



UNIVERSITÉ  
DE MONTPELLIER



# Rapport de Projet

## Manipulation Interactive des graphes avec SageMath

**Réalisé par**

Aymeric GOUJON

Benoit PRIGENT

Tristan VALADE

**Sous la direction de**

Petru VALICOV

**Pour l'obtention de la Licence Pro ACPI**

*Année universitaire 2019 - 2020*



## Remerciements

## Sommaire

Rapport de Projet	1
Remerciements	3
Sommaire	4
La table des figures	5
Glossaire	6
Introduction	7
Analyse	8
Rapport technique	17
Résultats	18
Gestion de projet	19
Conclusion	20
Référence bibliographique	21
Quatrième de couverture	23

## La table des figures

Figure 1 Présentation de divers utilisation de la console SageMath	7
Figure 2 Diverses commandes de création et édition de graphes	8
Figure 3 Commandes d'affichage d'un graphe en format image depuis la console	8
Figure 4 Graphe crée en format image	8
Figure 5 Graphe créer afficher sur un navigateur	9
Figure 6 Commandes d'affichage d'un graphe sur un navigateur depuis la console	9

## Glossaire

# Introduction

Dans un monde où les mathématiques sont omniprésente et complexes, le développement de logiciels dédié aux mathématiques est en expansion continue. SageMath est un logiciel de calcul symbolique permettant aux étudiants comme aux chercheurs d'appliquer toutes formes de mathématiques.

SageMath propose de nombreuses solutions pour l'édition et la manipulation des graphes depuis la console SageMath. Cependant, SageMath ne possède aucune méthode ou fonctionnalité permettant d'effectuer une réelle édition graphique d'un objet et cela est déroutant pour les utilisateurs, qui ne sont pas familiers avec une console de terminal. On ne peut donc pas éditer visuellement les données d'un graphe avec SageMath.

Notre projet a pour but de simuler la console SageMath à travers une interface graphique simplifiée. C'est à dire que l'utilisateur puisse éditer et manipuler son graphe graphiquement et le plus simplement possible. On doit donc créer une nouvelle méthode d'affichage. Elle doit permettre d'afficher le graphe sur un navigateur en proposant des fonctionnalités d'édition. Ces fonctionnalités que l'on va ajouter sont celles déjà implémentées dans SageMath et accessibles par la console. Nous choisirons seulement les fonctionnalités et commande les plus importantes pour les utilisateurs. Cette extension graphique à pour finalité de ce voir intégré dans le logiciel SageMath et ainsi disponible à tous les utilisateurs.

Pour commencer nous analyserons le logiciel sageMath, sur son fonctionnement est ses solutions existantes, ainsi que les ajouts de fonctionnalités a réalisé pour la réalisation du projet. Dans un second temps, nous rentrerons dans les détails techniques de la réalisation et conception du projet. Ensuite, on s'intéressera à nos résultat obtenu. Et enfin, on parlera de la façon dont nous nous sommes organisés.

# Analyse

Comme expliqué précédemment SageMath a pour but de fournir un logiciel qui permette d'explorer toutes sortes de constructions mathématiques et de faire des expériences en algèbre, géométrie, arithmétique, théorie des nombres, cryptographie, calcul scientifique et dans d'autres domaines apparentés. Nous allons ici voir comment fonctionne SageMath et les fonctionnalités qu'il propose. Nous allons également introduire les modifications que nous apporterons au logiciel afin de répondre à notre problématique.

## 1. analyse du logiciel

SageMath (anciennement Sage pour System for Algebra and Geometry Experimentation) est un système d'algèbre informatique dont les caractéristiques couvrent de nombreux aspects des mathématiques, notamment l'algèbre, la combinatoire, la théorie des graphes, l'analyse numérique, la théorie des nombres, le calcul et la statistique. Et même la création de gif à partir d'une liste d'objets graphiques. En d'autre termes, SageMath facilite l'expérimentation interactive avec des objets mathématiques. La première version de SageMath a été publiée le 24 février 2005 en tant que logiciel libre et open-source sous les termes de la Licence Publique Générale GNU version 2, avec les objectifs initiaux de créer une alternative open-source à Magma, Maple, Mathematica et MATLAB. L'initiateur et chef du projet SageMath, William Stein, est un mathématicien de l'Université de Washington. Son code source est en open source, ce qui signifie qu'il est disponible librement et lisible, de sorte que les utilisateurs puissent comprendre ce que fait le système et l'étendre facilement.

