ROBINET Perrine
STRAINCHAMPS Clothilde

Projet graphe : Coloration impropre

Exercice 1 : (fait en TP)

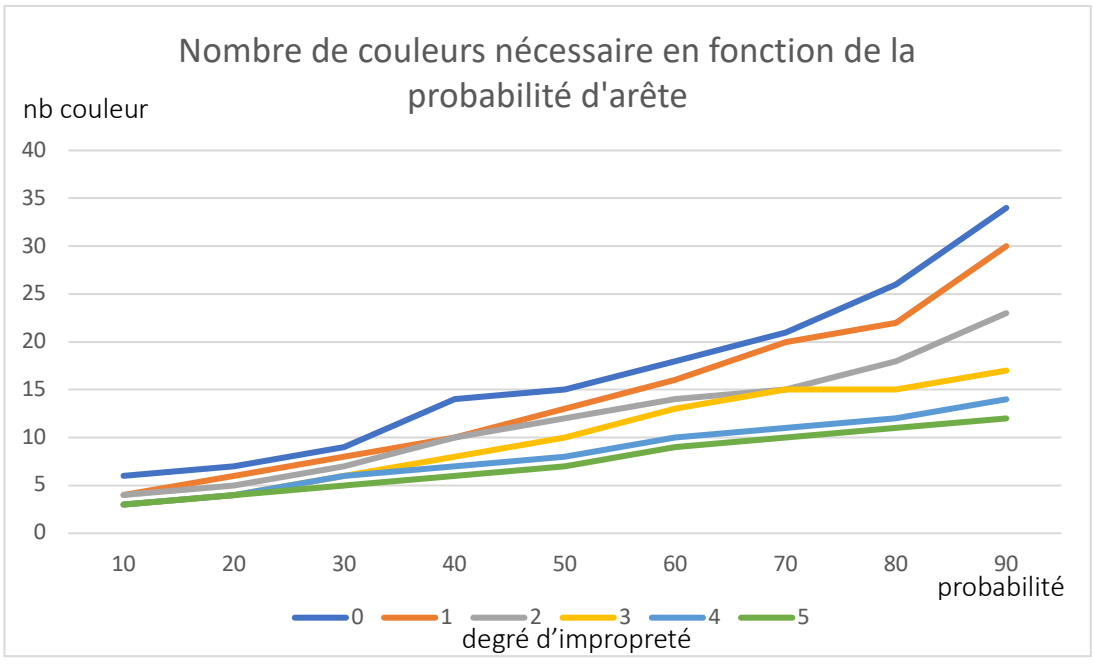
1) L'objectif ici est d'implémenter une version prenant en compte un degré d'impropreté de l'algorithme DSATUR. Pour cela, nous avons pris pour base de la fonction *DSATUR(k)* de la correction du TP3 et nous avons :

   - Créer une fonction *valide(x,c,k)* qui va compter le nombre de voisins de *x* ayant la couleur *c*, si le compteur est supérieur au degré d'impropreté *k* alors *x* ne peut pas être colorié avec *c* car on dépasserait *k* en coloriant *x*, donc si le compteur est inférieur, on peut colorier *x* avec *c*.

   - La fonction *convientDSAT(x,c)* à été reprise pour faire *convientDSAT2(x,c,k)*. *convientDSAT2(x,c,k)* teste si la couleur *c* peut être attribué au sommet *x* tout en respectant la contrainte de ne pas avoir plus de *k* voisins de même couleur, mais aussi la contrainte que pour l'algorithme DSATUR k-impropre qui impose que les voisins de *x* ayant la couleur *c* ait au plus *k-1* voisins de couleur *c*.

   - Mis en paramètre un entier *k* qui représente le degré d'impropreté dans la fonction DSATUR.

2) Pour tester cette version, on va générer un graphe non orienté de 70 sommets et de probabilité p d'arêtes entre deux points allant de 10 à 90 et pour chacune de ces probabilités (allant de 10 en 10) on testera un degré d'impropreté allant de 0 à 5.
   On obtient alors le graphique suivant : (courbe bleu 0 => pour un degré d'impropreté de 0, ...)



