Compte-rendu réunion 13/03/2018

1. Travail accompli

* Changement de l’implémentation
  + Création d’une bimap<int, int> pour stocker les ids des cartes correspondant aux ids de la matrice d’adjacence et pouvoir y accéder en O(1) dans les deux sens
  + Vector<Card> : inchangé
  + Vector<int> : pour les cartes qui vont être extraites du fichier
  + Vector<Edge> : pour le stockage des arêtes
* Changements et améliorations :
  + Class Edge :
    - Il n’y a plus que la variable pour le total
    - Idée proposée pour le stockage des valeurs des autres critères
  + Class Card :
    - Stockage des couleurs avec une chaine de bits
    - Vector<int> pour les blocs, éditions, types, sous-types, capacités
  + Class Graph :
    - Le calcul du critère color a été déplacé dans la fonction de création de matrice d’adjacence (Grosse amélioration en temps de calcul)
    - Ajout du calcul des critères édition, bloc, types, sous-types, capacités, total
    - Histogramme du poids des arêtes
* Calcul des critères :
  + Edition : 0 si bloc et éditions différents, 1 si même bloc, 2 si même édition
  + Type et sous-type différenciés (si jamais nous voulions donner des coefficients différents à ces deux critères) :
    - Type : +1 par type commun aux deux cartes
    - Sous-type : +1 par type commun aux deux cartes
  + Capacités : +1 par capacité commune aux deux cartes
  + Total : SOMME (critère\*coefficient\_du\_critère)
    - Pour le moment tous les coefficients sont à 1, ils seront ajustés en fonction des résultats obtenus
* Requêtes SQL :
  + Cartes
  + Blocs + éditions
  + Types
  + Sous-types
  + Super-types (éventuellement à supprimer plus tard si non-nécessaire)
  + Capacités
  + Ids : Pour avoir la correspondance multiverseid ⬄ ids des cartes
* Problème d’optimisation : 2 algos ont été implémentés indépendamment (1 par membre du binôme), ils seront présentés lors de la réunion

1. Sujets abordés

Lors de la réunion nous avons principalement présenté le travail effectué (voir 1ère partie) et expliqué les différents choix que nous avons fait.

Nous sommes tous d’accord qu’une interface graphique nous permettrait d’avoir une meilleure visualisation du deck et des propositions obtenues par l’algo. Ceci nous permettra ensuite de pouvoir ajuster les coefficients des critères ou de modifier notre algorithme.

Nous avons vu que le critère n’était pas pertinent et qu’il faudra donc le retirer (ou mettre son coefficient à 0 en attendant d’avoir une meilleure visualisation, peut-être nous sera-t-il utile plus tard).

Nous avons également discuté et réfléchi communément à propos de nos différents algorithmes.

1. Travail à faire

Le plus important et le plus urgent à faire est une interface graphique qui afficherait les cartes et qui permettrait de constituer un deck en choisissant une carte parmi celles proposées. Pour se faire plusieurs technologies répondant aux deux critères Visual Studio et C++ s’offrent à nous :

* Application de bureau Windows : les problèmes sont :
  + La prise en main
  + Le possible manque de documentation (brèves recherches non concluantes)
  + Pas d’interface graphique pour gérer les différents éléments (comme en c#.NET par exemple)
  + La création d’un nouveau projet
  + Une possible utilisation différente des bibliothèques que nous utilisons
* Projet Qt sous Visual Studio (solution que nous avons choisi)
  + Les problèmes sont :
    - La création d’un nouveau projet (et donc d’un nouveau repo github, problème mineur)
    - Solution en béta pour le moment du côté de Qt
  + Les avantages sont :
    - Projet facilement exportable, bibliothèques boost et MySQL C++ Connector compatibles avec la bibliothèque Qt
    - Une interface graphique pour mettre en place les éléments graphiques
    - Une documentation très fournie et une communauté active
* SFML :
  + Les avantages sont :
    - Pas de création de nouveau projet : librairie à inclure de la même manière que MySQL C++ Connector
    - Documentation garnie et en français
  + Les problèmes sont :
    - Tous les éléments sont à gérer dans le code
    - Pas d’élément d’UI comme des buttons, des labels ou autre, tout est à gérer avec les positions x et y des éléments, ce qui peut devenir vite compliqué si on multiplie les éléments graphiques
* Projet entièrement sous Qt :
  + Les avantages sont :
    - Tous les éléments de Qt à disposition
    - Une interface graphique pour mettre en place les éléments graphiques de notre UI
  + Les problèmes sont :
    - Projet à recréer sous Qt 🡺 changement de bibliothèque pour la connexion à la base de données ou solution différente pour récupérer les cartes dans le programme (fichier texte/JSON)
* Projet Visual C++ CLI/C++ :
  + Les avantages sont :
    - Continuer de travailler sous Visual Studio, donc bibliothèques compatibles
    - Interface graphique pour gérer les éléments graphiques
  + Les problèmes sont :
    - Langage différent : CLI/C++ =/= C++ donc refaire tout le code…

Après avoir passé un certain temps à tester toutes les solutions, celle qui apparait comme correspondant le plus à nos besoins et qui pourrait être développée dans des délais assez courts est le projet Qt sous Visual Studio.

Cette interface graphique devra comprendre les fonctionnalités suivantes :

* La création d’un deck en important les premières cartes
* La sauvegarde/exportation d’un deck
* La visualisation des cartes d’un deck ainsi que des propositions
* L’affichage des caractéristiques des cartes (couleur, édition, type, sous-type, capacités)
* La génération de statistiques (histogrammes au fur et à mesure du déroulement de l’algo)

Et si les délais le permettent :

* Le renseignement des coefficients pour l’évaluation des critères
* La génération de la matrice d’adjacence en fonction des critères renseignés
* Eventuellement d’autres fonctionnalités qui nous viendraient à l’esprit durant le développement

Outre l’interface graphique, nous devons également peaufiner nos algorithmes pour le problème d’optimisation et éventuellement réfléchir à d’autres solutions.

L’une des parties de ce projet concerne l’évaluation du deck généré par nos algos, il faut que nous commencions à nous renseigner sur ce sujet et à réfléchir à comment nous pourrions le faire sachant que nos connaissances dans le jeu Magic sont limitées.