Sorbonne Université
Master ANDROIDE

Partage de gâteaux sur un réseau

UE de projet M1

Lorenzo Carvalho – Karl Cleuziou – Clara Ziegler

2022

Table des matières

Chapitre 1

Introduction

Les relations et connections sociales nous régissent nous et le monde qui nous entoure. La compréhension de ces rapports de force entre individus est étudiée depuis des années et de nombreux modèles et méthodes ont été mis au point dans cet objectif. Notre projet concerne l'un de ces modèles : les graphes de partage. Ces graphes consistent en des nœuds (que nous appellerons agents) représentants des individus, et ces agents sont liés par des arcs non orientés, simulant leurs liens sociaux. Afin de montrer les forces de négociation au sein du graphe ainsi formé, nous permettons aux agents de réaliser un partage avec un de leur voisin, un partage consistant en la répartition d'une unité entre les deux agents (répartition qui n'est pas forcément équitable). Les agents ne peuvent réaliser qu'un seul partage à la fois, et si un agent ne réalise pas de partage alors il obtient un gain nul, ainsi, réaliser un partage, peut importe la quantité obtenue, est forcément meilleur que de ne pas en avoir. Ce type de fonctionnement permet de montrer les forces de négociation entre les différents agents, en effet, le but des agents est d'avoir le plus gros gain possible, et c'est grâce à leur place dans le graphe (leurs liens avec les autres agents) qu'ils pourront négocier et obtenir la plus grosse part possible. Par exemple : un agent en relation avec des agents qui ne sont liés qu'à lui aura une énorme force de négociation et donc pourra avoir le plus gros de l'unité partagée, tandis qu'un agent qui n'est connecté qu'à un unique autre agent sera obligé de baisser sa part dans l'espoir que l'autre agent mieux connecté accepte. Notre projet consiste alors à étudier ces graphes et à réaliser un programme permettant d'obtenir des partages équilibrés entre les agents, c'est à dire des partages reflétant les expérimentations sur l'humain et donc montrant de manière réaliste les forces de négociation entre individus en fonction de leur réseau relationnel et de celui des autres.

Ce dossier présentera notre travail en 4 parties avant lesquelles figure un état de l'art expliquant plus en détail comment le graphe fonctionne, d'où il vient, ainsi que les différents partages particulier qui existent. Les 4 parties sur notre contribution détaillent le fonctionnement du programme, tout d'abord le graphe, c'est la structure indépendante qui permet de modéliser les agents et leurs relations et de calculer les partages spéciaux, ensuite vient l'affichage, qui explique comment nous avons organisé l'interface, les boutons et l'espace de travail dans lequel on peut observer le graphe, enfin vient l'interaction, elle lie les deux premières parties en permettant de manipuler, à travers l'interface, la structure du graphe et de lui apporter les modifications désirées. La 4ème partie porte sur les tests, qui montrent les résultats obtenus avec ce programme. Enfin, nous conclurons

en exprimant les pistes d'améliorations et ce que nous aurions aimé pouvoir faire pour améliorer encore ce projet.

Le programme est accessible depuis ce lien qui vous renvoie vers le github et en annexe, figure un manuel utilisateur pour utiliser le logiciel sans avoir besoin de savoir comment il fonctionne.

Chapitre 2

État de l'art

Comme expliqué dans l'introduction, tous les graphes qui seront mentionnés tout au long de ce document sont composés de noeuds représentant des agents ainsi que d'arcs indiquant les relations entre ces agents. Un partage sera ici un ensemble de paires d'agents, indiquant que ces deux agents ont négocié et se sont mis d'accord pour partager la ressource qui était à leur disposition (dans notre cas nos agents se partageront toujours la valeur 1) mais attention un agent ne peut être impliqué dans un partage qu'avec un seul autre agent. La valeur associée à chaque noeud représente la valeur qu'il a réussi à obtenir grâce à ses négociations. Que l'on parle de sociologie ou d'économie, ce modèle peut être utilisé pour représenter le pouvoir que peut avoir un agent selon sa place dans le réseau et donc les relations qu'il a avec les autres agents. Comme indiqué dans le livre 'Networks, Crowds, and Markets : Reasoning about a Highly Connected World.' écrit par David Easley et Jon Kleinberg[2], ces graphes peuvent représenter différents réseaux et peuvent aussi bien être utilisés pour évaluer le pouvoir d'un agent dans le cadre d'un échange économique aussi bien que son pouvoir au sein d'un groupe d'amis.

Afin de bien comprendre les tenants et aboutissants de notre projet, nous nous sommes aidés de trois articles sur le sujet. Pour comprendre ce que ces graphes peuvent représenter nous avons lu un extrait du livre cité dans le paragraphe précédent. Pour appréhender les notions de partage stable et équilibrés, nous nous sommes intéressés à un article de Jon Kleinberg et Eva Tardos[1] dans lequel ils sont les premiers à nous présenter un algorithme qui calcule de façon efficace les partages équilibrés dans des graphes, article qui a été cité de nombreuses fois depuis, notamment par les auteurs de l'article ' Convergence of Local Dynamics to Balanced Outcomes in Exchange Networks'[3], article d'où vient l'algorithme que nous utilisons pour trouver des partages équilibrés. Dans ces articles, les notions de partage stable et équilibré sont présentés. Un partage stable est donc décrit comme un partage dans lequel il n'existe pas une paire d'agents pour lesquels il serait bénéfique de partager, c'est-à-dire un partage qui augmenterait la valeur des deux agents par rapport à ce qu'ils possèdent déjà. De façon peut être plus facile à comprendre, cela signifie que pour toute paire d'agents (u,v) de valeurs xu et xv qui ne partagent pas ensemble, on a $xu + xv >= 1$. Ainsi lorsqu'un partage est stable, il n'y a plus de négociation entre les agents. Cependant, un partage peut être stable tout en étant injuste et la notion de partage équilibré a pour objectif de rééquilibrer les écarts de gains entre les agents, ainsi le partage équilibré est censé représenter au mieux les résultats expérimentaux obtenus avec de vraies personnes mises dans les conditions pour simuler leur rôle dans différents

réseaux. Un partage équilibré est un partage respectant la solution de négociation de Nash, c'est-à-dire que pour chaque paire d'agents u et v, si on note αu et αv la meilleure alternative de chaque noeud et s_{uv} le surplus valant 1-αu-αv, alors la valeur de u doit être égale à αu+$s_{uv}/2$ et celle de v à αv+$s_{uv}/2$.

Chapitre 3

Contribution

Ce chapitre présente tout notre travail sur le logiciel, comment il fonctionne et les résultats obtenus avec, il est organisé en 4 parties comme précisé durant l'introduction, précédées d'une petite partie justifiant notre choix de langage de programmation.

3.1 Langage de programmation utilisé

Le programme est codé en python .3, ce n'est pas le type de langage idéal pour la réalisation d'un logiciel, cependant, durant le semestre nous suivions chacun l'UE d'Interaction Humain Machine qui nous a introduit à la librairie pyQT, qui permet de réaliser des interfaces graphiques. N'ayant jamais réalisé de programme avec une interface graphique au-paravent, nous ne connaissions pas d'autres alternative que celle que nous étions en train d'apprendre. Afin de gagner un maximum de temps et de ne pas recommencer de zéro l'apprentissage d'une librairie, nous avons décidé d'utiliser cette librairie sur python (QT est, à la base, une librairie réalisée pour C++ mais nous n'avons jamais abordé ce langage). Pour utiliser le logiciel il est donc nécessaire de posséder une version 3 de python ainsi que la librairie pyQT5.

3.2 Le Programme

Notre programme repose sur trois classes principales, Graphe où sont implémentées toutes les méthodes permettant de créer, modifier et tester la stablilité/équilibre des partages, Canvas gérant l'affichage de tout ce qui concerne les graphes et MainWindow qui permet de faire le lien entre ces deux premières classes et de tout afficher dans la fenêtre principale de notre logiciel. Il existe d'autres classes permettant le dialogue avec l'utilisateur, ainsi que des fichiers contenant des graphes qui peuvent être importés dans le logiciel. Logiciel qui s'ouvre en exécutant le Main.

Nous allons donc vous présenter les différents aspects et fonctionnalités de notre logiciel.

3.2.1 Le graphe

Fonctionnalités de bases

Afin de pouvoir implémenter nos graphes nous avons choisi de créer une classe qui nous permet de créer un graphe et mais qui propose également de nombreuses méthodes que nous détaillerons un peu plus tard. Etant donné notre choix d'implémentation pour l'affichage, cette classe Graphe a permis une manipulation plutôt simple des graphes.

Pour commencer, un graphe se caractérise par trois listes, toutes vides lors de la création du graphe. Les deux premières listes contiendront respectivement les noms des différents noeuds ainsi que leur valeur et les arcs, où chaque arc est représenté par un tuple (u,v) dans lequel u et v sont des noeuds du graphe. Notons ici que nous n'affectons pas de valeur aux arcs puisque nous ne considérons que le cas où la valeur à partager sur chaque arc est de 1. De son côté, la troisième liste a pour vocation de contenir les arcs faisant partie du partage, c'est-à-dire les arcs reliant deux noeuds partageant la ressource qui est à partager sur l'arc. Ces trois listes seront ensuite utilisées dans les différentes méthodes de la classe, que ce soit pour avoir des informations sur les noeuds ou modifier le graphe.

Puisque nous n'avons que trois listes pour garder toutes les informations concernant notre graphe, nous avons implémenter des méthodes permettant d'obtenir des informations sur les noeuds comme leur valeur, qui correspond à la valeur qu'il a réussi à obtenir en négociant avec ses voisins, leurs voisins ou savoir s'il est impliqué dans un partage. Pour les algorithmes que nous vous présenterons plus tard, nous avions également besoin de savoir qui de ses voisins lui propose la meilleure offre et la valeur de celle-ci. Toutes ces méthodes qui reposent sur les parcours de nos listes nous apportent toutes les informations dont nous avons besoin sur les noeuds.

Qu'il s'agisse de la création de graphe ou de modification lors des calculs de partage stable, il est indispensable de pouvoir modifier les noeuds, leur nom et valeur, les arcs mais aussi les partages. Pour ce faire, nous avons des méthodes permettant d'ajouter/supprimer des noeuds, arcs et partages et d'autres permettant de changer le nom d'un noeud, sa valeur ou encore de modifier la valeur d'un partage entre deux noeuds. Toutes ces méthodes de manipulation de graphe ont ensuite été utilisées pour créer des graphes aléatoirement, en effet il suffit de choisir un nombre de sommet n ainsi qu'une probabilité p pour que chaque arc existe. Cette méthode de génération aléatoire fonctionne de la façon suivante, n sommets sont créés, ensuite pour chaque paire de sommet u et v, on tire un réel aléatoire et si celui-ci est inférieur à p on créé l'arc (u,v). De plus, comme on travaille ici sur des partages dans un réseau si un noeuds n'est relié à aucun autre on le relie au premier ou deuxième noeud du graphe (s'il s'agit du premier noeud). Il est ensuite possible de générer un partage aléatoire, cette méthode parcours les arcs de manière aléatoire et affecte des valeurs aléatoires sommant à 1 aux deux noeuds si aucun d'entre eux ne fait partie d'un partage.

Partages stables et équilibrés

Maintenant que toutes les fonctionnalités de bases sont crées nous allons les utiliser pour s'intéresser aux problèmes de partages stables et équilibrés.

On va d'abord s'intéresser aux partages stable, et premièrement à comment nous vérifions si un partage est stable. Notre fonction parcours tous les arcs du graphe, on a donc a

chaque fois un arc entre deux sommets u et v ayant respectivement *val1* et *val2* comme valeur. Si un partage entre u et v serait bénéfique aux deux, on dit que l'arc ainsi que u et v ne sont pas stables, dès qu'il existe un arc non stable, le partage n'est pas stable. Pour savoir si le partage leur serait bénéfique, on regarde si ils pourraient tous deux obtenir une valeur plus élevée tout en se partageant la valeur 1, on regarde alors si (1-*val2*) est supérieur à *val1* et (1-*val1*) supérieur à *val2* ou non. Si le partage est stable, la fonction renvoie vrai avec une liste vide, sinon faux avec une liste contenant les noeuds pas stables et une autre contenant les arcs non stables.

Si le partage n'est pas stable on peut essayer de le rendre stable, en effet dans certains il n'existe pas de partage stable le graphe. Par exemple dans le graphe ci-dessous, étant donné le fait que les noeuds ne peuvent partager qu'avec un seul autre noeud, il y aura forcément un noeud mis à l'écart qui aura donc une valeur égale à 0, de ce fait, au moins l'un des deux autres noeud aura intérêt à partager avec le noeud exclu. Cela est vrai car si un noeud n'est pas intéressé cela signifie qu'il a déjà la valeur maximale, ici 1, mais cela veut aussi dire que le noeud avec qui il partage a une valeur égale à 0, il aura donc intérêt à partager avec le noeud exclu.

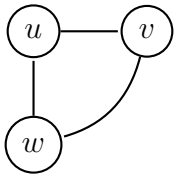


FIGURE 3.1 – Un graphe sans partage stable

Regardons maintenant comment nous allons essayer de rendre un partage stable. Tout d'abord pour cela nous avons besoin d'une méthode qui effectue une itération du processus pour rendre le partage stable, elle prend en paramètres les listes de noeuds et arcs non stables retournés par la fonction qui teste si le partage est stable. Cette fonction parcours les arcs non stables tant qu'il y en a et si les deux noeuds concernés n'ont pas encore changé de partage, on les fait partager la ressource de sorte à ce qu'ils se partagent le surplus équitablement. Si on a noeuds u et v de valeurs *val1* et *val2*, le surplus est 1-*val1*-*val2*. Comme vu dans l'article, 'Balanced Outcomes in Social Exchange Networks' de Jon Kleinberg et Eva Tardos il s'agit de la solution de négociation de Nash et puisqu'ils obtient chacun une valeur supérieure à la précédente, on tend vers un partage stable. On n'oublie pas de supprimer les partages qui n'existent plus et faire en sorte que si deux noeuds voisins ont perdu leur partage et se retrouvent avec une valeur égale à zéro alors ils partagent équitablement la ressource.

On appelle alors cette fonction tant que le partage n'est pas stable et que le nombre maximum d'itérations n'est pas atteint. On donne également le choix d'affiche chaque itération ou non, dans le cas où l'utilisateur veut afficher toutes les itérations, celui-ci peut mettre en pause et reprendre l'exécution lorsqu'il le souhaite et si le partage prend trop de temps avant de se stabiliser (ou s'il n'existe pas de partage stable pour ce graphe), il peut mettre fin à l'exécution et peut reprendre sans les affichages pour accélérer le processus.

Maintenant que nous avons vu le cas des partages stables, intéressons nous aux partages équilibrés et pour ce faire nous avons implémenté l'algorithme d'équilibrage des arcs présenté dans l'article 'Convergence of Local Dynamics to Balanced Outcomes in Exchange

Networks'.

On propose tout d'abord de pouvoir vérifier si un partage est équilibré ou non, or un partage est équilibré si tous les arcs faisant partie du partage sont quasi-équilibrés ou saturés. Pour vérifier si un arc est saturé il suffit de vérifier si l'un des deux noeuds impliqué dans le partage a une valeur égale à 0. Pour vérifier si un arc est quasi-équilibré, la procédure est un peu plus compliquée, il faut vérifier que la valeur *val1* du premier noeud est égale à $\alpha 1$+surplus/2, on rappelle que $\alpha 1$ correspond à la meilleure offre extérieure que peut avoir le premier noeud et que le surplus est égal au poids de l'arc (ici 1) moins la valeur de la meilleure alternative de chaque noeud ($\alpha 1$ et $\alpha 2$). Si tous les arcs du partages vérifient ces deux propriétés, alors le partage est équilibré. Si ce n'est pas le cas alors nous pouvons faire appel à une fonction qui va essayer de rendre le partage équilibré, cette fonction correspond à une itération et est appelée tant qu'il reste des arcs non équilibrés dans le partage et que le nombre maximum d'itérations n'est pas atteint. Elle choisit aléatoirement un arc non équilibré et le rend équilibré, voici le pseudo code correspondant :

Entrées *liste des arcs non équilibrés et non saturés*;
Résultat : équilibre un des arcs choisi aléatoirement
Choix d'un arc uv aléatoire;
Soient αu et αv les meilleurs alternatives pour u et v ;
Soit suv le surplu de l'arc uv ;
Soit wuv le poids de l'arc uv;
$xu' = \alpha u + suv/2$;
$xv' = \alpha v + suv/2$;
if $xu' < 0$ **then**
  $xu = 0$ et $xv = wuv$;
end
else if $xv' < 0$ **then**
  $xv = 0$ et $xu = wuv$;
end
else
  $xu = xu'$ et $xv = xv'$;
end

Algorithme 1 : Equilibrage des arcs

3.2.2 L'affichage

L'affichage du logiciel est entièrement réalisé par la librairie pyQT, mais il est géré de deux manière différentes, la première concerne les boutons, les menus et les fenêtres et la seconde concerne l'affichage du graphe. L'affichage général suit l'affichage classique des logiciels avec les menus et barre d'outils en haut à gauche, et la zone de travail, elle est juste en dessous et occupe tout le reste de l'espace disponible dans la fenêtre afin d'avoir une vision la plus large possible sur le graphe.