ROBINET Perrine
STRAINCHAMPS Clothilde

On peut donc en conclure que plus le degré d'impropreté est élevé, moins de couleurs sont nécessaires pour le colorier.

Exercice 2 : (fait en TP)

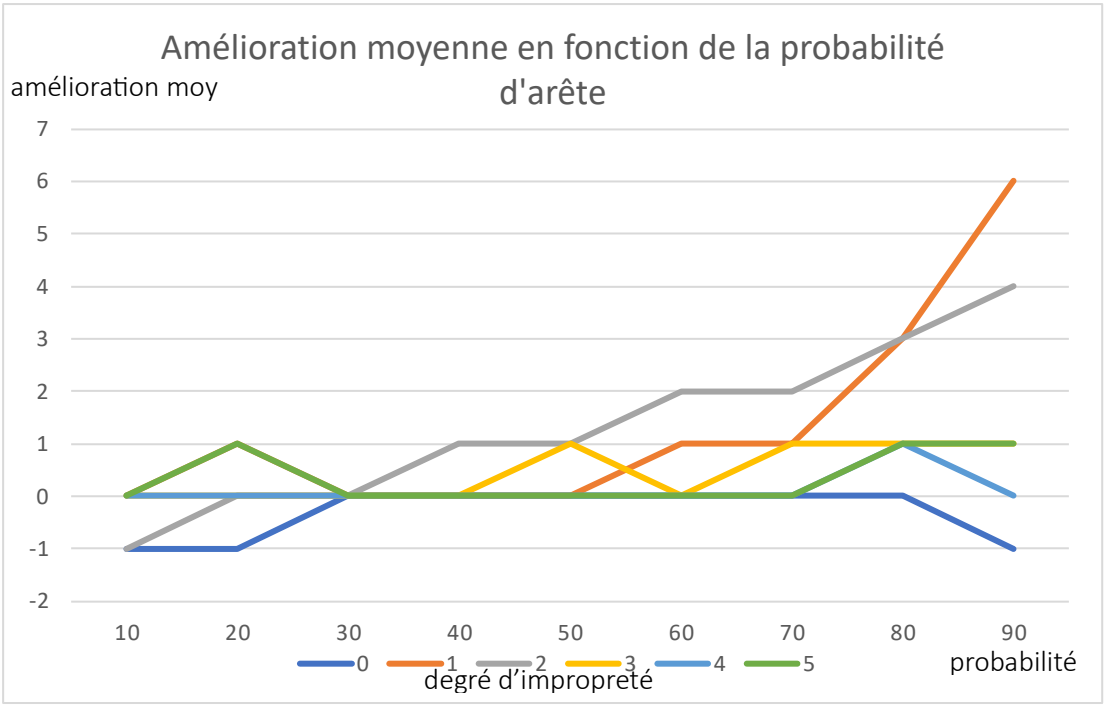
1) Nous avons créé une fonction *DSATUR2(k)* à partir de notre fonction *DSATUR(k)* ne l'exercice 1 et avons modifier la partie sur le calcul du degré de saturation des sommets.

   Dans *DSATUR*, le degré de saturation (DSAT) est initialisé à la valeur du degré de chaque sommet, puis à chaque fois qu'un sommet est colorié, on parcourt tous ses voisins non coloriés et on incrémente leur DSAT de 1 s'ils sont de la couleur choisie. Ainsi, le DSAT représente le nombre de voisins non coloriés de chaque sommet qui ont la même couleur que celle choisie pour le sommet.

   Dans *DSATUR2*, le DSAT est initialisé de la même manière, mais à chaque fois qu'un sommet est colorié, on parcourt tous les sommets du graphe et pour chaque couleur, on compte le nombre de voisins de chaque sommet qui ont cette couleur. On incrémente ensuite le DSAT du sommet avec le nombre de couleurs pour lesquelles le nombre de voisins ayant cette couleur est inférieur à *k* (le degré d'impropreté). Ainsi, le DSAT représente le nombre de couleurs différentes qui peuvent être utilisées pour colorer chaque sommet.

