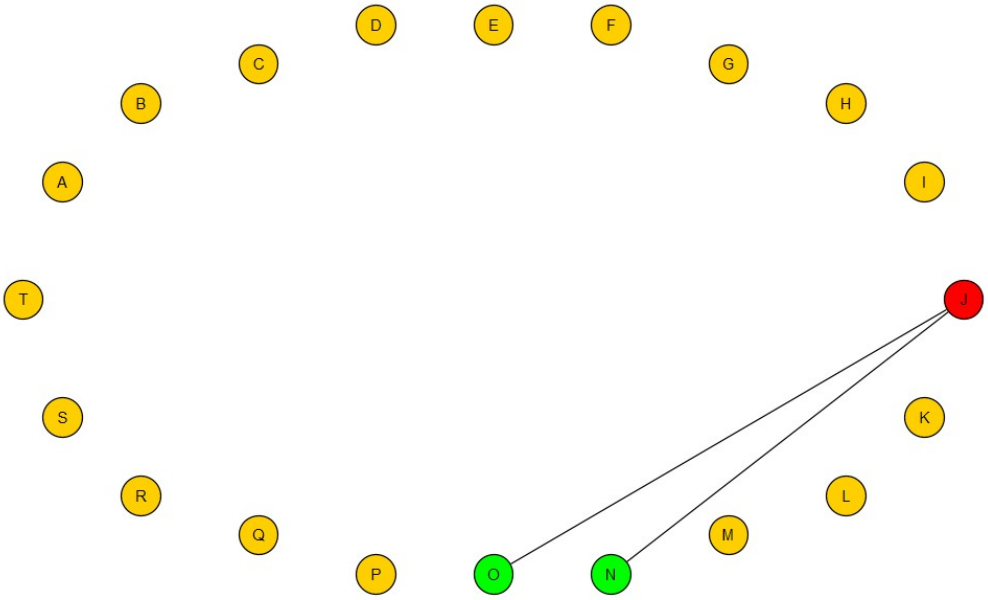
graphe vide

Le transmetteur	A
...est «proche» des transmetteurs	D E H K



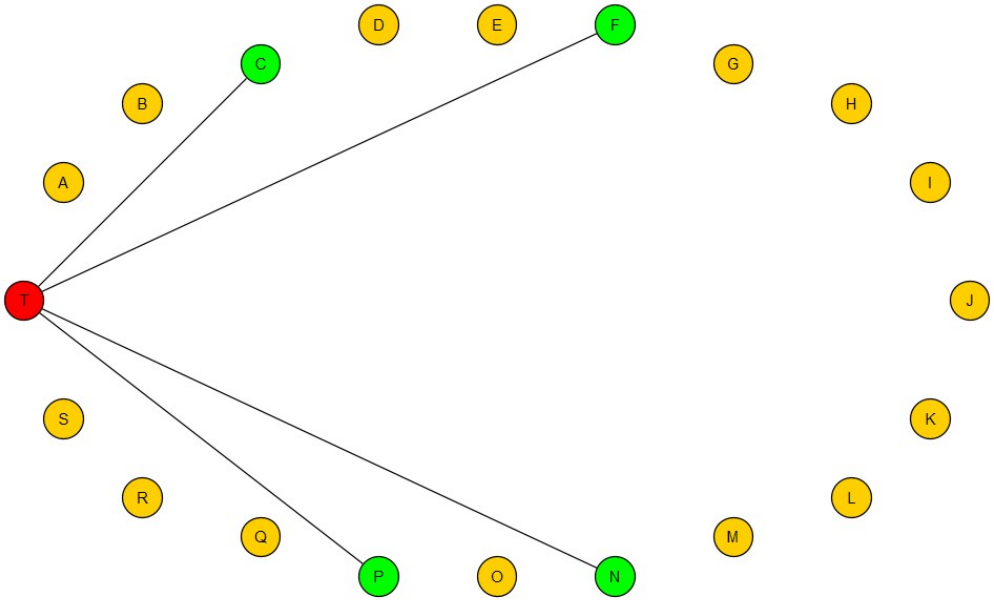
graphe cas A

Le transmetteur	J
...est «proche» des transmetteurs	N O



graphe cas J

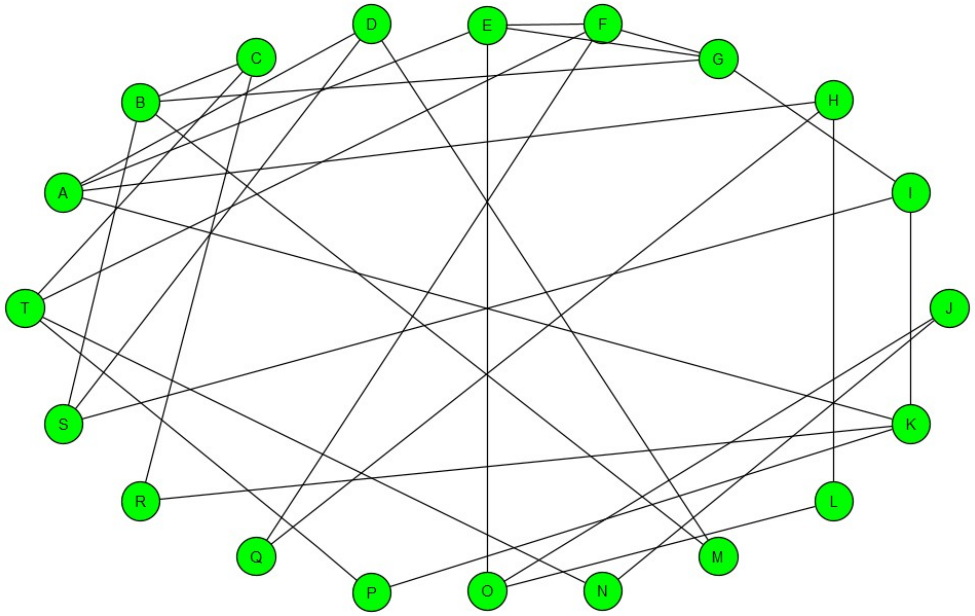
Le transmetteur	T
...est «proche» des transmetteurs	C F N P



graphe cas T

Le transmetteur	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	D	C	B	A	A	E	B	A	G	N	A	H	B	J	E	K	F	C	B	C
...est «proche» des	E	G	R	M	F	G	E	L	K	O	I	O	D	T	J	T	H	K	D	F
transmetteurs	H	M	T	S	G	Q	F	Q	S		P				L				I	N
	K	S			O	T	I				R									P

Mis bout à bout, si on réunit chaque transmetteurs et leur **relations d'incompatibilités,** on retrouve le tableau ci -dessus présenté dans le positionnement du problème, mais on obtient également le graphe **modèle d'incompatibilité des fréquences** suivantes:



modèle d'incompatibilité des fréquences du graphe

2- Ici, le problème consiste à **minimiser le nombre de fréquences à allouer** tout en garantissant leur **compatibilité** vers la disposition géographique des transmetteurs. La recherche du nombre minimum de fréquences compatibles peut être formulable en **termes d'un problème de coloration de graphes.** En effet, sur le modèle de graphe précédent, cela consiste a colorer les nœuds représentant les **fréquences** utilisables.

Les nœuds représentant les fréquences incompatibles sont reliés par une arête comme vu précédemment, ils doivent donc porter des **couleurs différentes.** La recherche du nombre minimum de fréquences compatibles se ramène donc au calcul du nombre minimum de couleurs nécessaires : c'est le **nombre chromatique** du graphe.

3- Nous souhaitons donc proposer une solution à ce problème, or, il est connu que le problème de **calcul du nombre chromatique** d'un graphe **est NP-complet** : il n'existe pas encore d'algorithme polynomial pour le résoudre. Seule une **solution approchée** peut être fournie.

A défaut d'un algorithme polynomial exact, l'informaticien se contentera d'un **algorithme polynomial** qui fournit une solution approchée. Par exemple l'algorithme de **Welsh-Powell** permet d'obtenir une **coloration de sommets d'un graphe** en utilisant un nombre k **«pas trop grand»** de couleurs sans, pour autant, assurer que **k soit minimum,** c'est à dire que $k = \gamma(G)$

k = γ(G), avec γ(G) désignant le **nombre chromatique** du graphe G.