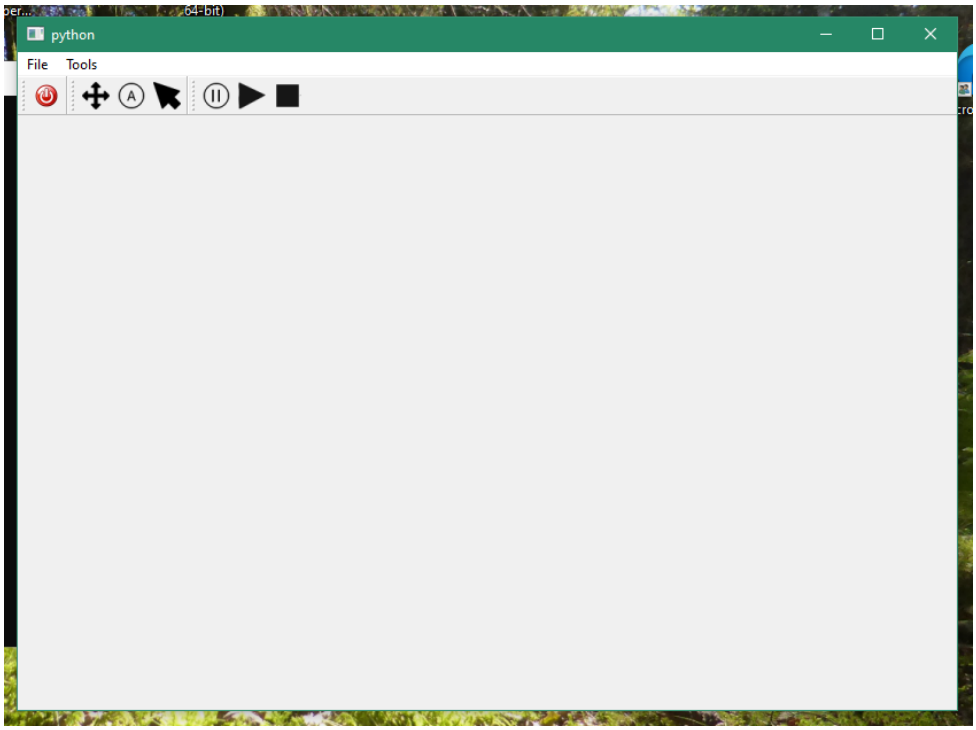


FIGURE 3.2 – Capture de la fenêtre du logiciel

L'affichage de tous les éléments graphiques redondant dans les logiciels est directement tiré des fonctions proposées par pyQT, cela concerne la fenêtre principale, les menus ainsi que les barres d'outils. Nous avons cependant choisi d'organiser l'accès aux commandes selon une certaine logique qui suis les normes et habitudes de tout logiciel. Nous avons un premier menu qui se nomme 'File' et qui regroupe toutes les méthodes liées à l'ouverture, l'exportation, l'importation de fichiers, mais qui n'ont pas d'impact sur le graphe actuellement ouvert.

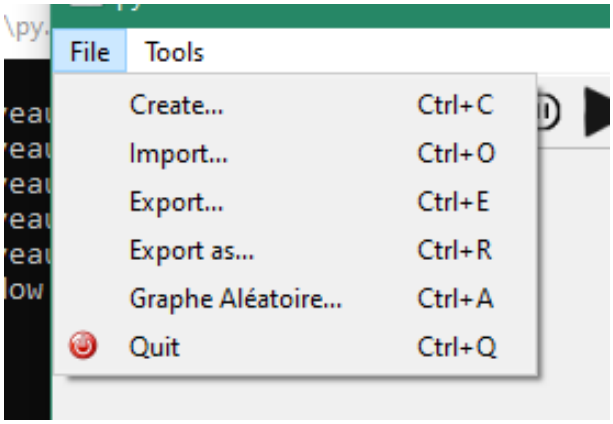


FIGURE 3.3 – Capture d'écran du menu "File"

Puis vient un deuxième menu nommé "Tools" qui, lui, possède les commandes permettant de modifier le graphe ouvert, en particulier, les commandes qui sont peu utilisées : les commandes qui appellent des algorithme automatiques qui vont modifier les partages afin d'atteindre un matching donné (balanced outcome et stable outcome). Les commandes des menus possèdent des raccourcis clavier fin que les utilisateurs ne soient pas obligés d'aller chercher la commande si jamais ils l'ont appris.

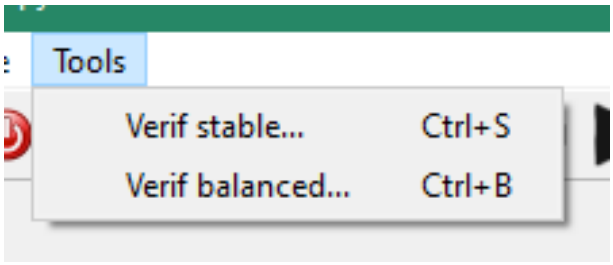


FIGURE 3.4 – Capture d'écran du menu "Tools"

Quant aux autres fonctionnalités, qui sont très souvent utilisées, nous les avons mises dans la barre d'outil afin qu'elle soient très rapidement accessibles avec des icônes facilement reconnaissables, la barre d'outil étant aussi un élément intégré à pyQT.

FIGURE 3.5 – Capture d'écran de la barre d'outils

Tout le reste de l'espace de la fenêtre est donc consacré à l'affichage du graphe, cet affichage est permis également grâce à un élément de pyQT : le canvas. Un canvas est un widget (les widgets sont les différents éléments disposés dans la fenêtre, les menus et la barre d'outil sont des widgets par exemple) qui permet d'y "dessiner" des formes, d'y faire apparaître des éléments graphiques en 2 dimensions. C'est grâces aux méthodes du canvas que nous pouvons afficher le graphe. Le canvas fonctionne comme une grille / un tableau en 2 dimensions, chaque pixel du canvas est accessible via des coordonnées uniques. L'origine (le point/pixel (0, 0)) se trouve tout en haut à gauche du canvas et c'est donc par rapport à cette origine que la position de chaque élément visuel est déterminée.

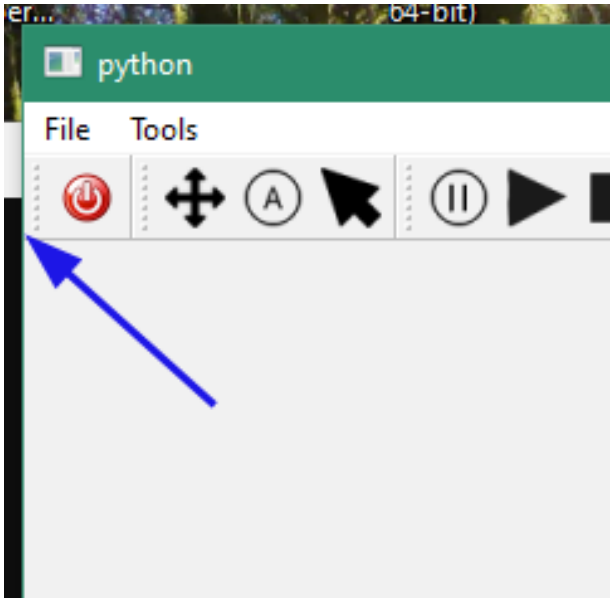


FIGURE 3.6 – La flèche bleue pointe sur l'origine du canvas

Pour afficher les différents éléments du graphe il faut que les noeuds de ce dernier possèdent des coordonnées par rapport à l'origine du canvas, chaque noeud est donc enregistré avec

ses coordonnées dans un dictionnaire du canvas. Les arcs sont également enregistrés dans un dictionnaire du canvas avec un booléen indiquant si oui ou non ils sont concernés par un partage. L'affichage des éléments sur le canvas se fait un élément à la fois, nous affichons donc dans un premier temps les arcs entre les noeuds, pour ce faire nous récupérons les coordonnées des noeuds puis nous traçons des lignes noires pour les arcs classiques et rouges pour les arcs qui sont utilisés dans un partage. Les arcs sont tracés en premier car ce sont les moins importants visuellement, les éléments dessinés sur le canvas se superposent si jamais il partagent des pixels en commun, et c'est l'élément qui a été dessiné en premier qui est recouvert par le plus récent, les arcs sont moins importants à voir que les noeuds, en effet il est préférable qu'un noeud recouvre un arc plutôt que l'inverse car un arc rendrait la lecture du nom d'un noeud difficile par exemple, et en plus comme on fait partir les arcs depuis le centre des noeuds, si les arcs recouvraient les noeuds, ces derniers seraient illisible.

Une fois que les arc sont affichés, ce sont les curseurs de partage qui le sont, les curseurs sont des petits points rouge et noir qui apparaissent sur les arcs rouges (donc les arcs utilisés par un partage), ces curseurs permettent de savoir d'un coup d'oeil à peut près quelle proportion de l'unité partagée les noeuds possèdent, en effet ils sont placés sur l'arc de manière à ce que la longueur entre chaque noeud et le curseur de son partage soit proportionnelle par rapport à la taille totale de l'arc selon la proportion obtenue de l'unité par chaque noeud lors du partage. Par exemple, dans un partage à 0,5 - 0,5 le curseur est placé au milieu de l'arc car chaque noeud possède la même quantité, par contre dans le cas d'un partage à 0,2 - 0,8, le curseur sera très proche du noeud qui a 0,2 et beaucoup moins du noeud qui a 0,8 (si l'arc entre les deux noeuds est de 100 pixels, alors le curseur sera placé sur le 20ème pixel en partant du noeud qui a 0.2, ainsi il sera à 80 pixels du noeud qui a 0.8, de cette manière il y a 0.2 fois la longueur de l'arc entre le noeud qui a 0,2 et le curseur et 0,8 fois la longueur de l'arc entre le noeud qui a 0,8 et le curseur, la distance entre le noeud et le curseur représente la quantité obtenue par le noeud lors du partage).

Enfin, les noeuds sont affichés, on dessine des cercles à partir de leur coordonnées dans lesquels on inscrit leur nom (le nom d'un noeud n'est affiché qu'à l'intérieur de son cercle, si jamais il est trop long il sera forcément rogné afin qu'il ne dépasse pas) et au dessus desquels on inscrit en rouge la valeur qu'ils possèdent (qu'ils réalisent un partage ou non, dans le cas où un noeud ne réalise aucun partage il affiche donc 0). La phase d'affichage sur le canvas est alors terminée, bien que chaque élément soit affiché l'un après l'autre dans un ordre bien précis, en réalité l'affichage est tellement rapide qu'il n'est pas visible à l'oeil humain (pour le nombre de noeuds et d'arc sur lesquels on a réalisé nos tests en tout cas). La phase d'affichage est d'ailleurs appelée avec la fonction update() qui efface tout l'affichage précédant avant d'afficher la nouvelle version, update est donc appelée après chaque modification du graphe que l'on détaillera dans la partie portant sur l'interaction.

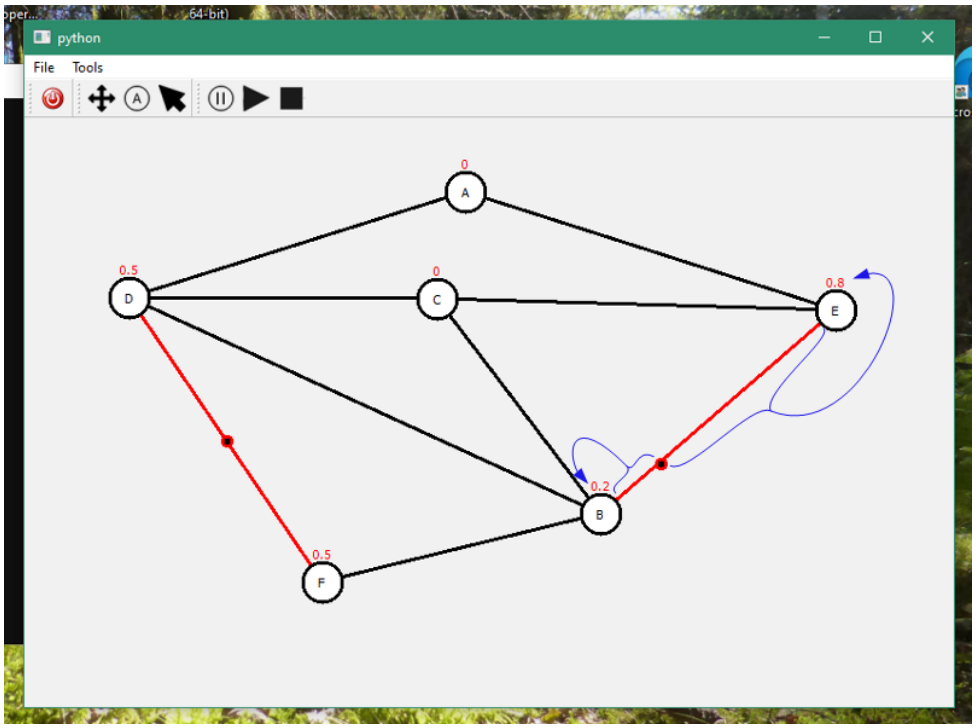


FIGURE 3.7 – Affichage d'un graphe avec ses partages, les flèches bleues illustrent la proportion du partage retranscrite par le curseur

Pour terminer la partie affichage et introduire la partie interaction, il existe des affichages particuliers qui surviennent suite à une interaction de la part de l'utilisateur, ce sont les boîtes de dialogues et la création d'un futur arc. Les boîtes de dialogue sont intégrées à pyQT, il y a juste à appeler la fonction qui permet de les générer, elles sont paramétrées comme des fenêtres classiques. Pour la création d'un nouvel arc, il faut relier un noeud avec un autre (ce fonctionnement sera détaillé dans la partie suivante), et afin de se rendre compte de l'arc que l'on est en train de créer, nous affichons l'arc en bleu, de cette manière il attire l'oeil (seul élément bleu de l'écran, le reste est gris, blanc, noir ou rouge, donc très visible par le contraste) et on comprend que c'est un arc particulier. Lorsque les deux arcs ont été reliés l'arc devient noir comme les autres, si au final on ne relie aucun noeud, il disparaît.

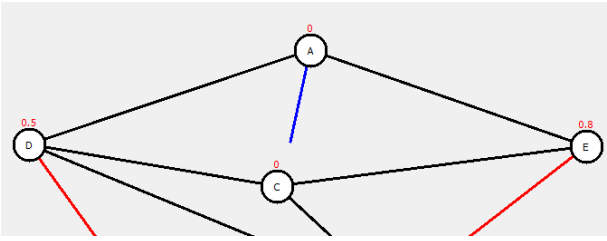


FIGURE 3.8 – On observe ici l'arc bleu qui part de A et qui s'apprête à le relier à C

Enfin, le graphe n'est jamais dépendant de l'interface et de l'affichage, le graphe, son fonctionnement, les méthodes qu'il possède, sont totalement indépendantes et auto suffisantes afin d'éviter un maximum de problèmes. L'affichage, lui par contre, est dépendant du graphe pour fonctionner, en effet il a besoin de récupérer des informations du graphes pour se mettre à jour et l'afficher, et un graphe codé d'une autre manière ne pourrait probablement pas être affiché par ce programme, à l'inverse il est possible de créer d'autres

interfaces pour la structure de ce graphe ou bien d'utiliser ses méthodes sans avoir d'affichage.

3.2.3 L'interaction

Les différentes fonctionnalités sont divisées en 3 parties : l'importation/exportation de graphes, les opérations réalisables sur les graphes et la barre de tâche. Dans la partie importation de graphe, on peut soit réinitialiser la fenêtre, soit importer aléatoirement un graphe en réglant le nombre de noeuds ainsi que la probabilité que chaque noeud soit relié à un autre, soit importer un graphe directement depuis un fichier texte qui doit évidemment avoir un format bien spécifique. Pour l'exportation il y a 2 modes différents : l'exportation classique, qui consiste à exporter le graphe sous la forme de fichier texte en choisissant son nom ainsi que son dossier de destination, ou l'exportation rapide qui exporte le graphe au format txt dans un dossier spécial graphe à la racine du projet, tout en lui donnant un nom générique. Le format txt des graphes est le suivant : sur la première ligne on indique le nombre de noeuds du graphe, sur la deuxième ligne on indique le nombre d'arcs du graphe. Ensuite sur les lignes suivantes, on écrit le nom de l'arc ainsi que son partage associé (de la forme "A,0.3" par exemple), évidemment le nombre d'arcs doit correspondre au nombre écrit sur la première ligne. Et enfin c'est la même chose pour les arcs, un arc par ligne de la forme "A,B" qui connectera donc ces 2 noeuds, et le nombre d'arc doit correspondre au nombre écrit sur la deuxième ligne. Donc pour l'exportation, qu'elle soit rapide ou non, on va décomposer le graphe en couple noeuds-valeurs, et on va récupérer les arcs pour créer un fichier texte qui corresponde également à ce format.

Au niveau des opérations réalisables sur les graphes, on peut soit vérifier s'il est stable, soit s'il est équilibré. S'il l'est, l'utilisateur a un message de validation, sinon on lui propose de le rendre stable (ou équilibré). Si l'utilisateur ne veut pas, on ferme la fenêtre et on revient sur le graphe, sinon une nouvelle fenêtre va s'ouvrir pour lui demander s'il veut voir chaque itération ou seulement le résultat final. S'il décide de voir seulement le résultat final on applique la méthode de stabilité vue précédemment sur le graphe. S'il décide de voir chaque itération, alors on va récupérer chaque étape de la stabilisation qu'on affichera avec une pause de quelques secondes. L'affichage se terminera quand le graphe sera stable ou que la limite d'itération soit atteinte sans que le programme ait réussi à déterminer un partage stable. Si le programme ne trouve pas de partage stable après la limite, alors un message s'affichera qui dira qu'il est impossible de rendre ce graphe stable. Sinon, s'il y parvient, un message affichera que le graphe est stable ainsi que le nombre d'itérations qu'il a fallu pour y parvenir. Ceci explique donc le fonctionnement du partage stable, le partage équilibré fonctionne exactement de la même manière, seul la méthode de calcul diffère, et c'est donc la méthode de recherche de partage équilibré vue précédemment.