SageMath intègre de nombreux progiciels mathématiques spécialisés dans une interface commune, pour laquelle l'utilisateur n'a besoin de connaître que Python. Cependant, SageMath contient des centaines de milliers de lignes de code uniques ajoutant de nouvelles fonctions et créant l'interface entre ses composants.

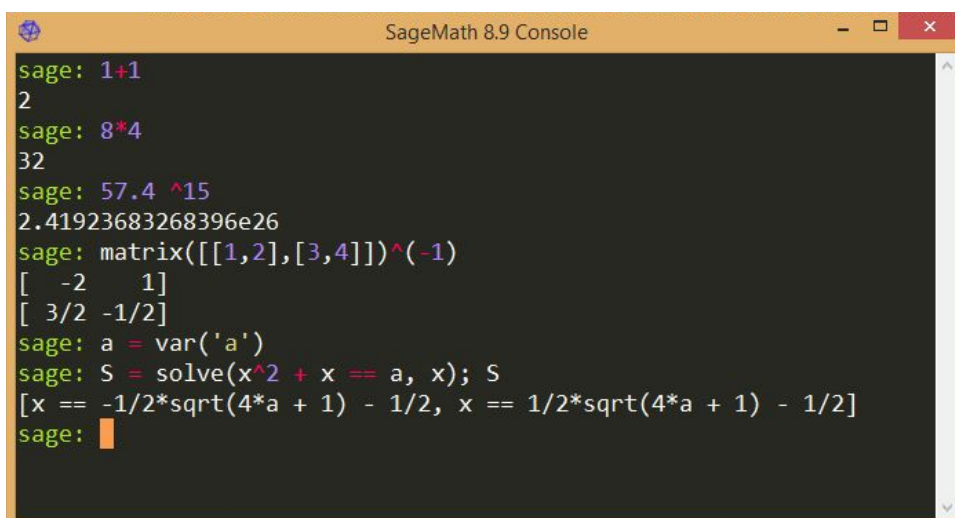


SageMath est un logiciel efficace, il fait appel à des logiciels matures et soigneusement optimisés comme GMP, PARI, GAP et NTL, ce qui le rend très rapide pour certaines opérations. Son code source est facile à compiler pour les utilisateurs de Linux, d'OS X et de Windows. Cela rend le système plus flexible pour les utilisateurs qui souhaitent le modifier.

Des logiciels similaires à SageMath existent, mais ils sont payants. L'objectif général de SageMath est de créer une alternative libre viable à ces logiciels.

Les principaux concurrents payants de SageMath sont : Maple, Mathematica, Magma et MATLAB. Ces logiciels sont bien plus performants et pratiques d'utilisation que SageMath, à l'image de Maple, qui est considéré comme le logiciel mathématique associant le moteur mathématique le plus puissant au monde. Mais cependant SageMath se fait sa place entre ces concurrents directs grâce à son code en open source qui lui permet de se voir rajouter de nombreuses nouvelles fonctionnalités depuis sa création. On estime actuellement à plusieurs millions les utilisateurs de SageMath. La vaste gamme de fonctionnalités proposée par SageMath permet de toucher un public très large comprenant les étudiants (du lycée au doctorat), les enseignants et les chercheurs en mathématiques.

SageMath est un logiciel à l'architecture d'un terminal, dans lequel on peut comme énumérés précédemment lui demander de réaliser une vaste gamme de mathématiques générales et avancées, pures et appliquées.



```
sage: 1+1
2
sage: 8*4
32
sage: 57.4 ^15
2.41923683268396e26
sage: matrix([[1,2],[3,4]])^(-1)
[ -2    1]
[ 3/2 -1/2]
sage: a = var('a')
sage: S = solve(x^2 + x == a, x); S
[x == -1/2*sqrt(4*a + 1) - 1/2, x == 1/2*sqrt(4*a + 1) - 1/2]
sage: 
```

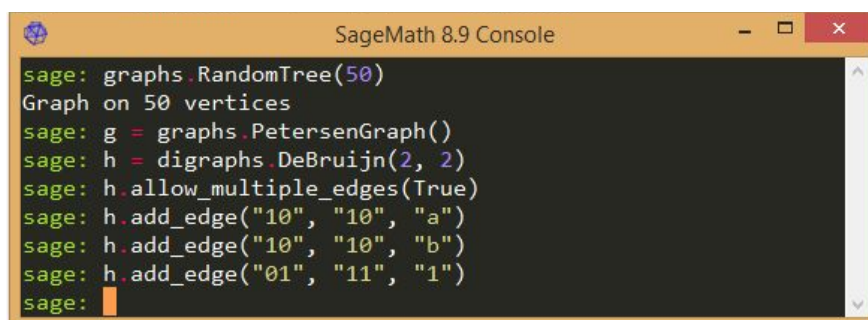
Voici quelques petits exemples simple, avec de basique calculs algébriques, un calcul de matrice et une résolution d'une équation quadratique.