   Pour résumé, *DSATUR* ne prend en compte que le nombre de voisins non coloriés ayant la même couleur que le sommet en question, tandis que *DSATUR2* prend en compte le nombre de voisins ayant chaque couleur et permet de considérer les sommets avec des voisins de différentes couleurs comme étant plus "saturés".

2) Pour tester cette version, on va generer un graphe non orienté de 70 sommets et de probabilité p d'arêtes entre deux points allant de 10 à 90 et pour chacune de ces probabilités (allant de 10 en 10) on testera un degré d'impropreté allant de 0 à 5. Et le degré moyen est calculé sur 100 graphes.



ROBINET Perrine
STRAINCHAMPS Clothilde

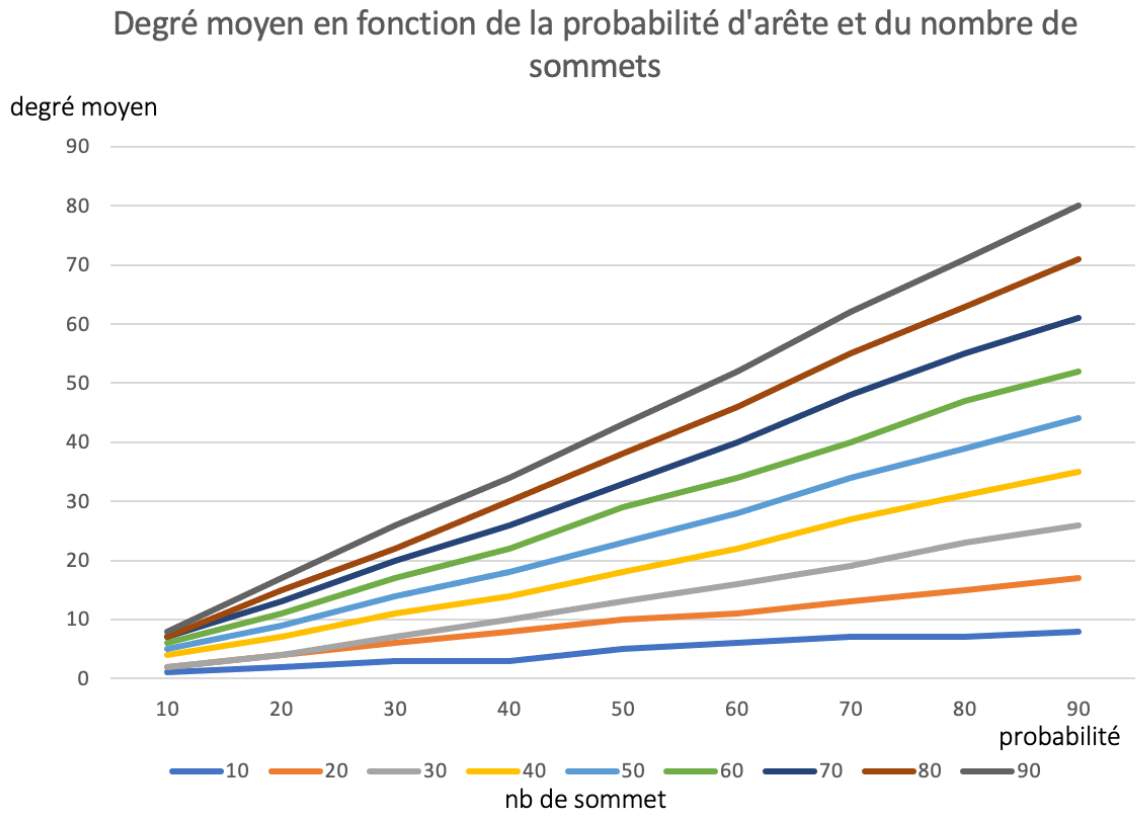
On peut donc en conclure que *DSATUR2* est meilleur que *DSATUR* dans le cas où le nombre du degré d'impropreté est failble (sans compter le 0) et que la probabilité d'arête est élevée.