Enfin il existe d'autres fonctionnalités en rapport à ces itérations, elles se situent sur la barre de tâches et sont les boutons pause, marche et arrêt. Tous ces boutons sont utilisables seulement quand une recherche de partage (stable ou équilibré) est lancée et que l'utilisateur a choisi de voir les itérations. Appuyer sur pause suspend la recherche, marche reprend la recherche et stop arrête la recherche. A noter que le bouton pause ne réinitialise pas le compteur d'itérations affiché en fin de calcul, alors que le bouton stop le réinitialise. Leur méthode de fonctionnement est simple, appuyer sur un de ces boutons change la valeur d'une des variables globales de la fenêtre, ces variables sont utilisés dans les méthodes de recherche de partage et si leur valeur change, les méthodes auront donc

un comportement différent en fonction de la valeur de ces variables.

Il est également possible de modifier le graphe ouvert dans le programme au travers du canvas précédemment présenté dans la partie sur l'affichage, ainsi il est possible de déplacer les noeuds, les supprimer, en créer de nouveaux, de nouveaux arcs etc... Toutes ces modifications du graphes sont accessibles via trois modes que nous avons défini : le mode Move, le mode Draw et le mode Select. Les interactions dans ces modes se font uniquement dans le canvas en cliquant sur les éléments avec le curseur de la souris.

Dans le mode Move, comme son nom l'indique, il est possible de déplacer les noeuds du graphe (un par un) dans l'espace afin de mieux l'organiser ou bien de déplacer tout le graphe à la fois. Pour déplacer un noeud unique l'utilisateur doit cliquer sur le noeud, sans relâcher le bouton de la souris, puis déplacer la souris, et relâcher le bouton. Cela est possible grâce à trois méthodes de la classe canvas, ces trois méthodes sont déclenchées lors d'une interaction avec la souris et permettent de récupérer la position du curseur au moment où la méthode à été appelée. La première méthode se déclenche au moment où l'on clique sur le bouton, dans le cas du mode move, on regarde si la position de la souris correspond à celle d'un noeud (si la souris est sur un noeud), si c'est le cas, alors on enregistre le nom de ce noeud. Le noeud est donc sélectionné, il est possible de le bouger grâce à la seconde méthode qui se déclenche lorsque l'on a le bouton de la souris appuyé et que cette dernière bouge. Comme pour la fonction précédente, on récupère la position du curseur au moment où la méthode est déclenchée, mais cette fois-ci, on ne regarde pas si oui ou non notre curseur est présent sur un noeud, étant donné qu'on est dans le cas où l'on bouge la souris, soit on a déjà un noeud sélectionné, soit il n'y en a pas. On se concentre, pour le moment, dans le cas où un noeud est déjà sélectionné. Comme un noeud est sélectionné et que l'on a commencé à bouger la souris, on va pouvoir déplacer notre noeud. Afin que le déplacement du noeud soit le plus agréable et normal possible, on ne donne pas les coordonnées récupérées au moment du déclenchement de la méthode au noeud, cela créerait un décalage bizarre. En effet, la position d'un noeud est donnée par rapport à un point qui est l'origine du noeud, sauf que lorsqu'on a cliqué sur le noeud pour le déplacer, on n'a pas forcément cliqué sur l'origine du noeud, donc si on donnait au noeud la position du curseur le noeud subirait une petite téléportation en faisant concorder son origine avec le curseur de la souris, ce qui nous a paru assez désagréable à l'usage. Donc au lieu de faire cette translation, on calcule la différence entre le point précédent récupéré par la méthode de la souris et le point actuel et on ajoute cette différence à notre noeud, ainsi il se déplace exactement de la même manière que la souris. Puis lorsqu'on relâche le bouton de la souris, cela appelle la 3ème méthode qui est donc déclenchée par le lâché du bouton, on ne souhaite plus bouger le noeud, donc à ce moment là on désélectionne le noeud. Pour déplacer le graphe en entier il faut cliquer dans le vide (un endroit du canvas où il n'y a ni noeud, ni arc, ni curseur de partage), cela va déclencher la première fonction d'interaction de la souris, pour le noeud unique on vérifiait si le curseur pointait un noeud, comme il s'agit de la même fonction, on fait la même chose, mais comme rien ne sera sélectionné le comportement de la deuxième fonction sera différent. En effet, on se retrouve dans la situation où on a bougé la souris avec le bouton appuyé mais sans rien de sélectionné, on va donc adopter un comportement différent qui permettra de faire bouger tous les noeuds du graphe à la fois. Le plus simple serait de déplacer l'origine du canvas, si tous nos éléments graphiques sont positionnés par rapport à ce dernier, le déplacer permettrait de tout bouger d'un seul coup. Ce n'est malheureusement pas possible, l'origine du canvas ne peut pas être déplacée pour résoudre

ce problème, mais nous n'avons, en réalité, pas positionné nos élément par rapport à l'origine du canvas, nous avons créé, au démarrage du logiciel, un point virtuel et invisible qui nous sert d'origine lorsque l'on fait l'affichage du graphe, et cette origine secondaire est elle même positionnée par rapport à la vraie origine du canvas. Nous n'avions pas précisé l'existence de cette origine intermédiaire avant car ce n'était pas nécessaire pour comprendre le fonctionnement des fonctions précédentes, la seule différence est que leur coordonnées sont par rapport à un point qui, lui, est réellement positionné par rapport à l'origine du canvas. Donc le graphe à afficher est en fait positionné par rapport à une origine que l'on peut déplacer, nous utilisons donc le même procédé que pour le noeud sélectionné pour bouger l'origine virtuelle qui va donc indirectement faire bouger tous les autres éléments du graphe de la même manière.

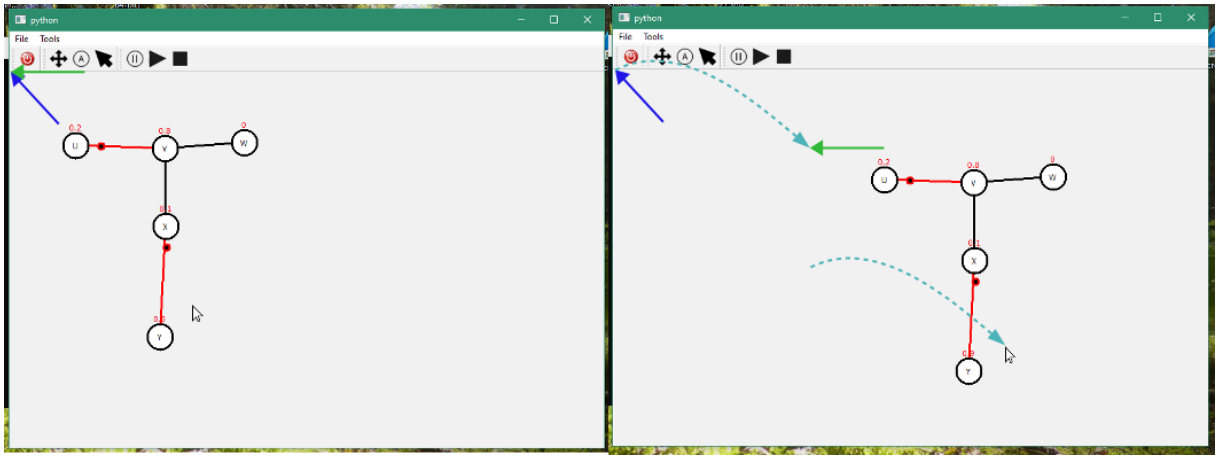


FIGURE 3.9 – Déplacement de l'intégralité du graphe, la flèche bleue pleine indique la position de l'origine du canvas, la flèche verte pleine indique la position de l'origine intermédiaire, la flèche turquoise en pointillés montre la translation effectuée par le curseur qui se répercute sur l'origine intermédiaire et qui du coup se répercute sur la position de l'intégralité du graphe dans la fenêtre et le canvas

Le mode Move est le seul qui ne modifie jamais le graphe d'origine, en effet seule la position des noeuds est modifiée, or la structure du graphe n'enregistre aucune position, c'est uniquement la partie affichage du logiciel qui possède des positions pour chaque noeuds afin de les afficher.

Vient ensuite le mode Draw, et comme son nom l'indique, il permet de dessiner, d'ajouter des éléments au graphe. Cette fois le graphe d'origine sera bel et bien modifié. Comme pour le mode Move, le mode Draw s'utilise de 2 façons, une première façon qui permet d'ajouter de nouveaux noeuds, et une seconde façon qui permet d'ajouter de nouveaux arcs, qui a été légèrement évoquée dans la partie précédente sur l'affichage. Pour ajouter un nouveau noeud il suffit simplement avec la souris de cliquer sur un endroit vide du canvas, comme pour le mode Move la fonction du click de la souris se déclenche, et comme nous ne sommes pas en mode Move mais en mode Draw, ce que la fonction va faire sera légèrement différent, en mode draw, on vérifie également si on clique sur un noeud, et si c'est le cas au lieu de sélectionner le noeud pour pouvoir le déplacer, on crée un faux arcs temporaire qui consiste simplement en l'enregistrement du nom du noeud et de la position du curseur au moment du clique. Mais là on se concentre sur la création d'un nouveau noeud, donc nous considérons que nous n'avons pas cliqué sur un noeud, si l'on bouge la souris il ne se passe presque rien, on continue d'enregistrer la position courante

du curseur lorsqu'il est cliqué mais c'est tout, ça n'a aucun impacte sur la création du noeud. Enfin lorsque l'on relâche le bouton de la souris, la dernière fonction de la souris est appelée et comme on n'a pas de faux arc de construction, on crée un nouveau noeud à l'emplacement où le curseur pointe au moment où l'on relâche le bouton. Le noeud se voit attribuer en nom la première lettre de l'alphabet qui n'est pas utilisée par un autre noeud. Maintenant, reprenons après l'exécution de la première fonction du clique de la souris, si l'on considère que l'on a cliqué sur un noeud, alors on a enregistré le nom de ce noeud et la position du curseur au moment du clique. En maintenant le bouton de la souris appuyé il es possible de la déplacer et de faire apparaître le faux arc de construction en bleu qui est en fait un guide permettant à l'utilisateur de bien se rendre compte de ce qu'il fait. Les informations du faux arcs sont renouvelées avec la position du curseur à chaque appel de la fonctions de déplacement de la souris avec le bouton appuyé. Enfin lorsque l'on relâche le bouton on va observer si le curseur pointe sur un noeud, si cest le cas alors on crée un nouvel arc entre le noeud sur lequel on avait cliqué au départ et le noeud sur lequel on a relâché le bouton (si jamais l'arc existe déjà ou si l'arc relie un noeud à lui même aucun arc n'est créé), le faux arc de construction est alors détruit afin qu'il ne soit plus affiché.

Enfin,la dernière interaction possible avec le graphe à travers le canvas se fait grâce au mode Select, ce mode permet de modifier les caractéristiques de l'élément sur lequel on a cliqué, si c'est sur un arc on peut modifier le partage de l'arc ou bien le supprimer, si c'est sur un noeud il est possible de changer le nom de ce dernier ou bien de supprimer le noeud. Donc comme pour les autres modes, le mode Select a deux effets. Pour modifier un noeud, est assez similaire avec le fonctionnement des autres modes, mais comme il est possible de sélectionner deux éléments différents (un noeud ou un arc) et qu'il faut faire un clique rapide (déplacer la souris en maintenant le bouton appuyé n'a pas d'intérêt en mode Select) l'organisation de la sélection est inversée. Dans les autres modes, lorsque l'on appuie sur le bouton, on fait directement la sélection du noeud et c'est en bougeant la souris et ou en relâchant le bouton que l'effet se terminait. Dans le mode Select, la première et la deuxième fonction d'interaction avec la souris (donc celles déclenchées lorsqu'on appuie sur le bouton ou lorsqu'on bouge la souris avec le bouton appuyé) n'ont pratiquement aucun effet autre que d'enregistrer la position du curseur, tout se passe au moment où l'on relâche le bouton. En relâchant le bouton on déclenche donc la 3ème fonction d'interaction de la souris et l'on récupère la position du curseur, comme pour les autres modes, on regarde d'abord si on a relâché le bouton sur un noeud, si c'est le cas, alors on ouvre une boîte de dialogue qui permet de modifier le nom du noeud ou de supprimer le noeud. Les noms ne doivent pas contenir d'espace ou de virgule (la virgule sert à la délimitation des éléments dans l'enregistrement des graphes en fichier texte) mais ils peuvent être aussi long que souhaité. Cependant si jamais on n'a pas cliqué sur noeud il se peut qu'on ai donc cliqué sur un arc. Et contrairement aux noeuds qui sont des cercles, les arcs sont beaucoup plus compliqués à sélectionner étant donné qu'ils peuvent certe apparaître épais à l'écran, mais en réalité ils sont modélisés par des segments infiniment fin et sur lesquels il est très compliqué de cliquer. Afin de palier à ce problème, nous avons créé un curseur exprès pour cette situation. En pyQT il existe des classes qui permettent de créer des lignes, et lune de ces classe propose un grand nombre de méthodes permettant de faire des calculs sur des droites, des segments etc, y compris une méthode qui détermine si 2 segments se croisent ou non. Les arcs de notre graphe étant des segments nous avons créé un curseur qui utilise deux petits segments formant un + avec l'intersection se trouvant au niveau du curseur de la souris, de cette manière

toutes les pentes des arcs sont prises en compte, car on vérifie si au moins un des deux petits segments coupe un arc, si c'est le cas alors on lance la boîte de dialogue permettant de modifier l'arc et son partage. Le curseur amélioré n'est évidemment pas visible c'est juste pour faciliter la sélection des arcs.

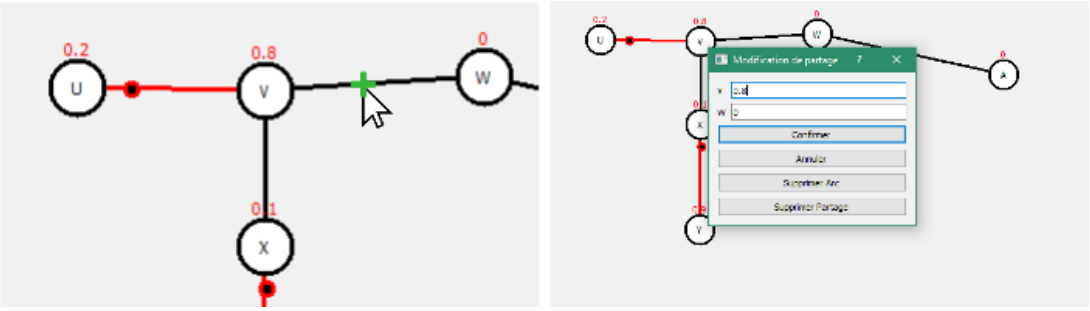


FIGURE 3.10 – Le curseur en forme de plus en représenté en vert sur l'image de gauche, on remarque qu'il coupe sans aucun doute l'arc (V, W), cela génère donc un boîte de dialogue permettant d'établir un partage entre les deux ou bien de supprimer l'arc

Enfin, lorsque l'on supprime un noeud ou un arc, tous les liens, les partages et les arcs qui étaient en relation avec celui qui est supprimé sont également supprimés.

3.2.4 Tests

Ayant la capacité de vérifier si un graphe possède au moins un partage stable et si c'est le cas, en combien d'itérations celui-ci peut être trouvé nous avons effectué quelques teste fin de voir l'impact du nombre de noeuds sur le nombre d'itérations nécessaire pour trouver un partage stable dans le graphe. Pour ce faire, nous avons fait des graphes composés de deux courbes, l'une représentant le nombre de graphes ayant au moins un partage stable sur 100 graphes, en fonction du nombre de noeuds et la seconde représente le nombre moyen d'itérations. Il faut noter qu'ici on n'inclus pas le nombre d'itérations lorsqu'aucun partage stable n'est trouvé, c'est pour cela que pour que les comparaisons aient du sens il nous faut savoir combien de graphes ont un partage stable.

Avant d'analyser nos résultats, nous voulons dire que puisque les graphes sont générés aléatoirement, nous pourrions obtenir des courbes légèrement différentes si nous essayions de refaire les tests.

Nous avons effectué trois tests avec des probabilité différentes (la probabilité étant celle pour un arc d'exister dans le graphe), 0.2, 0.5 et 0.8 afin de pouvoir observer l'impacte du nombre d'arc du graphe sur le nombre d'itérations nécessaire pour trouver un partage stable. A Chaque fois nous utilisions des graphes ayant un nombre de noeuds entre 3 et 10 inclus, et pour que les résultats soient les plus fiable possible nous avons généré 100 graphes pour chaque nombre de noeuds.

Comme vous pouvez le voir sur le graphe-ci dessous, la tendance est que plus le nombre de noeuds augmente, plus le nombre de graphes ayant un partage stable diminue et plus le nombre d'itération nécessaire pour en trouver un augmente.