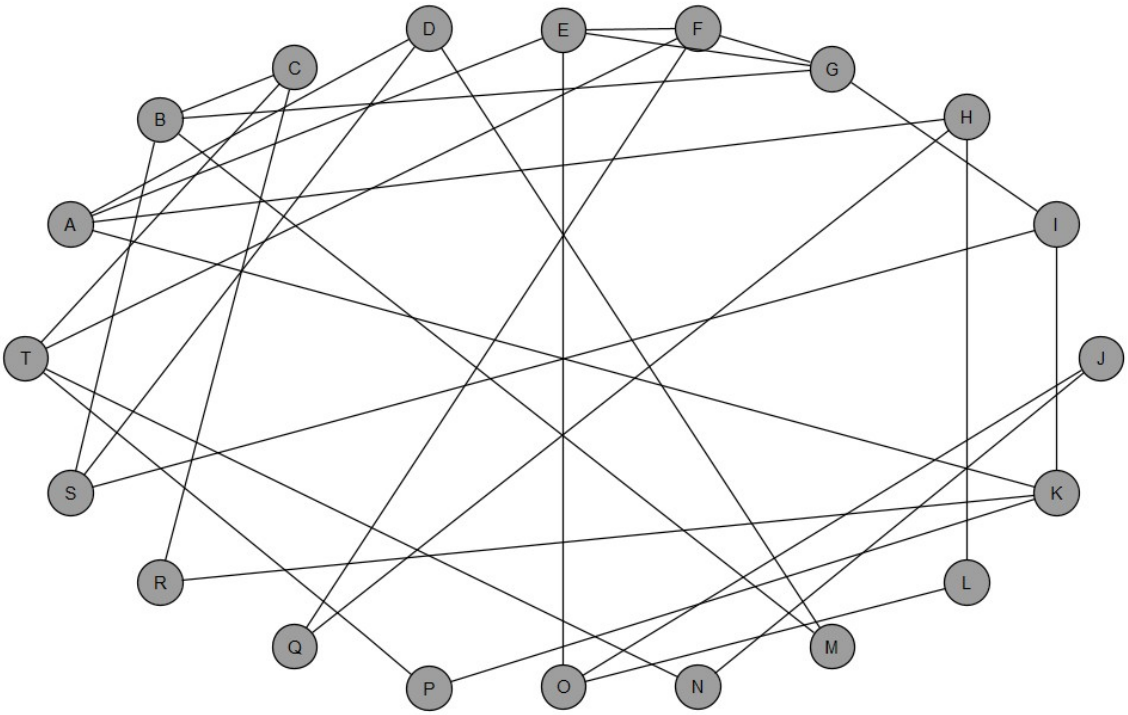
L'algorithme de Welsh-Powell possède une **complexité en O(n + m)** avec n le nombre de sommet et m le nombre d'arête du graphe G. Cette complexité provient du fait qu'on réalise **dans un premier temps un tri** (des sommets par degré décroissant) suivi de la **coloration de chaque sommet.** Il convient alors d'appliquer la procédure de Welsh-Powell, la mise en œuvre de cet algorithme se déroule en **trois étapes:**

Étape 1: Trier les sommets du graphe dans **l'ordre décroissant de leur degré** pour ensuite attribuer à chacun des **sommets son numéro** d'ordre dans la liste triée.

Étape 2: En parcourant la liste des sommets dans **l'ordre de tri,** on attribue une couleur **non encore utilisée** au premier sommet **non encore coloré** ainsi qu'a chaque sommet **non encore coloré et non adjacent** à un sommet de cette couleur.

Étape 3: S'il reste encore des sommets non colorés revenir à l'étape 2, sinon, la **coloration des sommets** est terminée

Déroulons alors l'algorithme de **Welsh-Powell** pour résoudre notre problème, pour cela on reprend notre graphe **modèle d'incompatibilité des fréquences** proposé en 1°: au départ nous avons le graphe G qui n'est pas coloré, après le lancement de notre algorithme, le terminal nous affiche le résultat ainsi la **trace suivante:**



graphe sans couleurs

Étape 1: On commence par **trier les nœuds** dans **l'ordre de leur degré décroissant,** pour obtenir le tableau suivant:

S	A	B	E	F	G	K	T	C	D	H	I	O	S	J	L	M	N	P	Q	R
d°(S)	4	4	4	4	4	4	4	3	3	3	3	3	3	2	2	2	2	2	2	2

Étape 2: On attribut la couleur bleu au nœud A et a tout les nœuds **non adjacents** à A **et non déjà colorés** selon l'ordre déterminé lors de l'étape 1.