Une version *DSATUR3* à été implémenter mais au vu de ces résultats nous avons convenu que *DSATUR2* était la bonne implémentation demandée.

Exercice 3 : (fait en TP)

1) Nous avons fait une fonction degmoy() qui calcule le degré moyen des sommets du graphe.

2) On teste le degré moyen pour chaque nombre de sommets (entre 10 et 90) et probabilité d'arête (entre 0 et 90).



On peut donc en conclure que plus le nombre de sommets et la probabitité d'arête augmente plus le degré moyen est élevé (donc que les sommets ont plus de voisins).

3) *DSATUR* et *DSATUR2* utilisent 2 couleurs quand on prend *k* égal au degré moyen.

ROBINET Perrine
STRAINCHAMPS Clothilde

Exercice 4 :

1) On veut implémenter une méthode de recherche de la coloration exacte d'un graphe avec un nombre chromatique i-impropre donné. La méthode de coloration exacte utilisée ici est une méthode de recherche récursive.

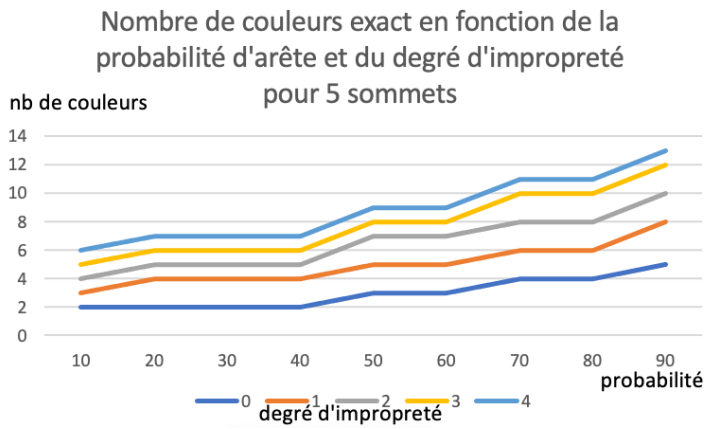
   La fonction *colorexactImpropre* prend en entrée le nombre chromatique *k* et le nombre d'impropreté *i*. Elle initialise la couleur de tous les sommets à 0 et appelle la fonction *colorRRImpropre* avec *x=0* pour démarrer la recherche de la coloration.

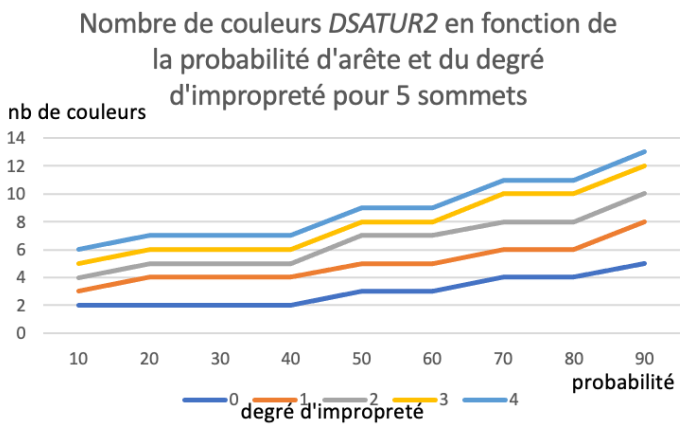
   La fonction *colorRRImpropre* prend en entrée le sommet *x*, le nombre chromatique *k* et le nombre d'impropreté *i* et teste toutes les couleurs possibles pour le sommet *x* avec la contrainte que la couleur ne peut être donnée au sommet *x* que si elle n'est pas utilisée par un de ses voisins et que le degré d'impropreté du sommet *x* est égal à *i*. Si une coloration est trouvée, la fonction affiche la coloration et met la variable *trouve* à true.