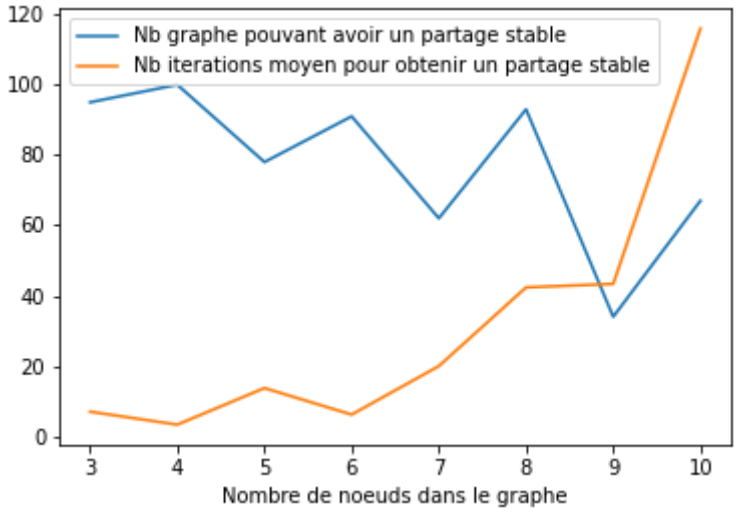


FIGURE 3.11 – Test avec une probabilité de 0.2

Contrairement au test précédent, on ne peut pas faire l'hypothèse que plus le nombre de noeuds augmente dans le graphe, plus le nombre de graphes ayant un partage stable diminue. En effet, ici on voit surtout que les graphes ayant un nombre pair de noeuds semble avoir quasiment toujours un partage stable alors que pour les graphes ayant un nombre impair de noeuds, plus ce chiffre est élevé moins il semble être possible de trouver un partage stable. Quand au nombre d'itérations nécessaire, on peut faire l'hypothèse que celui-ci augmente avec le nombre de noeuds (si celui-ci est impair) car on voit une augmentation entre les graphes à 3,5 et 7 noeuds. Cependant, vous voyez sûrement que le nombre d'itérations a chuté lorsque nos graphes ont 9 noeuds, et c'est ici que la présence de la deuxième courbe nous donne de précieuses informations. En effet, on voit que le nombre de graphes ayant un partage stable est nul, ce qui explique le nombre d'itérations puisque l'on ne compte pas d'itération lorsque l'on ne trouve pas de partage stable.

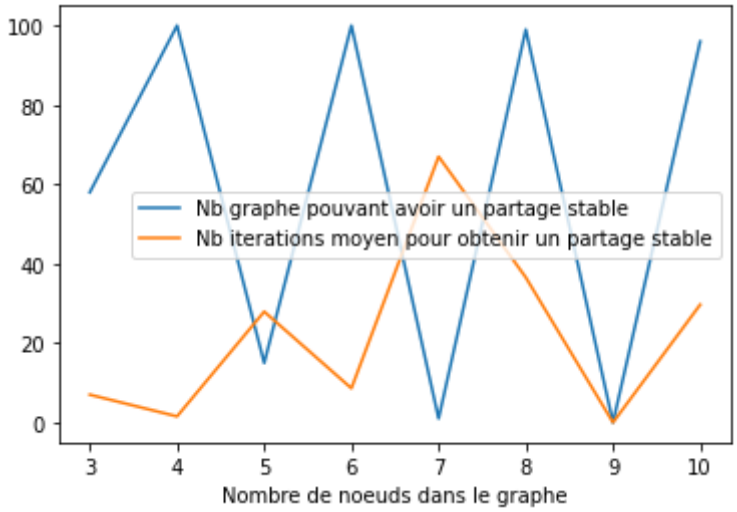


FIGURE 3.12 – Test avec une probabilité de 0.5

Enfin, regardons les tests que nous avons effectués avec une probabilité de 0.8. Les résultats sont très similaires à ceux obtenus juste avant lorsque nous avions une probabilité de 0.5,

cependant nous pouvons voir que maintenant pour 100 graphes à 7 noeuds aucun partage stable n'a pu être trouvé alors qu'il y en avait lorsque la probabilité était à 0.5. On peut également noter que le nombre d'itérations nécessaire pour trouver un partage stable pour des graphes à 5 noeuds à fortement augmenté alors que celui-ci a globalement diminuer pour tous les graphes ayant un nombre pair de noeuds.

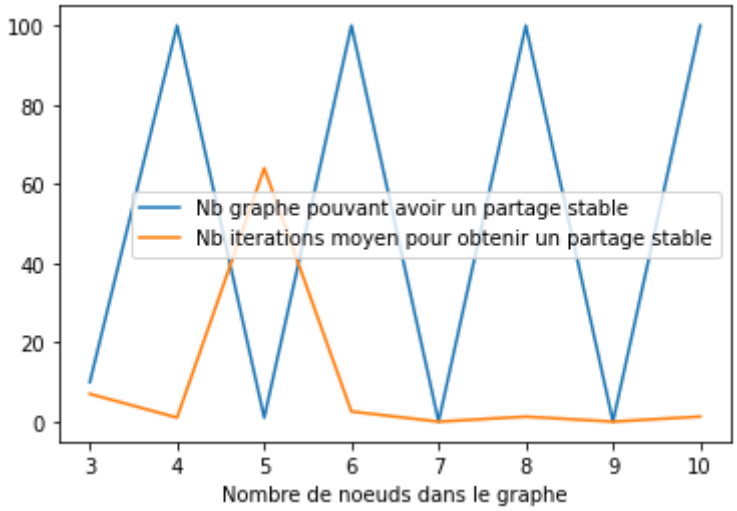


FIGURE 3.13 – Test avec une probabilité de 0.8

On peut donc conclure de ces trois tests que globalement, lorsque le graphe ne possède pas trop d'arcs, plus le nombre de noeuds augmente, plus le nombre d'itérations pour trouver un partage stable augmente. Cependant, lorsque le nombre d'arcs est plus élevé, plus il est rapide de trouver un partage stable pour des graphes à nombre de noeuds pairs mais plus long (voir impossible) d'en trouver pour des graphes à nombre de noeuds impairs.

Chapitre 4

Conclusion

(Ce que nous avons réussi à faire et ce qui n'a pas été réussi, comment nous pourrions améliorer le projet, ce que nous avons appris)

En conclusion, nous sommes assez satisfait de notre projet et du programme réalisé, nous sommes parvenu à implémenter presque toutes les fonctionnalités principales du cahier des charges et plusieurs autres secondaires. Ce qui nous permet de proposer un programme permettant à l'utilisateur de créer le graphe de partage qu'il veut, de le manipuler à sa guise, de pouvoir définir les partages entre les agents comme bon lui semble et de vérifier si les partages sont stables et équilibrés, et s'ils ne le sont pas de lui permettre de calculer automatiquement ces partages. Avec, enfin, les fonctionnalités classique des programmes d'éditions qui sont l'import et l'export (de graphes dans notre cas).

Cependant, il reste un certain nombre d'aspects qui peuvent être améliorés, en commençant par les fonctionnalités du cahier des charges que nous n'avons pas intégré, faute de temps principalement. La plus importante de ces fonctionnalité est la pondération des arcs, en effet il existe une version améliorée du système des graphes de partage avec des arcs pondérés, ce qui est plus réaliste par rapport aux liens sociaux puisque les gens sont plus ou moins "compatibles" et donc une entente entre deux individus aura une valeur différente selon les individus, ce qui est représenté par la pondération des arcs. Plusieurs de nos fonctions sont codées de manière à prendre en compte cette pondération, mais sa mise en place totale n'a malheureusement pas pu se faire à temps. Un deuxième élément qui figurait sur le cahier des charge principal était le fait de pouvoir zoomer ou dézoomer, ce qui peut être intéressant lorsqu'on manipule de grands graphes, mais c'est aussi une fonctionnalité que nous n'avons pas eu le temps d'implémenter, les autres nous semblant plus prioritaires. Enfin la dernière fonctionnalité qui figure dans le cahier des charges secondaire et que nous n'avons pas programmé est la détermination de balanced outcome sans conditions, le calcul via le edge balancing dynamics impose beaucoup d'hypothèses à vérifier par l'utilisateur pour s'assurer que le résultat soit bon, avoir un programme qui n'impose pas toutes ces hypothèses pour déterminer le balanced outcome aurait été bien mieux, même si bien plus compliqué et long à implémenter. Enfin, il existe plusieurs autres améliorations possibles qui permettraient d'améliorer le programme et qui ne figurent pas dans le cahier des charges. Une première qui concerne plutôt l'ergonomie serait de différencier le clique droit du clique gauche de la souris, en affectant au clique droit un mode unique, comme le mode Select par exemple, permettrait de retirer ce mode de la barre d'outil et de ne pas avoir à cliquer sur son bouton à chaque fois pour l'utiliser, rendant

l'édition des graphes plus confortable. Une seconde, qui est concerne aussi l'ergonomie, serait de rendre les curseurs sur les arcs interactifs, ces curseurs permettent de montrer la proportion que chaque agent perçoit du partage qu'ils réalisent, permettre de bouger le curseur le long de l'arc en cliquant dessus et en le déplaçant, comme un vrai curseur, serait très pratique pour modifier la valeur d'un partage très rapidement, l'utilisateur ne serait pas obligé d'ouvrir la boîte de dialogue pour modifier les valeurs, il aurait juste à cliquer sur le curseur et le déplacer, ce serait bien moins précis que l'entrée des valeurs via la boîte de dialogue, mais dans certaines situations il n'est pas nécessaire d'avoir des partages ultra précis. Enfin une amélioration qui serait beaucoup plus pratique pour le développement de l'application serait d'avoir des classes pour les noeuds et les arcs du graphe plutôt que les tuples que l'on utilise, l'accès aux informations de chaque noeud et chaque arc serait bien plus simple, malheureusement toute l'application est programmée sur la structure que l'on a utilisée, il faudrait presque tout reprendre depuis le début pour l'implémenter, même si pour l'utilisateur il n'y aurait pas de grandes différences.

Malgré toutes ces améliorations possibles dont certaines qui sont des fonctionnalités manquantes, nous sommes, comme dit en première partie de conclusion, assez satisfait du programme que nous avons réalisé. C'était, pour chacun de nous trois, la première fois que nous devions réaliser un aussi gros projet informatique, qui plus est, un logiciel en partant de 0. Et cela aura été, pour nous, une expérience très intéressante et enrichissante à laquelle nous avons pris plaisir de participer tout le long du semestre.

Chapitre 5

Bibliographie

[1] Jon Kleinberg and Éva Tardos. 2008. Balanced outcomes in social exchange networks. In Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC '08), Association for Computing Machinery, New York, NY, USA, 295–304. DOI :https ://-doi.org/10.1145/1374376.1376994

[2] Easley David and Kleinberg Jon. 2010. Networks, Crowds, and Markets : Reasoning About a Highly Connected World. Cambridge University Press, USA.

[3] Yossi Azar, Benjamin Birnbaum, L. Elisa Celis, Nikhil R. Devanur, and Yuval Peres. 2009. Convergence of Local Dynamics to Balanced Outcomes in Exchange Networks. In Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '09). IEEE Computer Society, USA, 293–302. DOI :https ://-doi.org/10.1109/FOCS.2009.33

[4] Mohsen Bayati, Christian Borgs, Jennifer Chayes, Yash Kanoria, and Andrea Montanari. 2015. Bargaining dynamics in exchange networks. Journal of Economic Theory 156, (2015), 417–454. DOI :https ://doi.org/10.1016/j.jet.2014.02.007

[5] Norman Braun and Thomas Gautschi. 2006. A Nash bargaining model for simple exchange networks. Social Networks 28, 1 (January 2006), 1–23.

DOI :https ://doi.org/10.1016/j.socnet.2004.11.011

[6] Tanmoy Chakraborty, Stephen Judd, Michael Kearns, and Jinsong Tan. 2010. A behavioral study of bargaining in social networks. In Proceedings of the 11th ACM conference on Electronic commerce (EC '10), Association for Computing Machinery, New York, NY, USA, 243–252. DOI :https ://doi.org/10.1145/1807342.1807382

[7] Konstantinos Georgiou, George Karakostas, Jochen Könemann, and Zuzanna Stamirowska. 2014. Social exchange networks with distant bargaining. Theoretical Computer Science 554, (October 2014), 263–274. DOI :https ://doi.org/10.1016/j.tcs.2013.11.033

[8] Moez Draief and Milan Vojnović. 2010. Bargaining dynamics in exchange networks. In 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 1303–1310. DOI :https ://doi.org/10.1109/ALLERTON.2010.5707064

[9] Yashodhan Kanoria, Mohsen Bayati, Christian Borgs, Jennifer Chayes, and Andrea Montanari. 2011. Fast Convergence of Natural Bargaining Dynamics in Exchange Net-

works. In Proceedings of the 2011 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). Society for Industrial and Applied Mathematics, 1518–1537. DOI :https ://-doi.org/10.1137/1.9781611973082.118

[10] Tanmoy Chakraborty and Michael Kearns. 2008. Bargaining Solutions in a Social Network. In Internet and Network Economics, Springer, Berlin, Heidelberg, 548–555. DOI :https ://doi.org/10.1007/978-3-540-92185-$1_6$1

[11] Antoni Calvó-Armengol. 2001. Bargaining power in communication networks. Mathematical Social Sciences 41, 1 (January 2001), 69–87. DOI :https ://doi.org/10.1016/S0165-4896(00)00049-4

Annexe A

Cahier des charges

Le logiciel concerne les partages de gâteaux entre plusieurs noeuds d'un graphe (non orienté). Ces graphes permettent de modéliser les forces de négociations entre plusieurs agents au sein d'un réseau. Dans un graphe, un partage ne peut avoir lieu qu'entre 2 agents et chaque agent ne peut partager qu'avec un seul autre agent à la fois, le partage est représenté par la proportion du gâteau que chacun des deux agents obtient (donc pour un partage entre 2 agents, deux nombres dont la somme est égale à 1). Le but est d'abord de déterminer si un graphe avec un partage donné est stable, c'est à dire que parmi les couples d'agents, aucun n'a d'intérêt à rompre son accord actuel pour réaliser un partage avec un autre agent. Le logiciel s'ouvre dans une fenêtre, il permet de visualiser au moins un graphe et possède des boutons afin d'agir sur ce dernier et d'appeler les différentes fonctionnalités. Les différentes fonctionnalités qui sont appelables à partir du logiciel sont :

- Créer un graphe (noeuds et arcs)
- Enregistrer un graphe (et son partage actuel) sous forme de fichier texte
- Ouvrir un graphe à partir d'un fichier texte
- Permettre à l'utilisateur de proposer un partage entre les différents agents du graphe ouvert (placer sur les arcs une pondération)
- Vérifier si le partage d'un graphe est stable (vérification automatique ou bien vérification suite à l'appel de la fonctionnalité par l'utilisateur)
- Permettre à l'utilisateur de zoomer sur le graphe
- Calcul du balanced outcome à l'aide de la méthode Edge balacing Dynamics
- Avoir des graphes de bases pour permettre de tester et prendre en main directement l'application à son lancement

Secondaire :

- Permettre à l'utilisateur de réorganiser les noeuds dans l'espace afin de rendre le graphe plus facile à lire
- Générer un partage stable à partir du partage actuel s'il n'est pas stable
- Permettre de voir l'évolution étape par étape du partage non stable vers le partage stable lorsqu'on demande une génération de partage stable
- Vérifier si un partage est un balanced outcome
- Générer le balanced outcome d'un graphe

— Permettre de voir l'évolution du partage jusqu'à obtenir le balanced outcome d'un graphe lorsque qu'il est généré

Annexe B

Manuel utilisateur

Bonjour, merci d'utiliser notre logiciel sur les graphes de partage, avant de vous lancer, il faut vérifier 2 prérequis important, en effet ce programme tourne sous python à l'aide de la librairie pyQT, pour vous assurer de pouvoir utiliser le programme normalement, veuillez vous assurer d'avoir installé sur votre machine :

— python version 3

— pyQT version 5

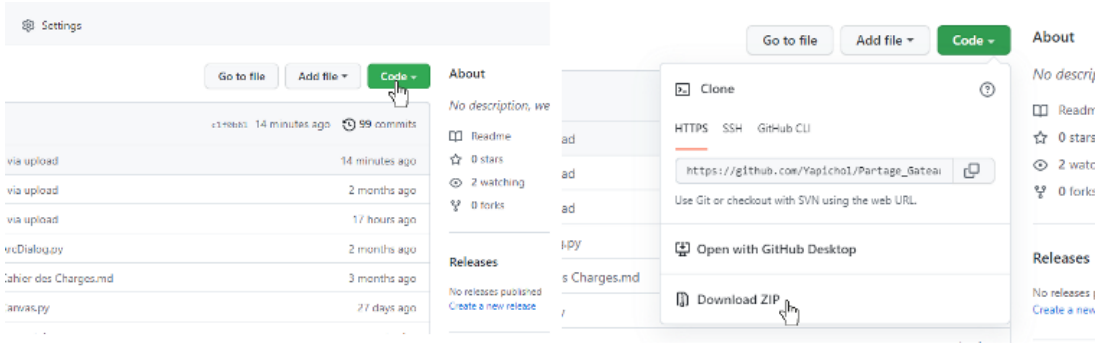
Si c'est fait alors nous pouvons commencer, ce manuel vas vous présenter toutes les fonctionnalités présentes dans ce logiciel et vous montrer comment les utiliser !

B.1 Télécharger et Démarrer le Logiciel

B.1.1 Téléchargement

Si vous ne possédez pas le code source de l'application vous pouvez le retrouver via ce lien qui vous amènera au github du projet.