S	d°(S)	S	d°(S)
A	**4**	**I**	**3**
B	**4**	**O**	**3**
E	**4**	**S**	**3**
F	**4**	**J**	**2**
G	**4**	**L**	**2**
K	**4**	**M**	**2**
T	**4**	**N**	**2**
C	**3**	**P**	**2**
D	**3**	**Q**	**2**
H	**3**	**R**	**2**

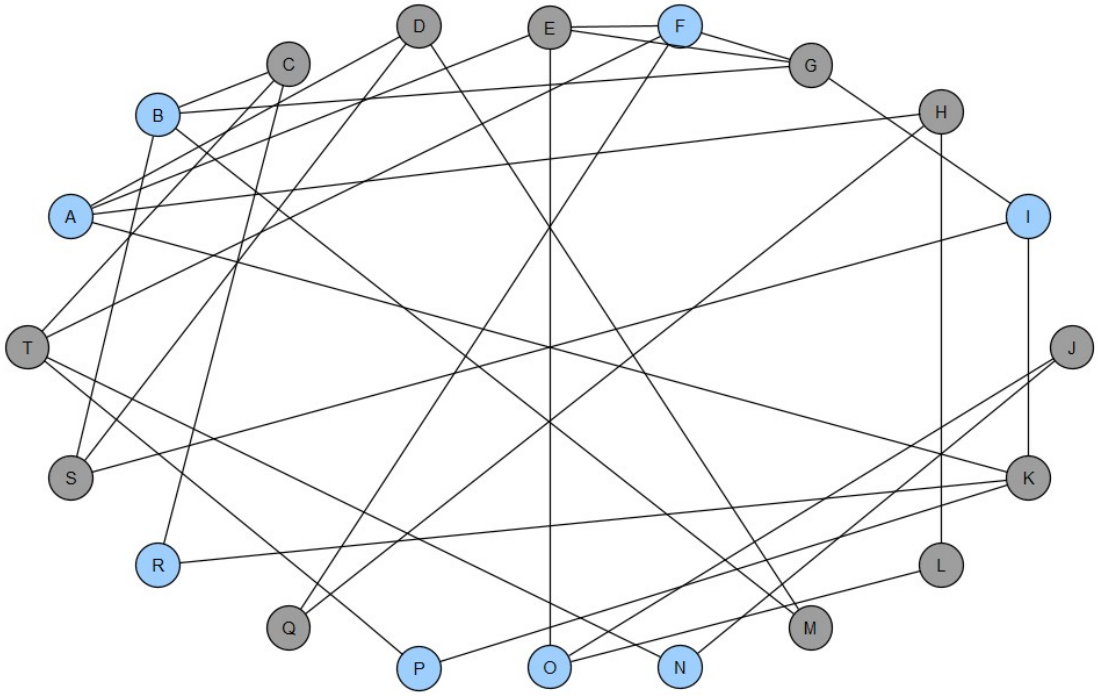


Figure 1: graphe nœud bleu

Stable bleue = {A,B,F,I,N,O,P,R}

Étape 2: On attribut la couleur verte au nœud E et a tout les nœuds **non adjacents** à E **et non déjà colorés** selon l'ordre déterminé lors de l'étape 1.

S	d°(S)	S	d°(S)
A	**4**	**I**	**3**
B	**4**	**O**	**3**
E	**4**	**S**	**3**
F	**4**	**J**	**2**
G	**4**	**L**	**2**
K	**4**	**M**	**2**
T	**4**	**N**	**2**
C	**3**	**P**	**2**
D	**3**	**Q**	**2**
H	**3**	**R**	**2**

