Manuel d’utilisation

* Ce document décrit ce qui est fait (pour ne pas qu'on oublie des aspects) et qui donne au moins un exemple d'utilisation.

Ce manuel décrit dans une première partie comment est organisé notre code dans chaque fichier java. Puis dans une seconde partie, nous décrirons comment utiliser l’interface graphique du projet.

# Les classes d’objets

Nous avons utilisé pour notre projet une conception objet : les graphes et les sommets seront des instances de classes. Les classes auront chacune des attributs qui identifiera leurs instances, et des méthodes qui détermineront leur comportement.

## Les sommets

Les sommets sont des agrégations du graphe, et leur fonctionnement est décrit dans le fichier Vertex.java. Un sommet a trois attributs qui l’identifie : un nom unique, une couleur et une liste de sommets voisins. Une arête entre deux sommets A et B est représentée par la présence du sommet B dans la liste de voisins de A et de la présence de B dans la liste de voisins de A.

## Les graphes

Un graphe est représenté dans Graph.java par une liste de sommets. Avec ce fichier, nous pouvons générer des graphes, charger des graphes à partir de fichiers sérialisés et les sauvegarder, et enfin les afficher. Cette classe contient aussi des méthodes et un attribut utilisables par les deux algorithmes que nous avons utilisés pour minimiser le nombre de couleurs : l’algorithme de recuit simulé (ARS) et l’algorithme de Welsh & Powell (AWP). L’attribut en commun est le nombre de couleurs que contient le graphe, et des méthodes permettent d’accéder à un sommet aléatoirement dans le graphe, à exécuter l’algorithme ou à le tester.

Deux classes filles de Graph.java vont adapter le graphe en fonction de l’algorithme qu’elles utilisent pour diminuer le nombre de couleurs du graphe.

* Graphe ARS

La première est GraphARS.java, où l’on va appliquer l’ARS. Cette classe contient des attributs supplémentaires :

* Un tableau de couleurs, indiquant la fréquence de chaque couleur dans le graphe. Cet attribut sera utile lorsque l’algorithme sélectionnera une nouvelle couleur pour un sommet.
* Un tableau contenant les valeurs de l’énergie, pour avoir une trace de l’évolution de l’énergie du graphe durant l’algorithme.
* Des copies du graphe pour revenir à une configuration précédente de la coloration du graphe.
* Des paramètres spécifiques à l’ARS, comme la température maximale par exemple.

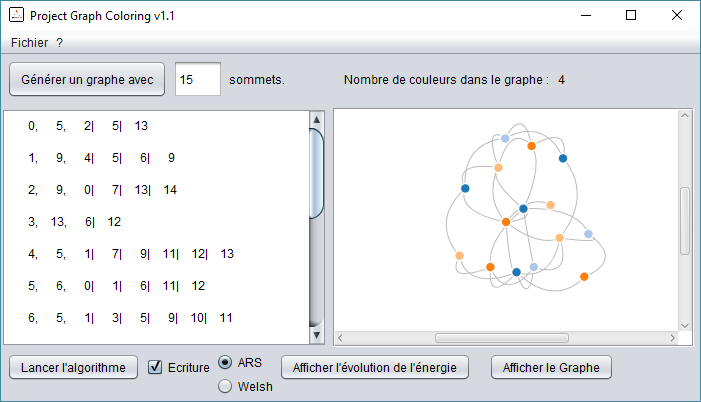
La méthode principale de cette classe est *applySimulatedAnnealingAlgorithm*. C’est elle qui va dérouler tout l’algorithme et faire diminuer le nombre de couleurs. Pour fonctionner, elle aura besoin de ces méthodes :

* *keepChange*, qui va déterminer si le changement apporté à la coloration du graphe doit être sauvegardé ou non.
* *changeColor* va se charger d’apporter une modification élémentaire à la coloration du graphe.
* *adaptNeighbours* sera utilisé pour configurer le graphe après une modification élémentaire, de sorte à ce que deux sommets voisins n’aient pas la même couleur.
* *getRandomColor*, qui va sélectionner une couleur déjà existante dans le graphe ou n’importe quelle couleur en fonction de ce qu’on lui demande.
* Graphe AWP

La deuxième est WelshPowell.java.

# L’interface

Nous avons développé une interface afin de montrer graphiquement le résultat que donne les deux algorithmes sur la coloration du graphe.



## Construire un graphe

L’utilisateur doit tout d’abord avoir un graphe pour pouvoir utiliser les algorithmes de minimisation de couleurs. Pour cela, il a deux possibilités :

* Soit générer un graphe en indiquant le nombre de sommets que ce graphe doit posséder et en cliquant sur « Générer un graphe avec ».
* Soit cliquer sur « Fichier », puis « Ouvrir », et sélectionner le fichier qui contient le graphe souhaité. Ce fichier doit avoir l’extension « .ser ».

Il verra alors apparaître sur la partie gauche de l’écran une série de lignes comportant des nombres. Ce tableau affiche en version texte le graphe. Par exemple, la ligne : « 0, 5, 2 | 5 |13 » s’interprète de la manière suivante : le sommet 0 est colorié avec la couleur 5 et est voisin avec les sommets 2, 5 et 13. Cet affichage nous a été utile au tout début de notre projet, lorsque nous ne pouvions pas encore afficher les graphes avec des points, des couleurs et des arêtes.