Une fois sur le site cliquez sur le bouton "Code" puis sur "Download ZIP" comme sur les figures ci-dessous :



Extrayez le dossier de l'archive vous devriez obtenir un dossier nommé "Partage_Gateaux-main", ça y est vous possédez le code source et pouvez y accéder comme bon semble.

B.1.2 Démarrage

Maintenant que vous possédez le code source du logiciel, vous allez pouvoir l'exécuter, mais tout d'abord ouvrez le dossier "Partage_Gateaux-main", vous devriez tomber sur un autre dossier du même nom, ouvrez le également, maintenant vous devriez avoir plusieurs fichiers python (.py) et des dossier, vous allez pouvoir lancer le logiciel, pour cela, 2 possibilités :

— Double-cliquez sur le fichier "Main.py" comme s'il s'agissait d'un exécutable

— Ouvrez un terminal / invite de commande et rendez vous dans le deuxième dossier "Partage_Gateaux-main" (celui avec les fichiers python) et exécutez le fichier "Main.py" (selon vos installations sur votre machine la commande peut légèrement varier, vous pouvez essayer ces deux là : "python Main.py" ou "python3 Main.py")

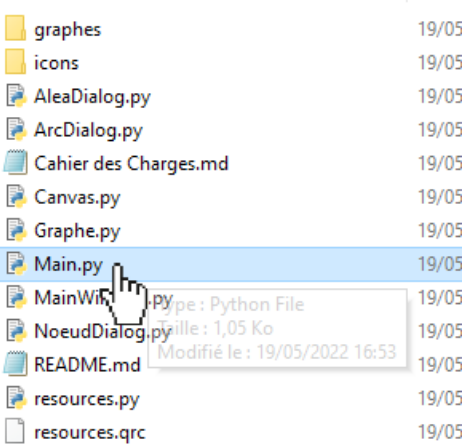


FIGURE B.1 – Lancement en double-cliquant, comme un exécutable

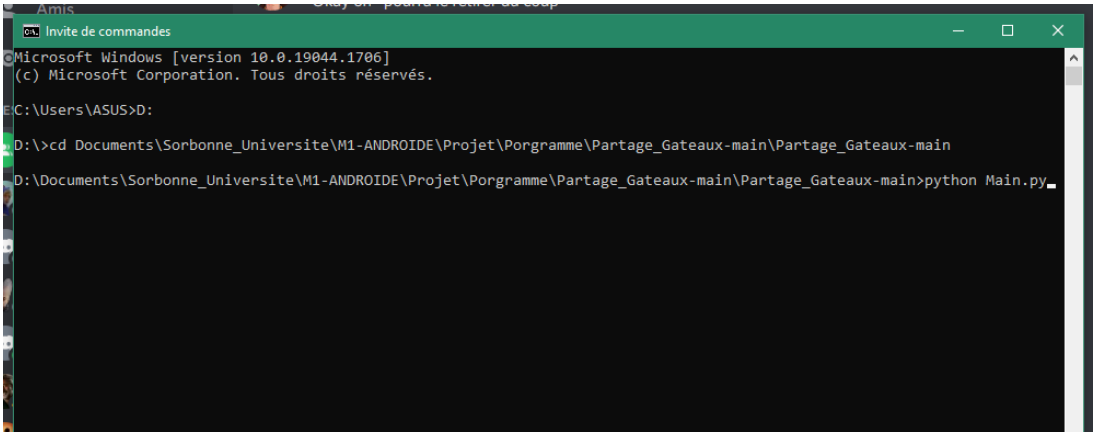


FIGURE B.2 – Lancement via l'invite de commande

Une fois le programme démarré vous devriez avoir, selon la façon dont vous l'avez lancé :

— Exécutable : 2 fenêtres qui se sont ouvertes une ressemblant à un terminal et une seconde nommée "python" avec une barre d'outils, 2 menus et un graphe affiché

à peu près au centre de la fenêtre (dans le terminal il sera peut-être affiché "Erreur au niveau des noeuds ou arcs", ce n'est pas un problème, c'est une remarque indiquant qu'un arc a voulu être placé de manière récursive ou bien 2 fois par le générateur aléatoire, ce qui n'est pas autorisé par le programme, ça ne signifie pas que l'application ne fonctionne pas au contraire)

— Terminal : une fenêtre nommée "python" qui s'est ouverte et composée d'une barre d'outils, 2 menus et un graphe affiché à peu près au centre de la fenêtre (dans le terminal il sera peut-être affiché "Erreur au niveau des noeuds ou arcs", ce n'est pas un problème, c'est une remarque indiquant qu'un arc a voulu être placé de manière récursive ou bien 2 fois par le générateur aléatoire, ce qui n'est pas autorisé par le programme, ça ne signifie pas que l'application ne fonctionne pas au contraire)

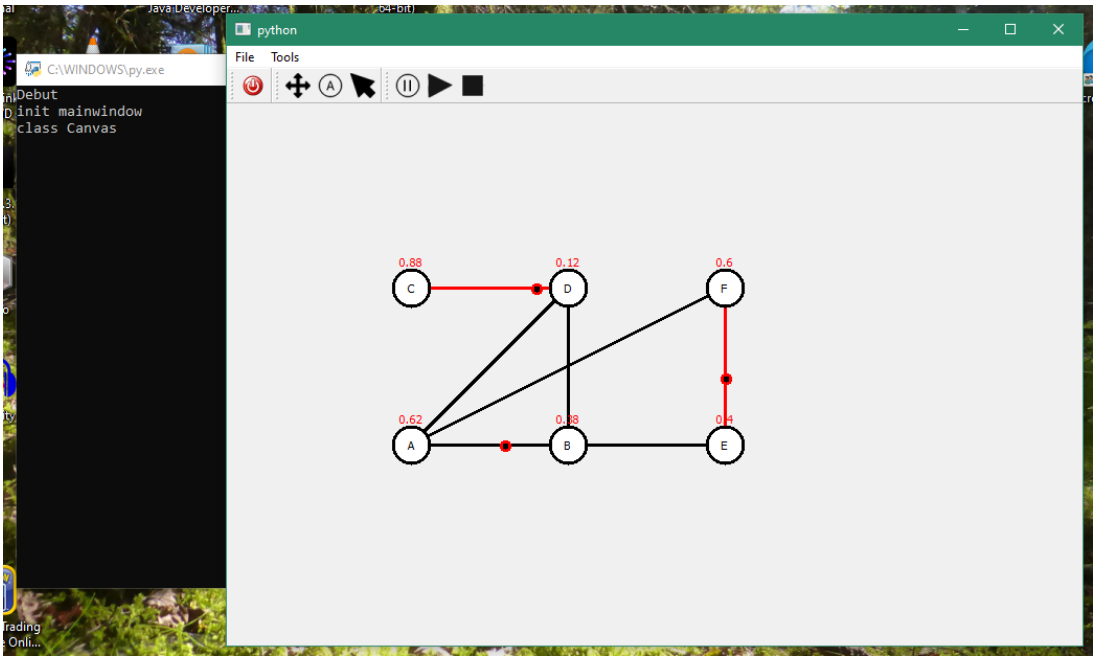


FIGURE B.3 – Fenêtres lors du démarrage en mode exécutable (le graphe est généré aléatoirement, il y a donc de grandes chances que vous n'ayez pas le même que sur cette figure lors de votre démarrage)

B.2 Le Tour des Menus

Votre application est ouverte avec un premier graphe qui a été généré aléatoirement, mais pour le moment il ne nous intéresse pas, nous allons d'abord faire un tour des menus.

B.2.1 File

Le premier menu, que l'on le retrouve dans toutes les applications, le menu File, dedans y figure des commandes classiques que nous allons présenter rapidement, ouvrez le également :

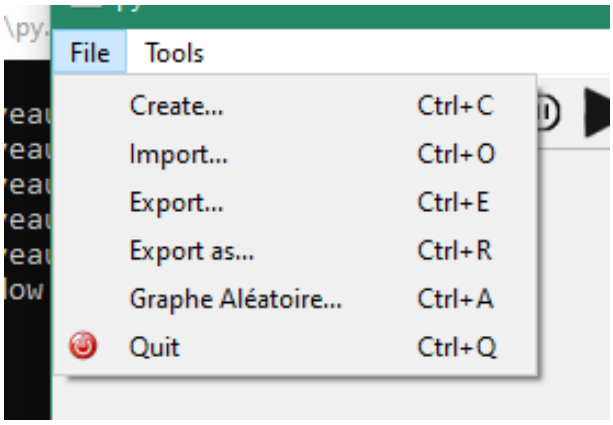


FIGURE B.4 – Apparence du menu File

Create

La première commande de ce menu est "Create", c'est l'équivalent de la commande "New" dans d'autres programmes, lorsque que vous appelez cette commande, le graphe qui était dans votre fenêtre disparaît totalement et vous vous retrouvez avec une fenêtre totalement vierge, nous n'avons pas besoin du graphe aléatoire pour le moment, cliquez pour obtenir un espace de travail vierge :

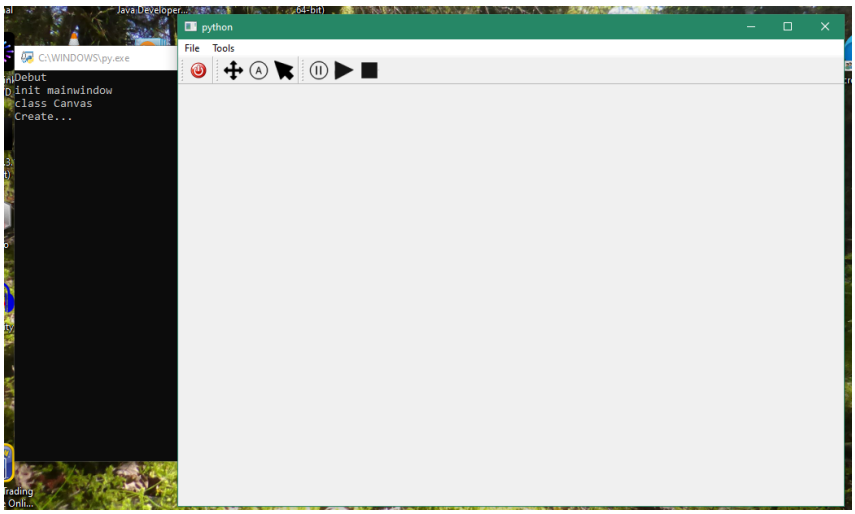


FIGURE B.5 – Lorsque vous cliquez sur Create, le nom de la commande s'affiche dans votre terminal, vous pouvez également accéder à la commande avec le raccourcis "Ctrl+C"

Import

Autre commande extrêmement classique, la commande Import, elle vous permet d'aller chercher dans vos dossier un fichier .txt d'un graphe que vous avez sauvegardé précédemment, normalement vous n'en avez aucun, mais notre programme en comporte un certain nombre qui sont préenregistrés, pour aller en chercher un lancez la commande :

— Vous arrivez normalement dans le dossier du programme, sinon rendez vous dans le dossier du programmes

— Ouvrez ensuite sur le dossier "Graphe"

— Vous tombez normalement sur une liste de fichiers .txt

— Ouvrez le fichier grapheT.txt

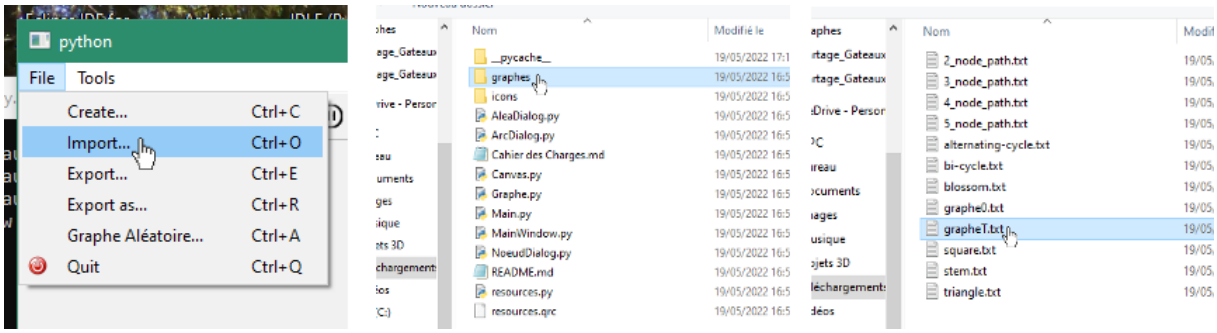


FIGURE B.6 – Illustration des étapes données juste au dessus

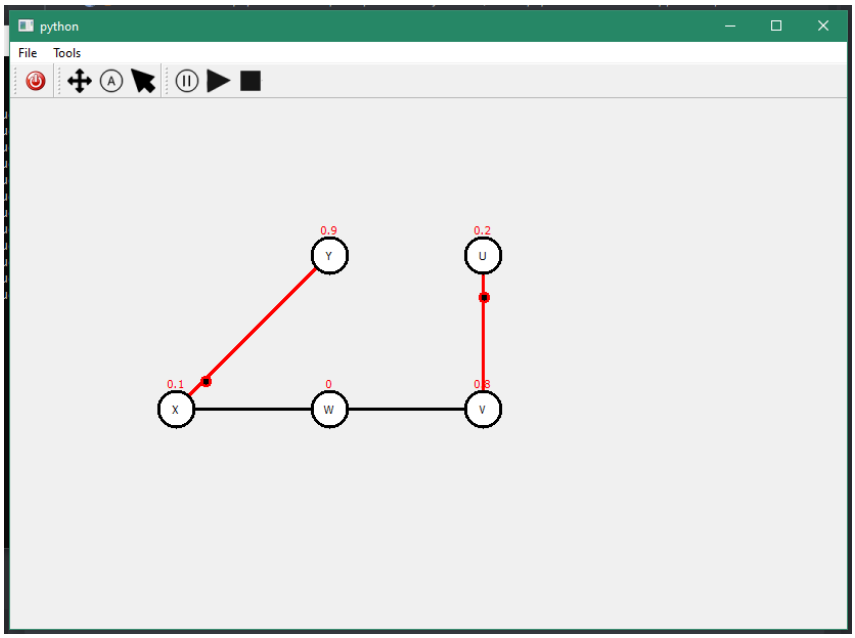


FIGURE B.7 – Vous devez normalement voir s'afficher ce graphe là, comme pour les autres commandes, Import devrait s'afficher aussi dans votre terminal

Export / Export as

Comme son nom l'indique cette commande sert à exporter, et donc sauvegarder votre graphe dans un fichier texte du même type que celui que vous venez d'importer, "Export" réalise une exportation automatique de votre graphe et l'enregistre dans le dossier "graphes" en lui donnant un nom généré, "Export as" sert quant à lui à exporter votre graphe dans le dossier que vous souhaitez et en lui donnant le nom que vous voulez, cliquez sur Export as :

— Allez dans le premier dossier "Partage_Gateaux-main" (celui avec l'autre dossier de même nom)

— Tapez le nom que vous voulez

— Confirmez

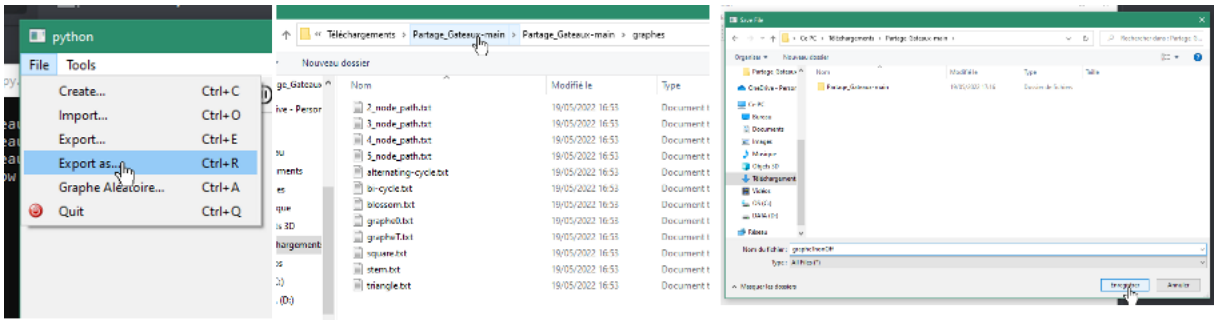


FIGURE B.8 – Illustration des étapes données juste au dessus

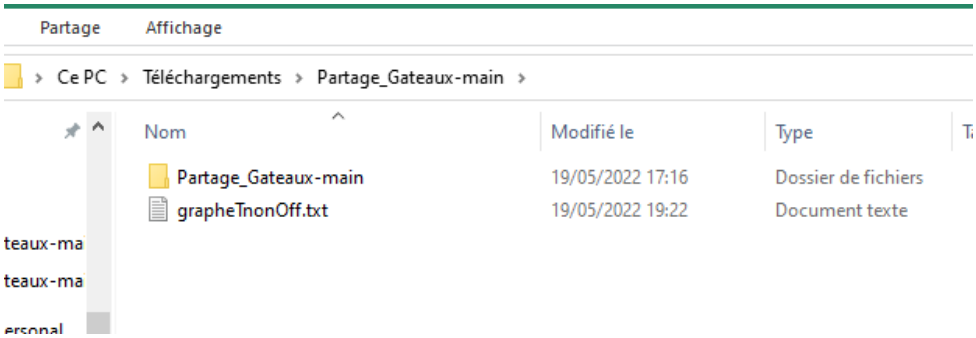


FIGURE B.9 – On remarque que l'exportation s'est bien déroulée et qu'il a bien pris le nom qu'on lui avait donné, si vous le souhaitez vous pouvez supprimer le fichier que vous venez de sauvegarder, il ne sert plus désormais