Stable verte = {D,E,H,J,K,T}

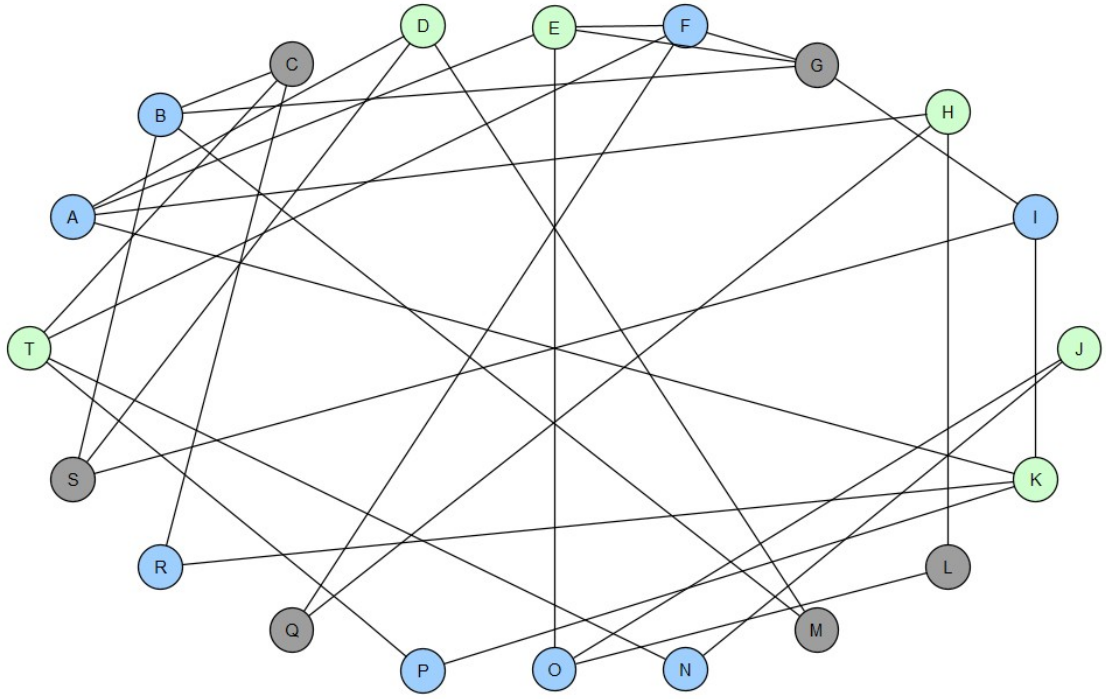


Figure 2: graphe nœud bleu vert

Étape 2: On attribut la couleur rouge au nœud G et a tout les nœuds **non adjacents** à G **et non déjà colorés** selon l'ordre déterminé lors de l'étape 1.

S	d°(S)	S	d°(S)
A	**4**	**I**	**3**
B	**4**	**O**	**3**
E	**4**	**S**	**3**
F	**4**	**J**	**2**
G	**4**	**L**	**2**
K	**4**	**M**	**2**
T	**4**	**N**	**2**
C	**3**	**P**	**2**
D	**3**	**Q**	**2**
H	**3**	**R**	**2**