La partie graphe de SageMath est extrêmement vaste et possède plus d'une centaine de milliers de fonctionnalités et de commandes réalisables depuis la console SageMath. Tout d'abord concentrons-nous sur ce qu'est un graphe. Un graphe est une collection d'éléments mis en relation entre eux. Géométriquement, on représente ces éléments par des points (les sommets) reliés entre eux par des arcs de courbe (les arêtes). Selon que l'on choisit d'orienter les arêtes ou de leur attribuer un poids, on parle de graphes orientés ou de graphes pondérés.

Parler de graphe nous renvoie inévitablement à la Théorie des graphes, qui est la discipline mathématique et informatique qui étudie les graphes. La théorie des graphes s'intéresse aux multiples propriétés des graphes.

SageMath propose une solution perspicace de travail aux chercheurs en théorie des graphes.

Voici quelques exemples de fonctionnalités pour les graphes :



```
sage: graphs.RandomTree(50)
Graph on 50 vertices
sage: g = graphs.PetersenGraph()
sage: h = digraphs.DeBruijn(2, 2)
sage: h.allow_multiple_edges(True)
sage: h.add_edge("10", "10", "a")
sage: h.add_edge("10", "10", "b")
sage: h.add_edge("01", "11", "1")
sage: 
```

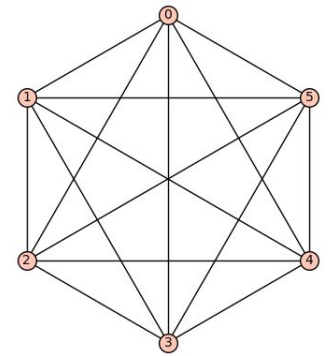
On peut créer des graphes de différents types, leur ajouter des sommets...

SageMath est également capable d'afficher en deux dimensions des cercles, des droites, des polygones, tous types de graphes, des lignes de niveau et des représentations de champs de vecteurs.

Voici un exemple d'affichage d'un graphe complet à 6 sommets, avec à

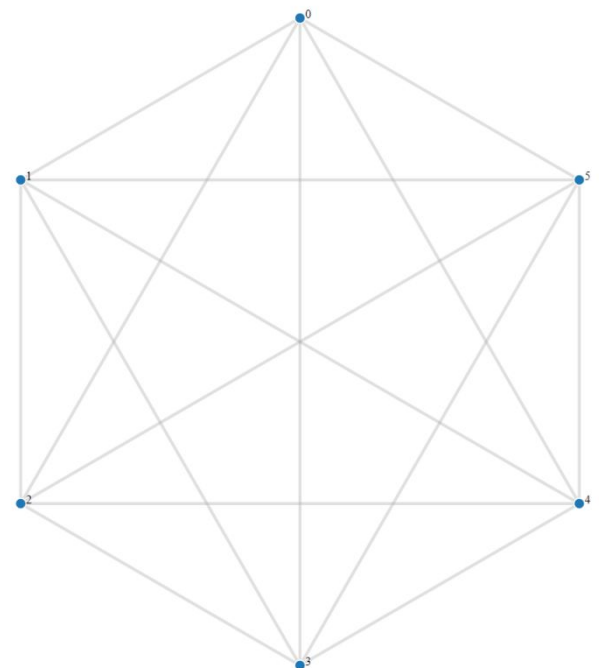
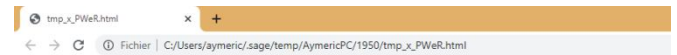
droite l'affichage du graphe en format image :

```
SageMath 8.9 Console
sage: g = graphs.CompleteGraph(6)
sage: show(g)
Launched png viewer for Graphics object consisting of 22 graphics primitives
sage: 
```