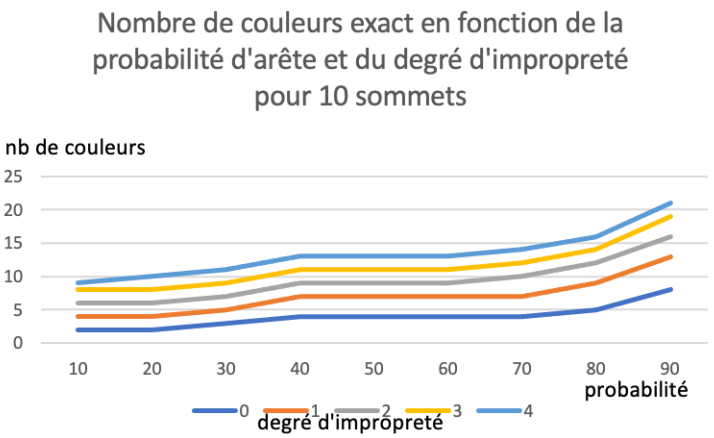
   La fonction *convientExactImpropre* teste si la couleur *c* peut être donnée au sommet *x*. Elle utilise la fonction *valideExactImpropre* pour vérifier que le sommet *x* a au plus *i* voisins de couleur *c* et que les voisins de couleur *c* de *x* ont au plus *i-1* voisins de couleur *c*.

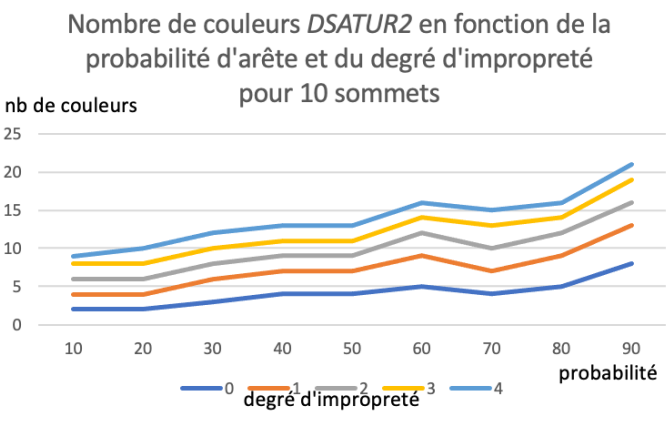
   La fonction *nbChromatiqueI* calcule le nombre chromatique *i*-impropre en testant à partir de *d* couleurs et en diminuant *k* tant que c'est possible. Elle utilise la méthode de recherche de coloration exacte pour chaque valeur de *k*.

2) On veut maintenant comparer les résultats de l'algorithme *nbChromatiqueI* et de *DSATUR2*. Pour cela, on va faire varier le nombre de sommets de 5 à 15 (de 5 en 5) et la probabilité d'arête de 10 à 90, et d'un degré d'impropreté variant de 0 à 4.