Stable rouge = {C,G,L,M,Q,S}

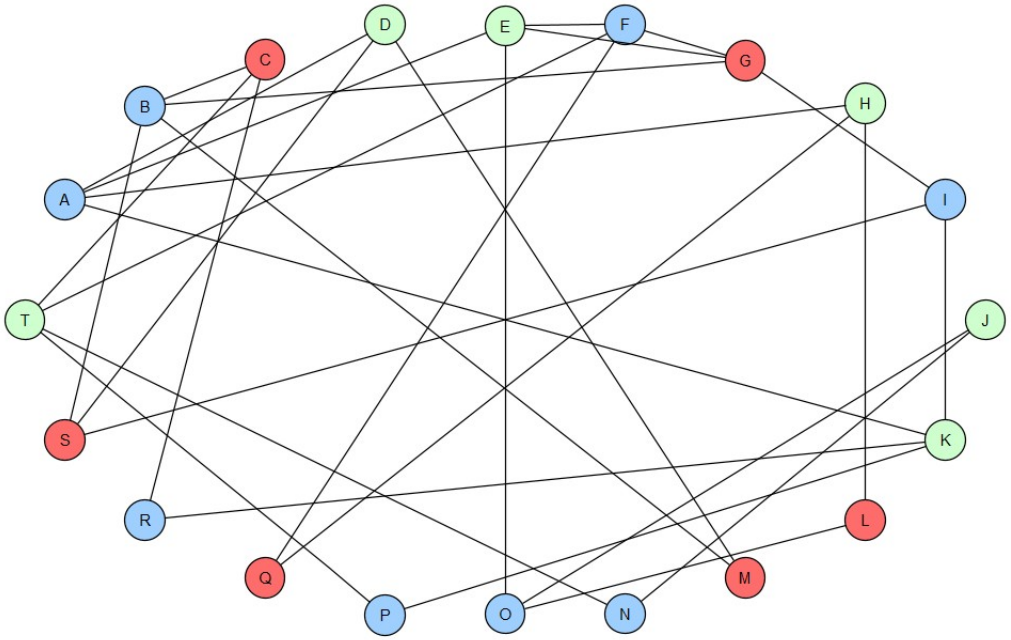


Figure 3: graphe nœud bleu vert rouge

Étape 3: Tout les nœuds sont colorés, l'algorithme s'arrête, la procédure se termine et propose de **colorer le graphe avec 3 couleurs** seulement : nous obtenons donc **k = 3** où k est une approximation de $\gamma(G) = k$: résultat de l'algorithme de Welsh-Powell ($\gamma(G) = 3$).

Le code en **C++** ci-après détaille le fonctionnement de l'algorithme de **Welsh-Powell** dans l'étude de cas précédente. L'ensemble du fichier (tp3.cpp) est fournit de **commentaires** afin de comprendre le fonctionnement de celui-ci, en premier lieu les définitions nécessaire à la création du graphe, la **matrice du graphe** étudié ainsi que l'algorithme de Welsh-Powell affichant les résultat dans le terminal lors de la compilation du fichier. Pour **diversifier notre manière de coder** nous avons décider de partir d'une matrice et non d'utiliser la librairie BoostGraph.

```cpp
// Pour std::cout.
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;

// Initialisation d'une boucle et d'un booléen.
bool stop = false;
int boucle = 0;
// Nombre de sommets du graphe.
const int x = 20;
// Différentes couleur possible pour notre graphe.
const int couleurs[x] = {0,1,2,3,5};

// Dans la structure, on définit les propriétés du graphe.
struct Graf
{
    int edges[x];
    char vertex_id[x];
    bool coloration[x];
    bool adj[x][x];
    int couleurs[x];
};

// Initialisation des différents degrés et sommets.
char vertex_names[x] = {'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T'};
int degre[x];

// Création de notre graphe g à l'aide d'une matrice [x][x].
bool graf[x][x] = {
    0,0,0,1,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,
    0,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,
    0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,
    1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,
    1,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,
    0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,
    0,1,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,
    1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,
    0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,
    1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,
    0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,
    0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,
    0,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,
    0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
    0,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,
    0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0
};

```

Initialisation du graphe à l'aide de la matrice

```cpp
// Algorithme de Welsh-Powell.
void Welsh_Powell(Graf g)
{
    // Initialisation des différentes variables.
    int edge = 0;
    int max = 0;
    boucle++;

    // Comptage des arêtes (adjacent ou non).
    if(boucle == 1)
        for(int i=0; i<x; i++)
            for(int j=0; j<x; j++)
                if(g.adj[i][j])
                    degre[i]++;

    // Sélection du noeud possédant le degré le plus fort.
    for (int w=0; w<x; w++)
        if (!g.coloration[w])
        {
            g.edges[w] = degre[w];
            if (edge < g.edges[w])
            {
                edge = g.edges[w];
                max = w;
            }
        }

    // Coloriation du sommet avec le degré le plus grand en premier.
    g.couleurs[max] = couleurs[boucle];
    cout << "\nChangement de couleur:\nLe sommet " << g.vertex_id[max] << " est de couleur " << g.couleurs[max]

    // Coloriation des sommets non adjacents et non déja colorés au sommet précédent.
    for(int e=0; e<x; e++)
        if(!g.adj[max][e] && max!=e && !g.coloration[e])
        {
            for (int t=0; t<x ;t++)
            {
                // Création du booleen stop qui sort de la boucle si les régles ne sont pas respectés.
                if(g.adj[e][t] && g.couleurs[t]==g.couleurs[max]) stop = true;
                if(t == x-1 && !stop)
                {
                    g.couleurs[e] = couleurs[boucle];
                    cout << "Le sommet " << g.vertex_id[e] << " est de couleur " << g.couleurs[e] << endl;
                    g.coloration[e] = true;
                    stop = false;
                }
                else if(t == x-1) stop = false;
            }
        }
        // Initialisation du sommet possédant le degré le plus grand.
        g.coloration[max] = true;

    // Si le graphe est totalement coloré, la procédure de Welsh_powell prend fin.
    if(all_of(begin(g.coloration), end(g.coloration), [](bool i) { return i; }))
    {
        cout << "\nLe graphe G est totalement coloré\n" << endl;
    }
    // Sinon usage de la récursivité pour continuer la coloration du graphe.
    else Welsh_Powell(g);
}
```