On peut également afficher des graphes ou autres sur un navigateur avec la méthode :

```
SageMath 8.9 Console
sage: g = graphs.CompleteGraph(6)
sage: g.show(method="js")
sage: 
```

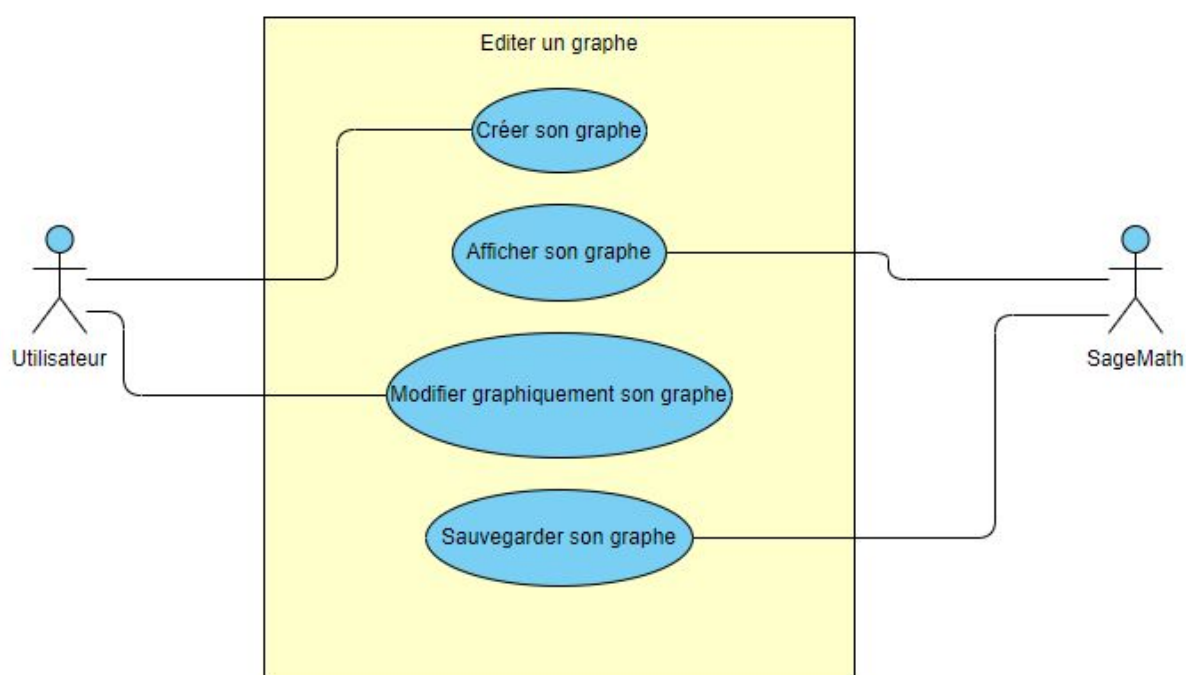


Cette méthode utilise le d3.js pour afficher le graph dans le navigateur par défaut de l'ordinateur. Cette méthode permet en plus la méthode précédente, de pouvoir déplacer graphiquement les sommets du graphe. Mais le déplacement des points n'est pas enregistré et ne modifie pas le graphe créé. C'est à dire que si l'on crée et affiche un graphe, puis que l'on modifie la position de ses sommets et qu'on souhaite le retracer, le graphe s'affichera à nouveau avec la position initiale (avant les modifications) de ses sommets. SageMath possède également de nombreuses autres méthodes d'affichage comme la méthode `show3d()` permettant d'afficher un graphe en 3 dimensions ou encore `plot()` permettant d'afficher un objet graphique, et bien d'autres. Malgré cela, aucune de ces méthodes d'affichage ne permet de répondre à notre besoin d'éditer graphiquement un objet.