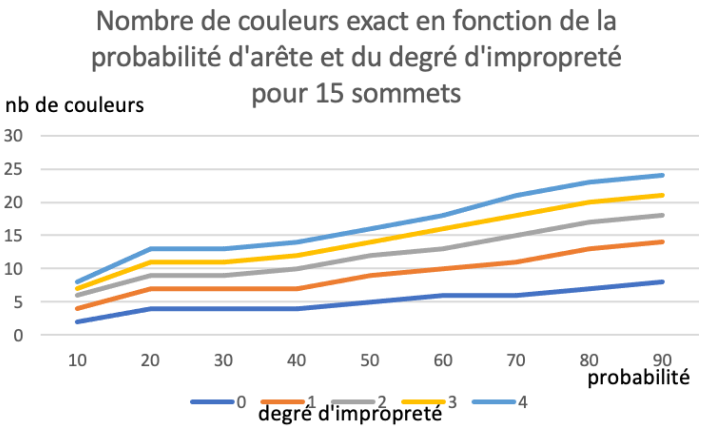


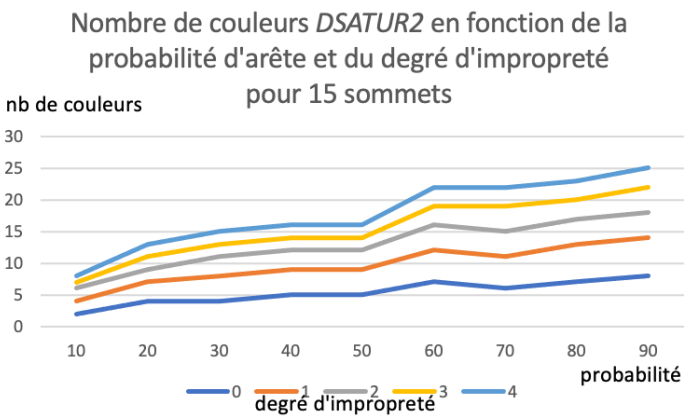






ROBINET Perrine
STRAINCHAMPS Clothilde



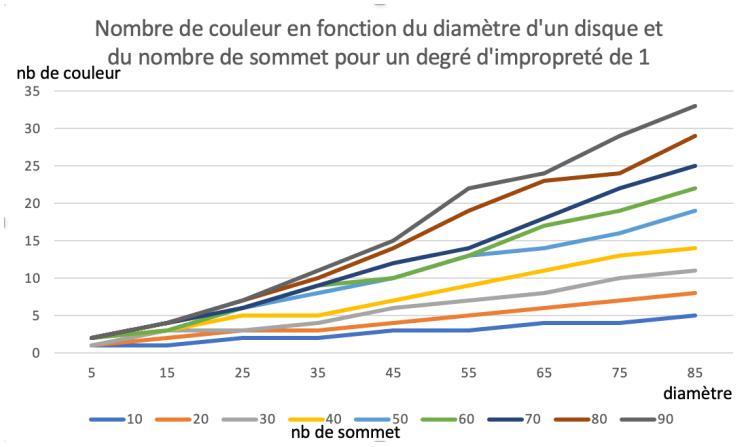


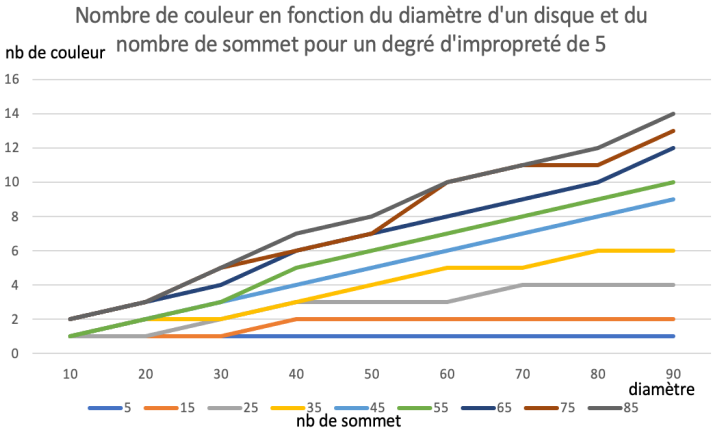
On peut remarquer que les résultats de l'algorithme exact son meilleur que ceux que l'algorithme DSATUR2 pour des graphes plus complexes.

Exercice 5 :

1) Pour cet exercice, nous avons construit 3 fonctions *construitSommets()*, *intersection()* et *construitGraphe()*.

   On veut tester les résultats produits par DSATUR2 (avec degre d'impropreté de 1 et de 5) avec cette configuration de graphe.
   Pour cela, on fixe le plan (100X100) et fait varier le nombre de sommets (de disques) de 10 à 90 puis le diamètre de 5 à 85.





On peut allors en conclure que plus le diamètre et le nombre de sommets est grand plus le nombre de couleur augmente.