Il est aussi possible d’afficher le graphe de manière plus visuelle en cliquant sur le bouton « Afficher le graphe ». Cet affichage est possible en traduisant le graphe au format JSON, puis en lançant un navigateur dans la fenêtre qui va traduire le graphe au format JSON pour modeler un graphe, grâce à des scripts JavaScript.

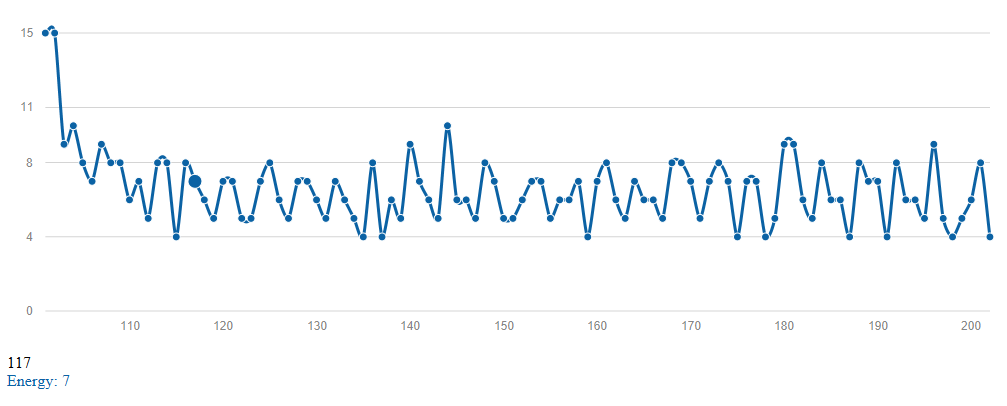
L’utilisateur a le choix de l’affichage des arêtes. Par défaut, elles sont toutes courbées, mais il peut aller dans « Fichier » et décocher « Utiliser un graphe courbé » pour que les arêtes soient droites.

## Exécuter les algorithmes

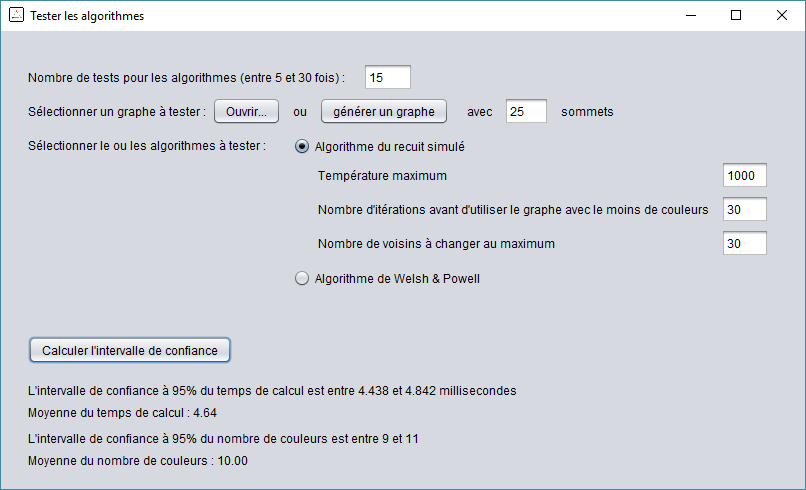
L’utilisateur doit ensuite choisir l’algorithme qu’il souhaite utiliser. Si c’est l’ARS, il doit cliquer sur bouton radio « ARS » en bas de l’écran, ou sur le bouton « Welsh » pour l’AWP. Pour exécuter l’algorithme, il lui faut cliquer sur le bouton « Lancer l’algorithme ». L’affichage du graphe va alors automatiquement se refaire. Le nombre de couleurs obtenu est affiché sur la partie en haut à droite de la fenêtre.

Il existe une option pour l’ARS : l’utilisateur peut aussi observer l’évolution de l’énergie (l’énergie représente le nombre de couleurs dans le graphe) pendant l’exécution de l’algorithme. Pour cela, il doit cocher la case « Ecriture » en bas de l’écran et relancer l’algorithme. L’écriture de l’évolution de l’énergie est coûteuse en temps de calcul, c’est pourquoi nous laissons le choix à l’utilisateur d’avoir accès à cette option ou non.

Le graphique représentant l’évolution de l’énergie est affiché quand l’utilisateur cliquera sur le bouton « Afficher l’évolution de l’énergie ». Ce graphique est également une page web : elle va traduire le texte au format JSON représentant le graphique de l’évolution de l’énergie.



## Tester les algorithmes



Pour obtenir des statistiques sur l’efficacité de chaque algorithme, l’utilisateur peut utiliser la fenêtre de tests, accessible en cliquant sur « Fichier » puis « Tester les algorithmes ».

La première ligne de la fenêtre représente le nombre de fois que l’on va exécuter un algorithme sur un même graphe. Plus ce nombre est élevé, plus nous avons de chances d’avoir des statistiques fiables, mais en contrepartie le temps de calcul sera également plus long.

Pour la deuxième ligne, l’utilisateur peut indiquer quel graphe il souhaite tester.

Il faut ensuite choisir l’algorithme à tester, ainsi que les paramètres à utiliser pour l’ARS.

Enfin, en cliquant sur « Calculer l’intervalle de confiance », l’intervalle de confiance à 95% ainsi que la moyenne du temps de calcul et du nombre de couleurs est affiché en bas de la fenêtre.