## 2. besoin fonctionnel

Fonctionnalités	Demandé	Proposé	Réalisé
Gérer l'ajout et la suppression de nœuds et de lien	X		X
Gérer les forces qui agissent sur le graphe, permettre de freeze/defreeze le graphe	X		X
Faire une mise à jour de sageMath en temps réel pour répercuter les changement visuels sur le graphe	X		X
Répercuter les changements du graphe sur l'affichage JS	X		X
Permettre une sélection multiple avec la souris	X		X
Permettre la subdivision des liens	X		X
Graphe orienté	X		X
Ajout de partition (coloration)	X		X
Permettre le retour en arrière	X		X

Tests de l'intégralité des fonctions	X		X
Arrêter le serveur à la fermeture du dernier client connecté	X		X



Titre :

Lancer l'affichage d'un graphe

Résumé :

L'utilisateur demande l'affichage du graphe

Acteurs

L'utilisateur (principal)

Pré condition

L'utilisateur a téléchargé le dossier de travail

Scénario nominal

1. L'utilisateur ouvre la console sage
2. L'utilisateur attache le dossier de travail avec la commande : attach('localisation du fichier init\_CustomJS.py ')
3. Le système vérifie la validité de l'attache
4. Le système demande à l'utilisateur le chemin du répertoire contenant le dossier de travail
5. L'utilisateur saisie le chemin du répertoire contenant son dossier de travail
6. Le système vérifie la validité du chemin du répertoire
7. L'utilisateur crée un graphe
8. L'utilisateur affiche le graph avec la méthode : show\_CustomJS()
9. Le système ouvre une page Ethernet sur le navigateur par défaut de l'utilisateur et affiche le graphe

Enchaînement alternatif

A1 : L'attache du fichier init\_CustomJS.py n'est pas valide

L'enchaînement A1 démarre après le point 3

4. Le système enregistre l'échec
  5. Le système indique que l'attache du fichier init\_CustomJS.py n'est pas correct
- Le scénario nominal reprend au point 2

A2 : Le chemin du répertoire de travail n'est pas valide

L'enchaînement A2 démarre après le point 6

7. Le système enregistre l'échec
  8. Le système indique que le chemin du répertoire de travail n'est pas correct
- Le scénario nominal reprend au point 5

Titre :

Sauvegarder les modification effectuées sur un graphe

Résumé :

L'utilisateur enregistre les modifications qu'il a réalisé sur le graphe

Acteurs :

L'utilisateur (principal)

Pré condition

Le graphe est créé et affiché

Scénario nominal

1. L'utilisateur modifie graphiquement le graphe
2. L'utilisateur enregistre le graphe
3. Le système créer un fichier .png de l'image du graphe

Scénario alternatif

A1 : Le graphe a déjà été enregistrer

L'enchaînement A1 démarre après le point 2 du scénario nominal

3. Le système enregistre les modifications du graphe

Le scénario nominal ce termine

### 3. besoin non fonctionnel

SageMath combine la puissance de nombreux programmes libres dans une interface commune basée sur le langage de programmation Python.

Mais il possède des extensions utilisant d'autre langage comme JavaScript que l'on avons vu précédemment avec la méthode d'affichage `show(method="js")` utilisant le `d3.js`. Le `d3.js` est une librairie de JavaScript dédié à la manipulation de documents basés sur des données, il permet de donner vie aux données en utilisant des fichiers HTML et CSS. Les graphes prennent vie avec les objets JSON. Le JSON est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée.

Pour résumer, SageMath utilise un ensemble de fichier, avec une base en python, des fichiers JSON, des fichiers en CSS et en HTML, et des fichiers en Javascript permettant de modéliser ces fichiers.



# Rapport technique

- Afficher le graphe dans D3js -> reprise depuis le code existant dans `show(method="js")`
- suppression/ajout de noeuds et arêtes -> depuis d3 car il permet de le faire
- positions d'un graphe qui n'en a pas de base -> simulation d3JS
- annuler une action -> pattern Memento ? non ça marche pas. pattern Command ?  
oui

## Mise en place d'une pile d'action :

Une fois que les opérations de bases fonctionnaient correctement au travers de l'éditeur graphique, il restait à implémenter une fonction très répandue et indispensable des outils d'édition : l'annulation et la reproduction de chaque action. C'est le fameux Ctrl+Z et Ctrl+Y.

Pour cela, il existe des *patterns* (patron de conception) de programmation.

On a tout d'abord envisagé le pattern **Memento** qui selon Wikipedia : "donne la capacité de restaurer un objet dans un état antérieur". Par extension, si l'on est déjà revenu en arrière dans l'historique des actions effectuées sur l'objet, il permet aussi de restaurer un état plus avancé d'un objet. Cependant, ce pattern n'a pas été retenu car il s'intégrait très mal avec D3.JS. D'une part, car D3.JS étant toujours en cours d'exécution, il ne tolère pas que l'on modifie les objets qu'il manipule, car cela crée des incohérences qu'il ne peut résoudre. D'autre part, car les arêtes d'un graphe étant dépendantes des noeuds, l'objet idéal à conserver dans le pattern memento pour garantir la cohérence du graphe était le graphe tout entier lui même, et non chacun de ses éléments. Ce qui amène à détruire, puis réinterpréter l'intégralité du graphe à chaque changement d'état et cela est très lourd en terme de performance.