Graphe Aléatoire

Cette commande permet en quelque sorte de relancer le programme depuis le début, cela va effacer le graphe que vous avez actuellement (comme un Create) et créer un graphe aléatoire selon les informations que vous aurez indiqué dans la boîte de dialogue, vous pourrez mentionner le nombre de noeuds que vous désirez ainsi que la proportions de partages parmi les arcs, lancez la comande :

— Entrez 9 pour le nombre de noeuds
— Mettez la valeur que vous voulez pour la proportion (entre 0 et 1)
— Et enfin, Confirmez

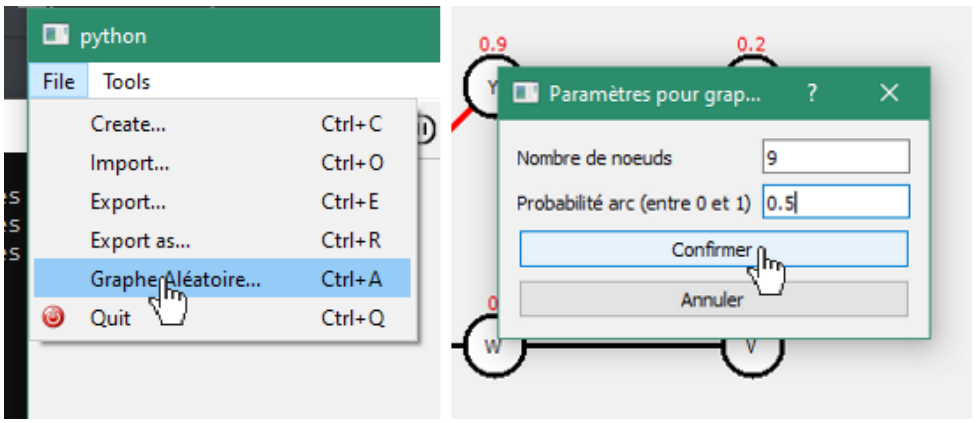


FIGURE B.10 – Illustration des étapes

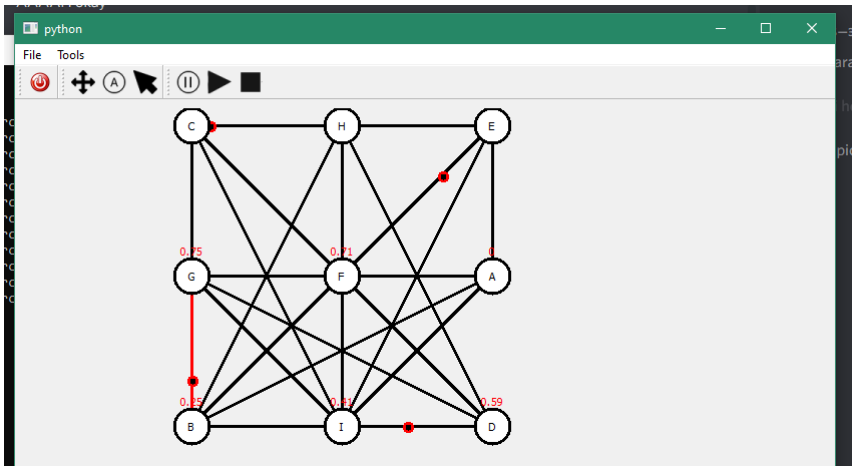


FIGURE B.11 – Vous devriez obtenir quelque chose de cette forme là, les graphes sont affichés pour la première fois de manière à ce qu'aucun noeud ne se superpose avec un autre

Quit

Cette commande sert simplement à fermer l'application, elle a la même fonctionnalité que la croix de la fenêtre, en quittant vous perdez les modifications que vous avez apporté à votre graphe, mais le programme demande une confirmation afin d'éviter de quitter sans faire exprès ou en ayant oublié quelque chose. Ne quittez pas tout de suite, nous allons passer au second menu...

B.2.2 Tools

Le menu des Tools (ou Outils) est aussi très répandu, c'est ici que se trouvent les fonctions pour apporter des modifications au graphe dans le cas présent nous en avons 2, une commande concernant le partage stable et une concernant le partage équilibré :

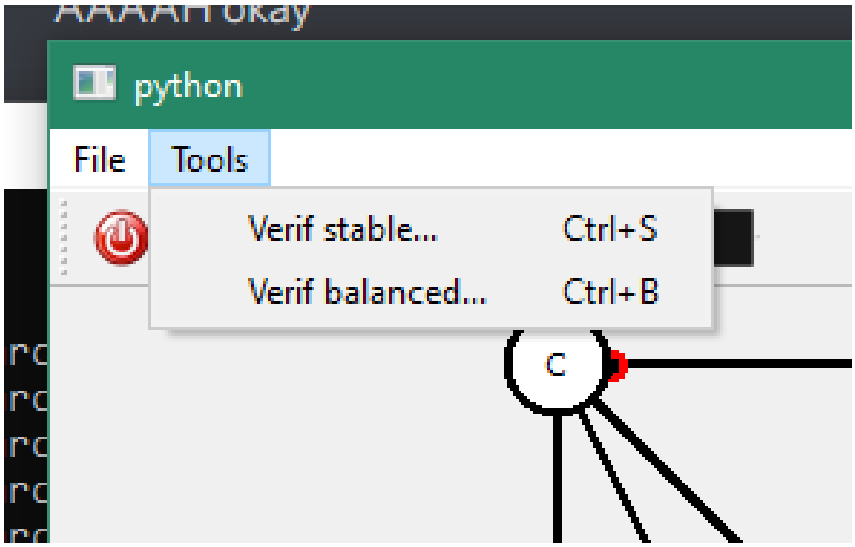


FIGURE B.12 – Le menu Tools ne possède que 2 commandes, mais elles sont particulièrement importante pour ce logiciel

Verif Stable

Cette commande vérifie si votre graphe est dans une configuration de partage stable / stable outcome, si vous avez toujours votre graphe de 9 noeuds, importez le graphe 4_node_path.txt, sur un graphe de 9 noeuds l'exécution prendrait trop de temps. Ce graphe n'est pas stable, le programme va donc vous proposer de le rendre stable, vous pourrez soit suivre l'exécution pas à pas en observant les modifications au niveau des partages, soit décider d'observer directement le résultat :

- Importez le 4_node_path.txt depuis le dossier "graphes"
- Appelez la commande "Verif stable" du logiciel
- Le programme devrais vous dire que le graphe n'est pas stable, cliquez sur "Ok"
- Le programme va ensuite vous proposer de rendre votre graphe stable, répondez "Yes"
- Enfin le programme va vous proposer d'afficher les itérations de la résolution, choisissez à nouveau "Yes" puis observez le déroulement de la résolution (très rapide sur ce petit graphe là)
- Une dernière boîte de dialogue vous renseigne sur le nombre d'itération qui fut nécessaire à rendre le graphe stable, répondez "Ok" pour clore la vérification. Si jamais il n'y a pas de partage stable possible, une boîte de dialogue vous en informera et vous indiquera les noeuds qui n'ont pas été rendu stables, et les afficheras en gris

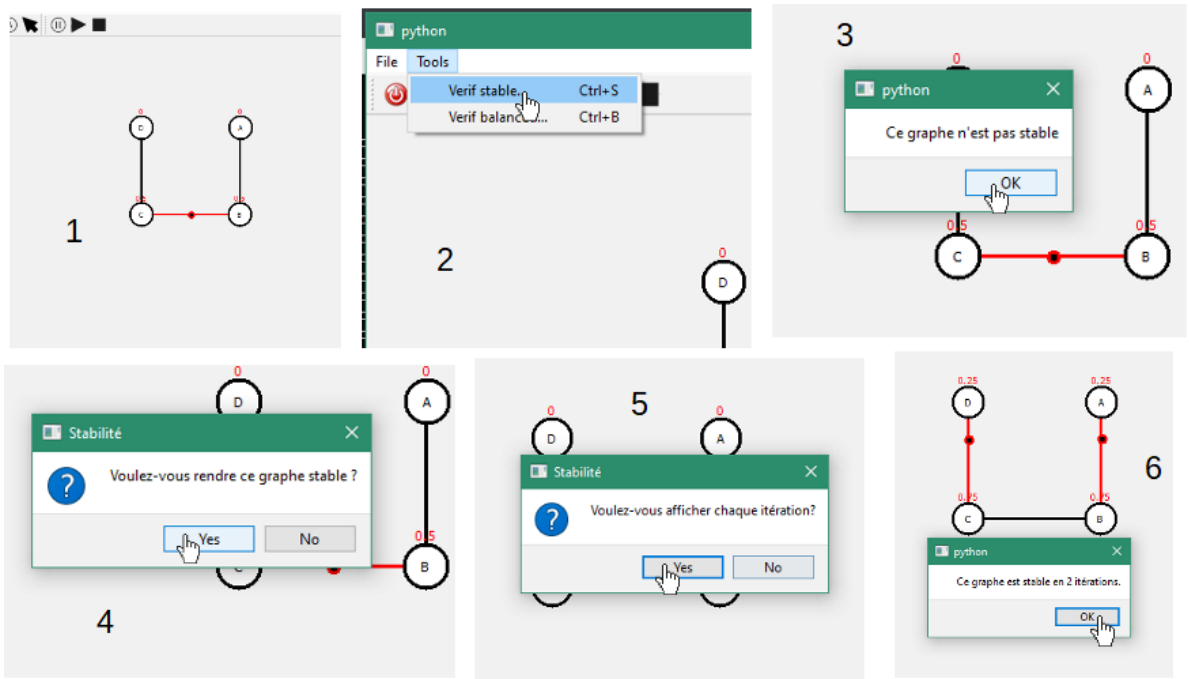


FIGURE B.13 – Illustration des étapes données pour stabiliser un graphe en observant étape par étape la progression

Durant l'exécution de la stabilisation en étape par étape, vous pouvez mettre pause à tout moment dans le processus, il vous suffit d'appuyer sur le bouton pause de la barre d'outil, pour reprendre l'exécution, il s'uffit de cliquer sur le bouton play juste à roite d bouton pause. Enfin si jamais vous souhaitez arrêter l'exécution, vous pouvez appuyer sur le bouton stop à droite du bouton play.

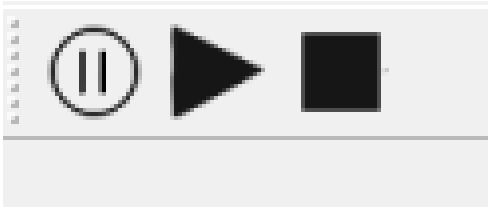


FIGURE B.14 – Tout à gauche : Pause, au centre : Play, tout à droite : Stop

Verif Balanced

La commande vérif balanced fonctionne exactement sur le même principe que celle de Verif Stable, la seule différence est que l'algorithme de Verif Balance ne permet pas de changer, d'ajouter ou de retirer des partages, il ne fait qu'équilibrer les valeur des partages déjà réalisés, il est donc important de préparer les partages en prévision, en essayant d'ajouter le plus de partages possible, les modification des partages seront expliquées dans la prochaine partie qui portera sur la modification du graphe (pour faire fonctionner la commande verif balanced, il faudra éviter de mettre des partage 1 - 0, ou 0,5 - 0,5). Mais il est possible de se rendre compte du fonctionnement de Verif Balanced grâce au graphe grapheT.txt que nous avions déjà importé précédemment :

- Importez le grapheT.txt depuis le dossier "graphes"
- Appelez la commande "Verif balanced" du logiciel
- Le programme vous demandera si vous souhaitez rendre les arcs balanced (équilibrés), répondez "Yes"
- Le programme va ensuite vous proposer d'afficher chaque itération, choisissez à nouveau "Yes", regardez les valeurs des partages se mettre à jour
- Une dernière boîte de dialogue vous renseigne sur le nombre d'itération qui fut nécessaire pour rendre les arcs balanced, répondez "Ok" pour clore la vérification. Si jamais il n'y a pas de balanced outcome possible, une boîte de dialogue vous en informera également

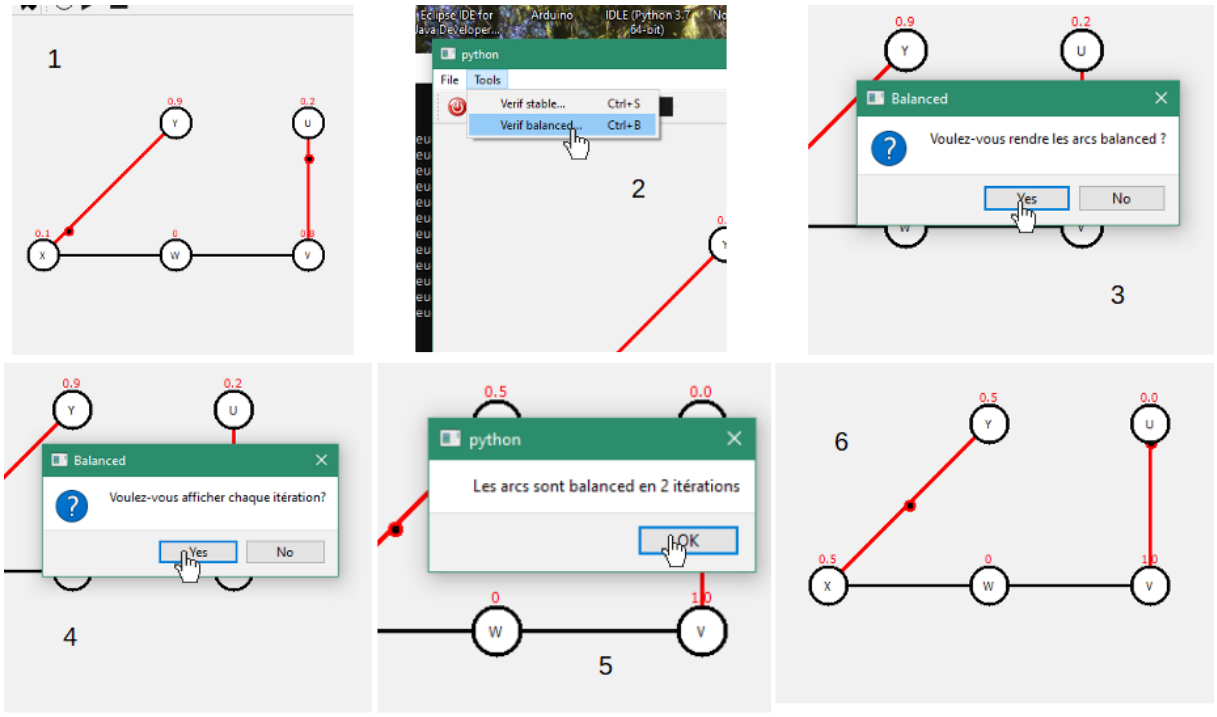


FIGURE B.15 – Illustration des étapes données pour rendre balanced les arcs d'un graphe en observant étape par étape la progression

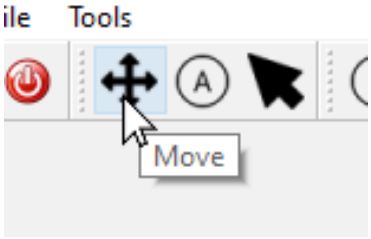
De la même manière que pour Verif Stable, il est possible de mettre pause, de reprendre l'exécution ou de l'arrêter durant un balancing en étape par étape.

B.3 Modifiez votre Graphe

Voici la dernière partie de ce guide pour utiliser ce logiciel, cette partie porte sur les modifications que vous pouvez apporter au graphe que vous avez ouvert, cela passe par 3 modes, le mode Move, le mode Draw et enfin le Mode Select.