Algorithme de Welsh-Powell

```cpp
// Fonction d'application.
int main()
{
    // Création du graphe.
    Graf g;

    // Initiation des couleurs du graphe.
    for(int y=0; y<x; y++)
    {
        g.couleurs[y] = 10;
        g.coloration[y] = false;
    }

    // Initiation de g avec une copie des valeurs d'une source vers destination.
    memcpy(&g.adj, &graf, sizeof(g.adj));
    memcpy(&g.vertex_id, &vertex_names, sizeof(g.vertex_id));

    // On lance la procédure de Welsh_Powell.
    Welsh_Powell(g);
    return 0;
}
```

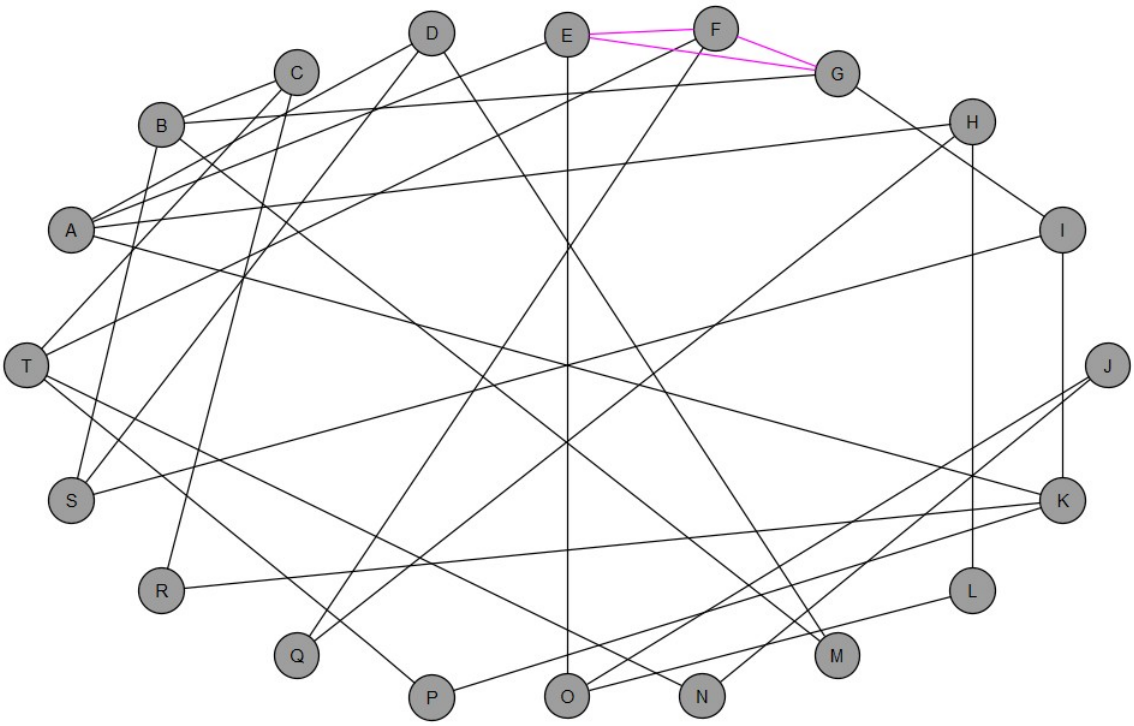
Fonction d'application.