Ainsi, c'est le pattern **Command** qui a été retenu pour développer cette fonctionnalité. Il a pour particularité d'encapsuler chaque action de l'utilisateur dans un objet "Command" qui va définir, une fonction à exécuter pour réaliser l'action, une autre pour annuler l'action et éventuellement des paramètres pour ces fonctions.

De plus, le pattern Command sauvegarde chacune des commandes effectuées dans une pile d'action, qu'il suffit de dépiler en invoquant la fonction d'annulation pour inverser toutes les modifications faites par l'utilisateur. Enfin, il est très économe en terme de ressources car il se sauvegarde aucun objet modifié, mais seulement les actions effectués par ces objets.

### Extension du pattern Command :

Une fois que la pattern de base a été mis en place. Il a fallu l'améliorer pour qu'il s'adapte au mieux à nos besoins.

La première tâche a été de permettre à l'utilisateur de reproduire facilement les actions qu'il a annulé. Or le pattern Command classique permet seulement de revenir en arrière. Ainsi pour reconstituer les actions annulées, il suffit simplement d'ajouter une deuxième pile d'action au pattern qui elle va contenir les actions annulées et non pas les actions effectuées. Cela amène à se poser des questions assez rapidement, quant aux interactions entre ces deux piles : Est-ce que la reproduction d'une action annulée doit ajouter une commande dans la pile d'action annulable ou reproductible ? Est-ce qu'une action nouvelle, qui interviendrait suite à plusieurs annulations, doit être placée à la fin de la pile d'action annulable ou quelque part à la fin de ces deux piles ? Et si on annule encore cette action là ?

Pour résoudre, ce casse tête on s'est contenté de reproduire le comportement de base d'un éditeur classique tel que Word ou Photoshop. C'est la solution la plus ergonomique pour l'utilisateur, car est ergonomique quelque chose qui paraît naturel et simple d'utilisation, or rien n'est plus naturel qu'un comportement qu'il connaît déjà.

Ainsi, toutes les actions sont annulables puis reproductibles, sauf si l'utilisateur effectue une nouvelle action. Auquel cas, la pile d'action reproductible est vidée afin de garder une cohérence "chronologique" des actions. Concrètement, si vous fait des actions A, B et C et que vous annulez C. Vous pourrez reproduire C, uniquement si vous n'effectuez pas d'action D entre temps. De cette manière, il n'existe qu'un ordre logique soit ABCD (avec reproduction), soit ABD (sans reproduction).

- subdivision d'une arête -> suppression de l'arête, noeud au milieu, ajout de deux arêtes (depuis d3 car seulement des actions de base de d3)
- quand on bouge un noeud, tout le graphe bouge en bordel -> touche pour fixer le graphe/arrêter la simulation après quelques secondes d'affichage
- **coloration d'un graphe (partie graphique)** ->
- récupérer le graphe du navigateur vers Sage (1) -> sauvegarde du navigateur avec une action utilisateur vers un fichier JSON, puis autre action utilisateur pour récupérer le graphe en JSON et en faire un objet Graph dans Sage (fonctionnel mais trop laborieux en terme d'UX)
- récupérer le graphe du navigateur vers Sage (2) -> utilisation d'une classe websocket server écrite par Johan Hanssen Seferidis  
<https://github.com/Pithikos/python-websocket-server>
- mettre à jour le graphe dans Sage -> fonctions d'update qui utilisent les méthodes de la classe Graph et donc permettent de le modifier en en prenant un autre en exemple
- relou de sélectionner en double cliquant -> rectangle de sélection
- ouvrir plusieurs onglets pour des graphes différents -> dictionnaire de graphe/client[id] (plus tard, fermeture de la connexion lorsqu'un client s'ouvre sur un graphe déjà ouvert)
- appel de méthodes de Sage sur le graphe ouvert dans d3JS -> envoi d'un message avec le graphe et un paramètre de demande, en python un dict statique de méthodes permettant de savoir laquelle appliquer sur le graphe, renvoi d'un message dépendant de ce qui était demandé et traitement de la réponse en JS  
|\_> sous-partie pour les différentes fonctions ?
- changement de nature du graphe (graphs/DiGraph) -> création d'un nouveau graphe sauvegardé temporairement avec les mêmes propriétés que le graphe d'origine
- bug sur la comparaison des graph.edges() (ne comparait pas la taille des listes et donc retournait True si l'une des deux était vide) -> signalement à Sage car pas notre problème
- la distance entre les noeuds que génère d3JS empêche de voir les boucles dans le show classique
-