B.3.1 Mode Move

Le mode move est le mode par éfaut qui est activé lorsque vous démarrez le logiciel, il vous permet de déplacer les noeuds des graphes et même de déplacer le graphe d'un seule coup. Le mode Move peut aussi être activé en appuyant sur le bouton en forme de crois fléchée :



Déplacer un Noeud

Déplacer un noeud peut être très utile lorsqu'il s'agit de rendre un graphe plus lisible, surtout que des fois, si des arcs ou des noeuds se superposent, on ne peut pas connaître exactement la structure du graphe. pour déplacer un noeud en mode Move rien de plus facile :

— Cliquez avec votre souris sur un noeud du graphe (le noeud W par exemple si vous avez toujours le grapheT.txt d'ouvert)

— en maintenant le bouton de la souris cliqué (enfoncé), déplacez votre souris vers là où vous souhaitez déplacer votre noeud, le noeud suit votre souris tant que vous gardez le bouton enfoncé

— Pour refixer le noeud, lâchez le bouton de la souris, le noeud restera à l'endroit où vous avez arrêté d'appuyer sur la souris

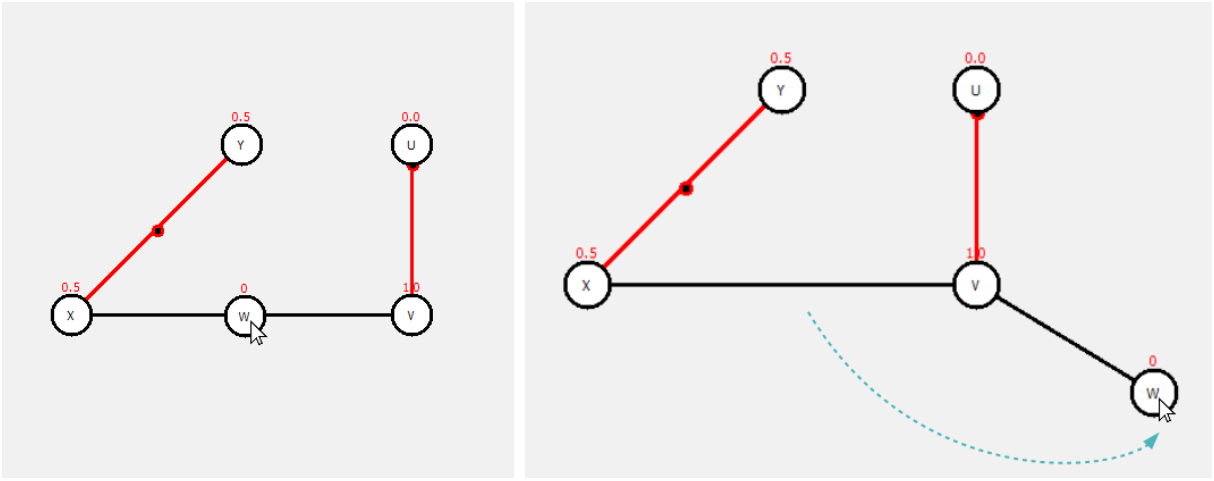


FIGURE B.16 – Déplacement du noeud W

Déplacer le Graphe

Déplacer le graphe est tout aussi simple que de déplacer un noeud, cela peut de plus être très pratique lorsque votre graphe, suite à vos manipulation, se trouve un peu excentré, pour le déplacer :

— Cliquez avec votre souris sur un endroit vide de l'espace d'affichage (n'importe où tant qu'il n'y a pas de noeud ou d'arc là où votre curseur pointe)

— en maintenant le bouton de la souris cliqué (enfoncé), déplacez votre souris vers là où vous souhaitez déplacer votre graphe, le graphe suit votre mouvement tant que vous gardez le bouton enfoncé

— Pour relâcher le graphe, lâchez le bouton de la souris, le graphe restera à l'endroit où vous avez arrêté d'appuyer sur la souris

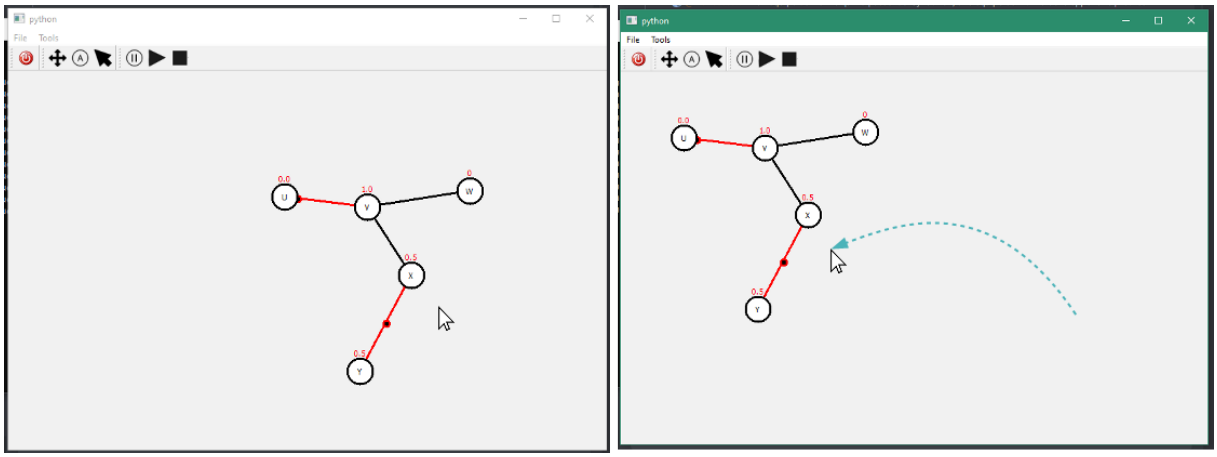
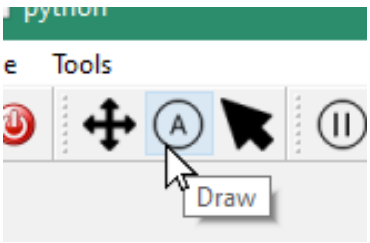


FIGURE B.17 – Déplacement du graphe vers la droite

Attention en déplaçant votre graphe, il est possible de le faire sortir du champ de la fenêtre, vous pourrez le ramener dedans même si vous le lâchez, il suffit de recliquer pour à nouveau le déplacer, mais en dehors de la fenêtre il n'est pas évident de bien se rendre compte de ce qu'on fait et donc vous pourriez le perdre et ne plus parvenir à le faire revenir dans la fenêtre. Si jamais cela vous arrive et qu'il s'agissait d'un graphe que vous ne souhaitiez pas perdre, exportez le (avec "Exporter as" par exemple), puis fermez l'application et redémarrez la, les coordonnées des noeuds et du graphe ne sont pas enregistrés dans le fichier lors de l'export et en relançant le programme, ça remet l'origine du repère au centre de la fenêtre, lorsque vous importerez votre graphe il devrait s'afficher au centre de la fenêtre)

B.3.2 Mode Draw

Le mode Draw est un mode permettant d'ajouter des éléments à votre graphe, de "dessiner" de nouvelles parties au graphe. Il vous permet donc de l'agrandir et de le complexifier. Pour passer en mode Draw il faut cliquer sur le bouton avec un A encerclé, représentant un nouveau noeud :



Ajouter un Noeud

Pour ajouter un noeud en mode Draw c'est très simple, il suffit de cliquer avec votre souris sur une partie vide de la fenêtre d'affichage (donc un endroit où il n'y a ni arc ni noeud), un nouveau noeud dénommé par une lettre de l'alphabet est créé à l'endroit où vous pointiez avec votre curseur au moment de cliquer.

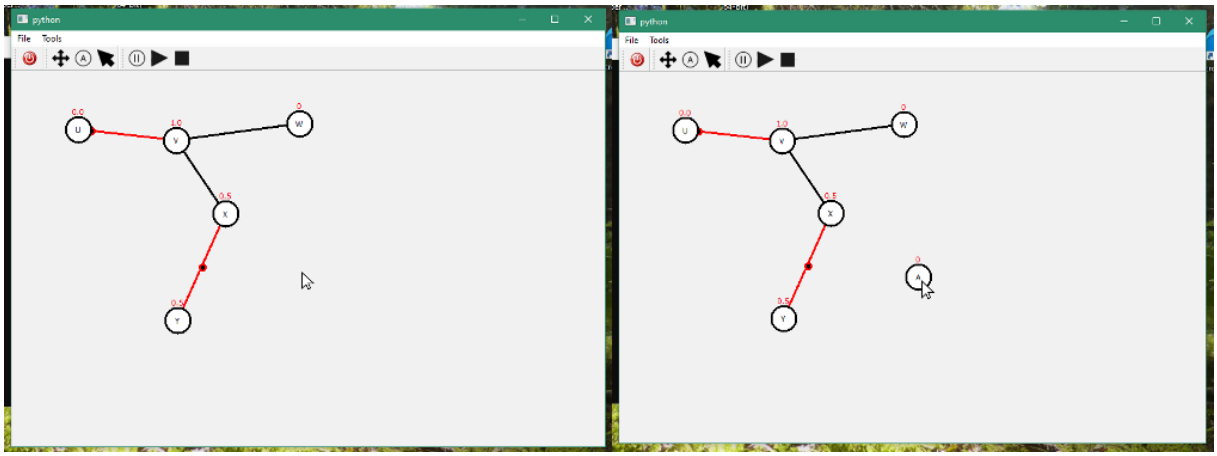


FIGURE B.18 – En mode Draw, sur la première image c'est juste avant de cliquer, et sur la deuxième c'est juste après avoir cliqué, le noeud apparaît à l'emplacement où vous cliquez

Ajouter un Arc

Pour ajouter un arc entre 2 noeuds c'est aussi très simple, mais cela demande d'avantage d'étapes que pour ajouter simplement un noeud :

- Cliquez avec votre souris sur un des deux noeuds que vous voulez joindre via ce nouvel arc
- en maintenant le bouton de la souris cliqué (enfoncé), déplacez votre souris du premier noeud vers le deuxième noeud, un trait bleu apparaît il est là pour représenter le futur arc que vous allez créer et vous permettre de mieux voir ce que vous êtes en train de faire
- Lorsque vous avez atteint l'autre noeud que vous souhaitiez lier au premier, donc lorsque les deux noeuds que vous vouliez lier par un nouvel arc sont reliés par l'arc bleu, il vous suffit de lâcher le bouton de la souris, l'arc bleu va se transformer en un vrai arc noir comme les autres.

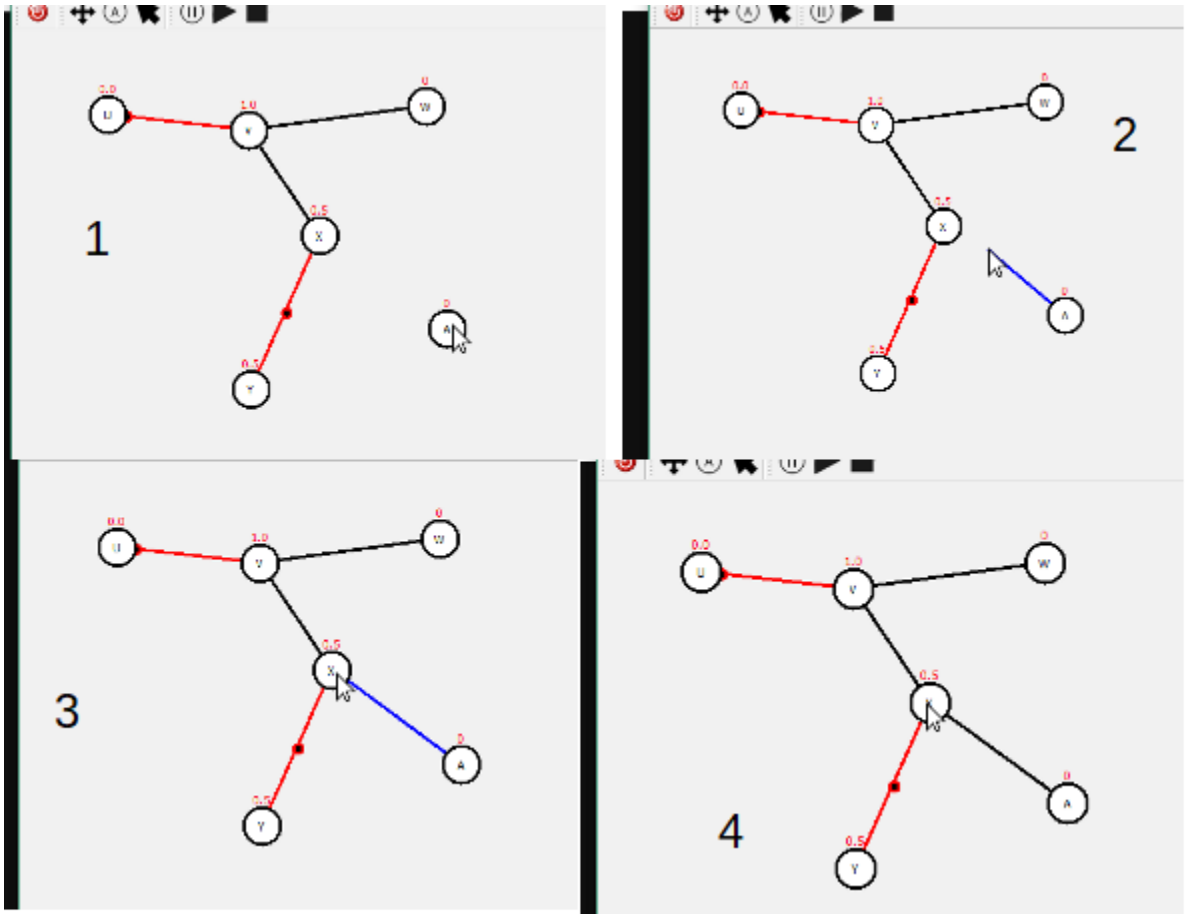
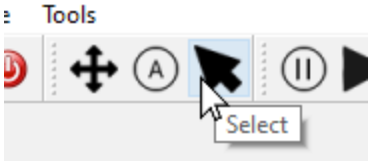


FIGURE B.19 – Illustration des indications ci-dessus, il faut cliquer sur un premier noeud, maintenir le clique, et tirer l'arc du premier noeud vers un deuxième noeud, une fois le deuxième atteint on peut lâcher le clique de la souris, l'arc est créé et devient noir

Si jamais vous lâchez le clique de la souris pendant que vous étiez en train de créer un arc (par erreur, car vous avez changé d'avis, ou pour une quelconque autre raison) l'arc bleu disparaît et la création de l'arc est annulée, vous pouvez évidemment recommencer la création de cet arc.

B.3.3 Mode Select

Enfin, la dernière façon d'interagir avec le graphe, le mode Select. Le mode Select est très intéressant car il permet de modifier les éléments qui existent déjà, il peut permettre de définir un partage entre 2 noeuds (qui étaient donc déjà liés par un arc mais pas un partage), ou d'en annuler un, il peut faire changer le nom d'un noeud également, et il permet enfin de détruire des éléments (noeuds et arcs) afin de vous permettre de modifier votre graphe comme bon vous semble. Pour passer en mode Select vous devez cliquer sur le bouton à l'icône de grosse fleche :



Renommer un Noeud

Pour renommer un noeud c'est très simple, il faut juste savoir que le nom d'un noeud ne peut pas être composé d'espaces ni de virgules (si vous entrez un nom qui en comporte, le nom de votre noeud restera inchangé), tous les autres caractères sont autorisés, les noms sont également restreint en terme d'affichage, le nom du noeud peut être aussi long que vous le souhaitez, mais son affichage est limité, les noms ne peuvent pas dépasser les limites de leur noeud, afin de garder une certain lisibilité au niveau de la structure du graphe. Maintenant, pour changer le nom d'un noeud vous devez :

— Cliquer sur le noeud que vous souhaitez renommer

— Une boîte de dialogue va alors s'ouvrir avec un champs texte avec le nom actuel de votre noeud

— Supprimer le nom actuel du champs texte et y inscrire le nouveau nom de votre noeud

— Cliquer sur confirmer et votre noeud est renommé

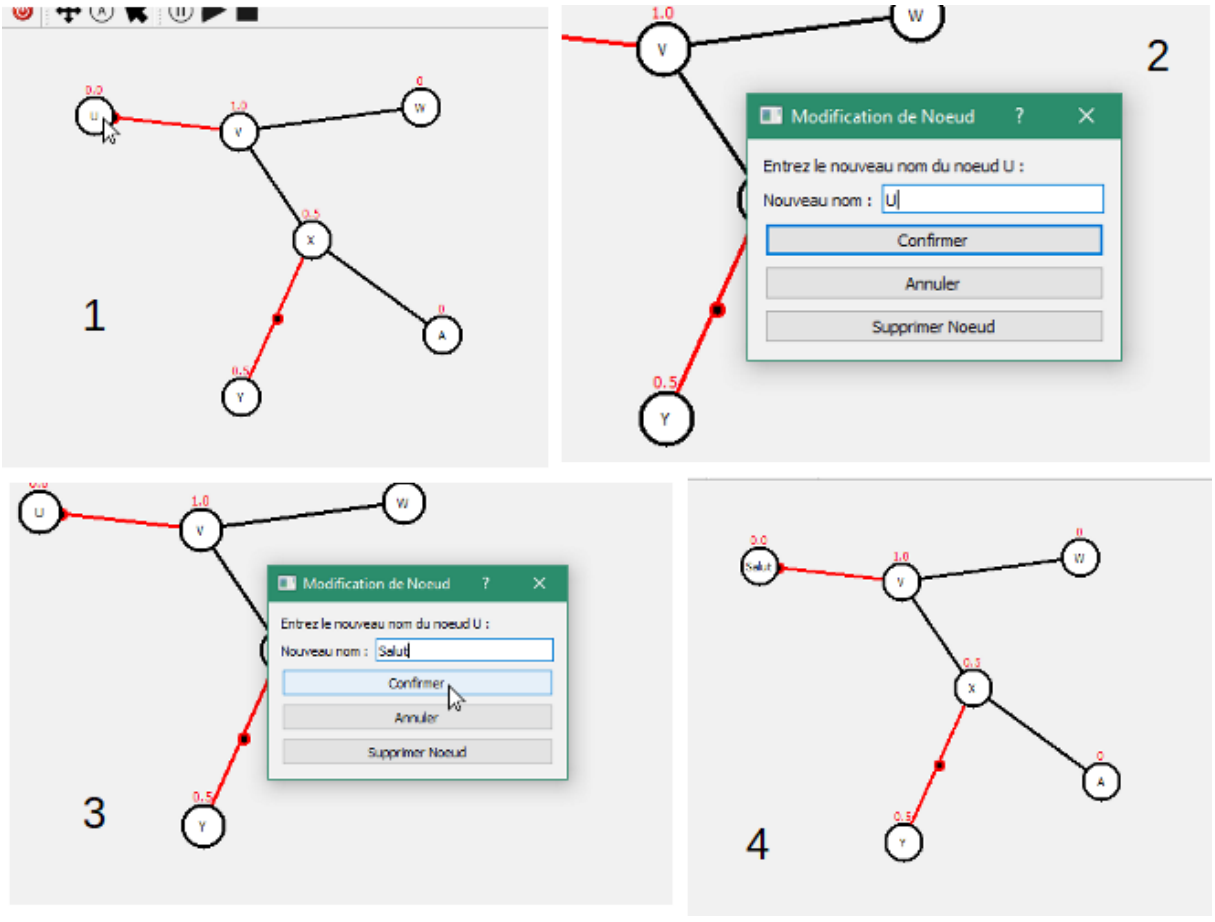


FIGURE B.20 – Illustration des indications ci-dessus

Supprimer un Noeud

La suppression d'un noeud suis la même procédure que pour le changement de nom :