Une fois l'algorithme lancé nous obtenons l'affichage suivant dans notre commande de terminal, on remarque affectivement les même résultats que précédemment, la procédure se termine et propose de **colorer le graphe avec 3 couleurs** seulement.

```
schristoph@scinfe054 ~/Bureau/stockage/TP3 - CHRISTOPH Samuel - JANKOWIAK Matthias
g++ -o tp3.o tp3.cpp
schristoph@scinfe054 ~/Bureau/stockage/TP3 - CHRISTOPH Samuel - JANKOWIAK Matthias
./tp3.o

Changement de couleur:
Le sommet A est de couleur 1
Le sommet B est de couleur 1
Le sommet F est de couleur 1
Le sommet I est de couleur 1
Le sommet O est de couleur 1
Le sommet N est de couleur 1
Le sommet P est de couleur 1
Le sommet R est de couleur 1

Changement de couleur:
Le sommet E est de couleur 2
Le sommet K est de couleur 2
Le sommet T est de couleur 2
Le sommet D est de couleur 2
Le sommet H est de couleur 2
Le sommet J est de couleur 2

Changement de couleur:
Le sommet G est de couleur 3
Le sommet C est de couleur 3
Le sommet S est de couleur 3
Le sommet L est de couleur 3
Le sommet M est de couleur 3
Le sommet Q est de couleur 3

Le graphe G est totalement coloré
```

Résultats terminal

L'algorithme de **Welsh-Powell** permet d'avoir une coloration des sommets cohérente. Cependant, la coloration proposée n'est **pas toujours optimale.** À la fin de l'algorithme, on obtient une **approximation du nombre chromatique,** mais rien ne prouve que la valeur soit la plus petite possible. Il est donc **important d'encadrer** le nombre de couleurs trouvé avec l'algorithme de Welsh-Powell afin de savoir à quel point notre **approximation du nombre chromatique k est fiable et optimal.**



ω(G) = plus grande clique du graphe = {E,F,G}

On peut alors vérifier si k respecte les encadrements de $\gamma(G)$:

$n = 20$ (nombre de nœuds),
$\alpha(G) = 8$ (taille du plus grand stable (stable bleu)),
$\rightarrow n + 1 - \alpha(G) = 13$

$r = 4$ (degré maximal),
$\omega(G) = 3$ (taille de la plus grande clique),
$\rightarrow r + 1 = 5$

k vérifie bien tous les encadrements de $\gamma(G)$:
$\omega(G) \leqslant k \leqslant r + 1$
$\rightarrow 3 \leqslant k \leqslant 5$
ou
$\omega(G) \leqslant k \leqslant n + 1 - \alpha(G)$
$\rightarrow 3 \leqslant k \leqslant 13$

on retient alors l'encadrement le plus minime
$\rightarrow$ **$3 \leqslant k \leqslant 5$**

4- Avec les valeurs obtenues, nous pouvons donc donner une **interprétation du résultat:** chaque couleur modélise **une fréquence différente,** la fréquence alloué doit être différentes selon la couleur, confirmé par le terminal dans notre cas **trois fréquences** différentes sont donc nécessaires pour couvrir sans, a priori, de **risque d'interférences,** l'intégralité du territoire souhaité avec l'ensemble des 20 transmetteurs, cette **solution n'est pas unique**. En pratique, le développeur d'application pourrait équilibrer les charges entre les 3 fréquences en appliquant, au résultat obtenu, un algorithme de **programmation dynamique** mais cela n'est pas si simple.

III-Bilan/Conclusion:

1- Nous avons appris grâce à ce TP que l'application des modèles et **problèmes réels** peuvent être formulés en termes de **problème de coloration de la théorie des graphes**. Dans notre cas, c'est un problème NP-complet, à défaut d'utiliser un algorithme polynomial exact, nous nous sommes contenté d'un algorithme polynomial qui fournit une solution approchée, celui de **Welsh-Powell,** qui par ailleurs nous a permis de mieux comprendre la coloration d'un graphe a travers l'exemple d'un problème réel. Il en devient donc simple de résoudre de tels problèmes, à l'aide du **modèle de graphe** correspondant et des outils à dispositions.

2- Nous retenons également que la plus part **problèmes réels,** concernant le déploiement d'un réseau mobile dans le souci d'une **d'allocation de fréquences nécessaires** pour couvrir l'intégralité d'un territoire ou zone ciblée**,** sont facilement traduisibles en **problème de la théorie des graphes** et que nous pourrions être confrontés à ce **type de problématique** dans le futur si nous sommes amenés, par exemple à travailler dans le **domaine de la télécommunication**.