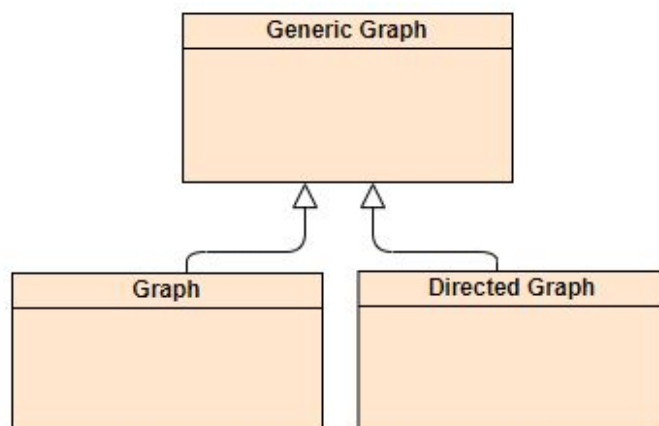


**Besoin utilisateur :** Permettre d'orienter un graphe non orienté

Notre idée initial était que l'utilisateur puisse graphiquement diriger son graphe, on a donc créé un bouton is Directed, qui oriente le graphe de manière "aléatoire". Plus précisément, lors de la création d'une arête, SageMath prend la position d'un sommet et trace l'arête jusqu'à l'autre sommet. On a donc une mémoire de la trajectoire réalisé par le tracé de l'arête, SageMath ce sert donc de cette information pour orienté le graphe.

```
57 function SetIsDirected(bool){
58     isDirected = bool;
59     graphJSON.directed = bool;
60
61     DisplayArrows();
62     UpdateDirectedRelatedElements();
63 }
```

Cette fonction permet donc d'afficher le graphe de façon orienté. Cependant, avec cette méthode on travail sur une copie du graphe, ce qui ne rend pas l'objet orienté. Ce problème est dû à l'architecture du code source de SageMath, puisque on a deux classes, la classe graphe et la classe graphe dirigée qui hérite de la même classe graphe générique, tel que représenté ci-dessous :



## Résultats

### 1) Manuel d'installation

fichier de type “lisez-moi” pour l’installation et l’utilisation du logiciel

### 2) Manuel d'utilisation

### 3) tests de validation

Présentation du code des fichiers de tests commenté créer et affichage des résultats

# Gestion de projet

## Conclusion



## Référence bibliographique

## Annexes techniques

## Quatrième de couverture

SageMath est un logiciel de calcul symbolique, permettant une manipulation des mathématiques. Le projet est une extension du logiciel dont le but est d'ajouter une interaction graphique, pour la partie graphe de SageMath, à partir des fonctions déjà définies du logiciel, ceci dans le but de faciliter la visualisation et la modification des graphes, en la rendant plus rapide et plus accessible aux utilisateurs. En prenant compte que les utilisateurs peuvent être de simple étudiants, comme des chercheurs en mathématiques avancées.

Cette extension a été développée dans le logiciel SageMath. Elle utilise la librairie d3.js (Javascript), le code source en Python du logiciel. La définition et la mise en forme des éléments des pages utilise le CSS et le HTML.

Mot-clés : extension SageMath, graphe, interaction graphique, visualisation, d3.js, Python.

SageMath is a symbolic calculation software, allowing a manipulation of mathematics. The project is an extension of the software whose purpose is to add a graphical interaction, for the graph part of SageMath, from the already defined functions of the software, in order to facilitate the visualization and the modification of graphs, by making it faster and more accessible to the users. Taking into account that the users can be simple students, as well as researchers in advanced mathematics.

This extension has been developed in the SageMath software. It uses the d3.js (Javascript) library, the Python source code of the software. The definition and formatting of page elements uses CSS and HTML.

Keywords: SageMath extension, graph, graphical interaction, visualization, d3.js, Python.