— Cliquez sur le noeud que vous souhaitez supprimer

— Une boîte de dialogue va alors s'ouvrir, c'est la même boîte de dialogue que pour le renommage, il y aura les même champs texte et bouton de confirmation, et il y a également le bouton "Supprimer Noeud" que l'on n'a pas mentionné dans le changement de nom car elle n'était pas nécessaire

— Cliquez sur le bouton "Supprimer Noeud"

— Une nouvelle boîte de dialogue va alors s'ouvrir vous demandant de confirmer la suppression du noeud (au cas où vous changiez d'avis ou que vous ayez cliqué dessus sans le vouloir)

— Confirmez la suppression en cliquant sur "Yes", votre noeud est alors supprimé (si vous cliquez sur "No" vous revenez à la première boîte de dialogue avec le changement de nom et la suppression)

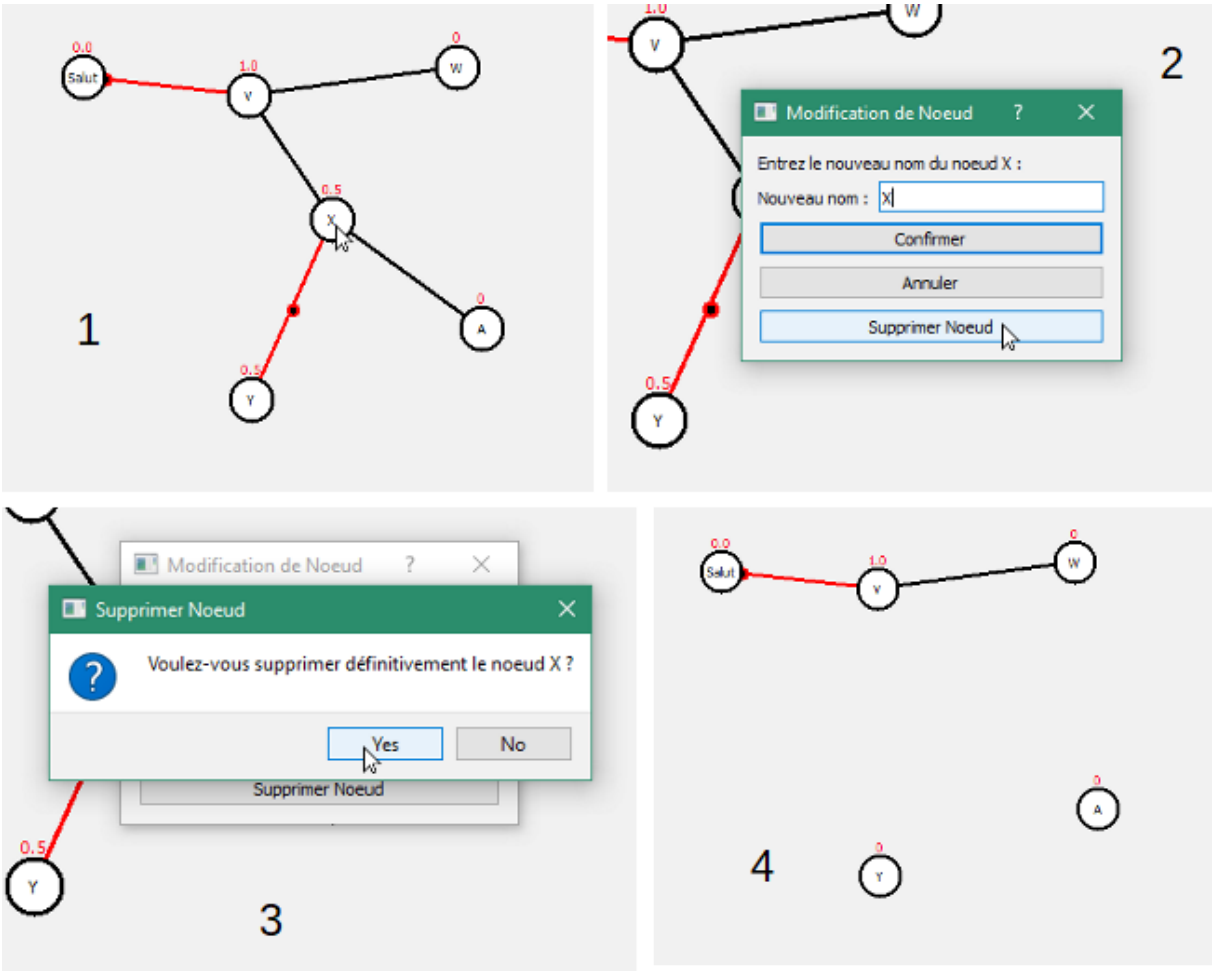


FIGURE B.21 – Illustration des indications ci-dessus en supprimant X, on remarque également que du coup, tous les arcs qui étaient reliés à X ont été également effacés ainsi que son partage annulé (Y est passé à 0 alors qu'il était à 0.5, car X n'était plus là pour partager avec lui, il perds tout son gain)

Modifier le partage d'un Arc

La modification du partage d'un arc repose exactement sur le même principe que le changement de nom d'un noeud, il y a juste une légère différence puisque dans ce cas il s'agit de valeur. Les valeurs entrées sont normalisées, le poids des arcs étant à 1 il est

impossible d'avoir un partage qui dépasse cette quantité, cependant lorsque l'on définit le partage il est tout à fait possible de rentrer des valeurs supérieures au poids de l'arc, ces valeurs seront simplement remises au niveau de poids de manière proportionnelle (par exemple si vous entrez 0,5 et 0,5, les noeuds auront ces valeurs, et si vous entrez 10000 et 10000 aussi puisqu'ils ont al même quantité, si vous entrez 2 et 8, alors les noeuds auront 0,2 et 0,8, la proportion du partage est respectée, elle est simplement ramenée au poids de l'arc). Vous pouvez également entrer des nombres décimaux (, et . fonctionnent) ainsi que des fractions, et vous pouvez entrer 2 types de valeur différents pour un même partage (par exemple 1/2 et 0,5, 12 et 5.8, 3.5 et 7,6, 3 et 6/11, il n'est cependant pas possible de faire des fractions de nombre décimaux comme 2,5/3). La modification d'un partage se passe donc comme pour le changement de nom :

— Cliquez sur l'arc que vous souhaitez modifier

— Une boîte de dialogue va alors s'ouvrir, tout comme pour le noeud il s'agit de la même boîte de dialogue qui servirai à supprimer l'arc. La boîte comporte 2 champs modifiables, un pour chacun des deux noeuds liés par l'arc, dedans figure le gain qu'ils possède actuellement (ce gain est la valeur qu'ils ont dans leur partage respectif, mais ce peut être le gain d'un partage avec un autre agent)

— Entrez les valeurs que vous souhaitez que les agents possèdent via leur partage (si vous entrez 0 et 0, il ne réaliserons aucun partage, cela annulera leur partage s'ils en ont l'un avec l'autre, mais s'ils sont en partage avec d'autres agents ça ne changera rien)

— Cliquez ensuite sur "Confirmer", votre modification est prise en compte, s'ils ne réalisaient pas de partage ils sont désormais en partage, l'arc devient alors rouge, s'ils étaient déjà en partage l'un avec l'autre les valeurs que vous avez entrées remplacent (après normalisation) les anciennes valeur. Si vous cliquez sur "Supprimer Partage" plutôt que sur "Confirmer" cela revient à passer 0 et 0, cela annule leur partage s'ils en avaient un l'un avec l'autre (leur arc devient noir) et s'ils n'étaient pas en partage cela n'a aucune effet), ce bouton est là pour vous faire gagner du temps lorsque vous voulez retirer un partage, au lieu d'entrer 0 et 0 vous n'avez qu'à cliquer sur "Supprimer Partage", cela ne supprime pas l'arc, cela retire juste le partage entre les deux noeuds.

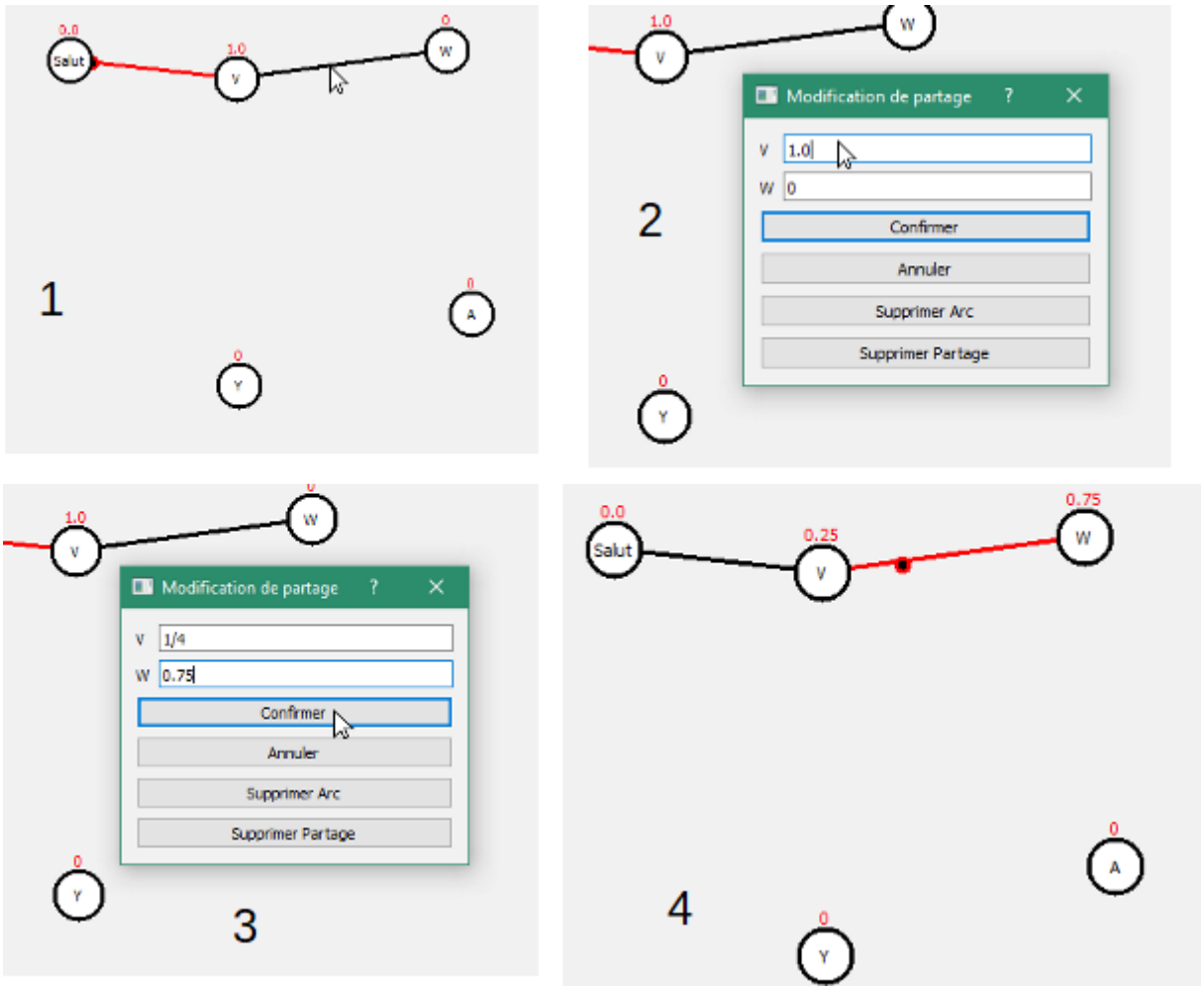


FIGURE B.22 – Illustration des indications ci-dessus en modifiant l'arc V - W, on remarque en case 2 que V possède 1 même s'il n'est pas en partage avec W, c'est le gain de son partage avec Salut, en case 3 on entre 2 types de valeurs différentes qui donnent un résultat correcte en case 4, V ayant changé d'agent avec qui il partage, Salut se retrouve à 0

Supprimer un Arc

Enfin, la dernière fonctionnalité de ce logiciel, la suppression d'un arc, la suppression d'un arc se passe exactement de la même manière que la suppression d'un noeud :

- Cliquez sur l'arc que vous souhaitez supprimer
- la boîte de dialogue du mode Select apparaît
- Cliquez sur le bouton "Supprimer Arc" (attention, "Supprimer Partage" n'a pas du tout le même effet)
- Une nouvelle boîte de dialogue va alors s'ouvrir vous demandant de confirmer la suppression de l'arc (au cas où vous changiez d'avis ou que vous ayez cliqué dessus sans le vouloir)
- Confirmez la suppression en cliquant sur "Yes", votre arc est alors supprimé (si vous cliquez sur "No" vous revenez à la première boîte de dialogue avec la modification de partage et la suppression)

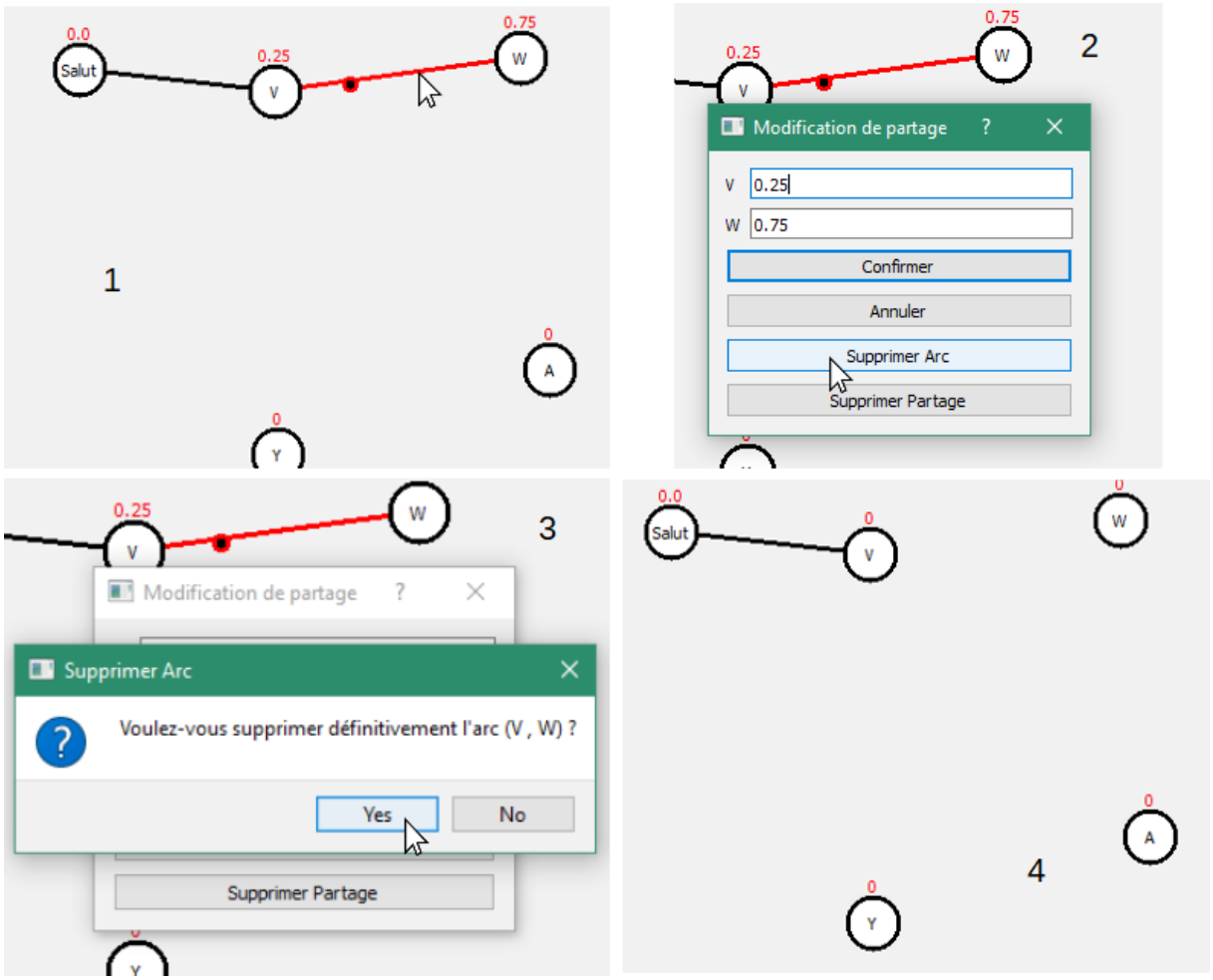


FIGURE B.23 – Illustration des indications ci-dessus en supprimant l'arc (V, W), on remarque également que du coup, n'étant plus lié, leur partage disparaît, leurs gains respectifs tombent à 0

Voilà ce qui conclut notre guide d'utilisateur, nous espérons que vous avez réussi à prendre le logiciel en main correctement, que les indications et explication étaient claires et que ce que nous proposons avec cette application vous satisfera.