PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Analyse des Graphes issus d'un processus d'écriture
Auteur: Author:	Florence Tanoï Namio
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Tanoï Namio, F. (2025). Analyse des Graphes issus d'un processus d'écriture [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/65802/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/65802/
Directeurs de recherche: Advisors:	Nadia Lahrichi, Gilles Caporossi, & Branko Ladanyi
Programme: Program:	DR-Mathématiques

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Analyse des graphes issus d'un processus d'écriture

FLORENCE TANOÏ NAMIO
Département de mathématiques et génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Avril 2025

© Florence Tanoï Namio, 2025.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Analyse des graphes issus d'un processus d'écriture

présentée par **Florence TANOÏ NAMIO**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Nadia LAHRICHI, présidente
Alain HERTZ, membre et directeur de recherche
Gilles CAPOROSSI, membre et codirecteur de recherche
Christophe LEBLAY, membre et codirecteur de recherche
Sébastien LE DIGABEL, membre
Claire DOQUET, membre externe

DÉDICACE

À ma maman, partie bien trop tôt.

Ta présence aimante, ta sagesse et ton soutien inébranlable ont été des cadeaux inestimables dans ma vie. Tu as enrichi ma vie et façonné la personne que je suis aujourd'hui.

À mes chers neveux et nièces, afin qu'ils trouvent le courage de persévérer.

REMERCIEMENTS

Je tiens, tout d'abord à remercier mes directeurs de recherche, Alain HERTZ, Gilles CAPOROSSI et Christophe LEBLAY, pour leur encadrement, leurs conseils, leur patience et leurs encouragements.

Je tiens ensuite à remercier les membres du jury d'avoir accepté d'évaluer ce travail.

Je remercie le GERAD pour son environnement de travail et l'amabilité et la disponibilité de son personnel.

Je suis profondément reconnaissante envers ma famille et mes amis pour leur soutien et leur compréhension durant les moments difficiles. Leur encouragement m'a donné la force de persévérer.

Merci particulièrement à Cécile, Évelyne, Dorothée et Édith pour leur amour et pour ces heures de causerie. Ces moments m'ont permis de maintenir le cap. Merci également à Prosper, Jean-Claude, Bruno et Théodore pour tous vos mots de réconfort.

Je tiens à exprimer ma gratitude envers Jean-Claude, Barbara et Chris pour leur affection, leur appui et la confiance qu'ils ont placée en moi.

Enfin, je tiens à exprimer toute ma reconnaissance à mon père, grâce à qui tout a été possible et qui, contre vents et marées, a toujours été là pour moi. Papa, aucun mot ne saurait suffire pour te remercier.

RÉSUMÉ

Lorsque nous écrivons un texte, nous effectuons de nombreux changements qui n'apparaissent pas toujours dans la version finale du texte. De plus, dans la plupart des cas, nous n'avons pas d'idée de l'impact de ces modifications sur le texte final. Certes, il existe plusieurs outils permettant d'analyser les textes, mais cette thèse propose des outils d'analyse des processus d'écriture basés sur le graphe des processus d'écriture. En effet, à partir du *modèle sans perte* (graphe représentant le processus d'écriture), nous utilisons les techniques mathématiques d'analyse, particulièrement la théorie des graphes, pour définir deux nouvelles notions qui servent d'outils d'aide à l'analyse des processus d'écriture. Les deux notions définies sont la communauté (ou épisode) de modification et l'indice de modification.

La notion de communauté de modification, aussi appelée épisode de modification, regroupe les sommets des graphes de processus d'écriture en ensembles traduisant les différentes modifications effectuées. La visualisation des communautés de modification proposée permet d'observer les différentes étapes qui mènent à l'obtention du texte final. La fusion de ces communautés offre une vision plus précise et concise des modifications et de l'évolution du texte. De plus, les diagrammes de distribution et de contribution synthétisent les informations contenues dans chaque communauté afin de produire un portrait plus concis du processus d'écriture.

La notion d'indice de modification donne une idée du nombre de modifications effectuées à un endroit spécifique dans le texte. Ainsi, l'importance d'un sommet est évaluée en fonction du nombre de modifications effectuées dans son voisinage, ce qui permet d'obtenir une nouvelle représentation du processus d'écriture suivant l'importance du sommet.

Finalement, des exemples d'applications possibles des notions dans l'analyse du processus d'écriture sont décrits, notamment la classification des processus d'écriture, la visualisation et l'analyse des modifications. En effet, nous montrons comment la notion de communauté de modification peut être utilisée pour construire un modèle de classification des processus d'écriture en utilisant des mesures de lisibilité.

ABSTRACT

When we write a text, we make many changes that do not always appear in the final version. Moreover, in most cases, we have no idea of the impact these changes have on the final text. Certainly, there are several tools for analyzing texts, but this thesis proposes tools for analyzing writing processes based on the graph of writing processes. Indeed, using the lossless model (a graph representing the writing process), we apply mathematical analysis techniques, particularly graph theory, to define two new concepts that serve as tools to assist in analyzing writing processes. The two defined concepts are the community (or episode) of modification and the modification index.

The notion of community of modification, also called episode of modification, groups the vertices of the graphs of writing processes into sets reflecting the different modifications made. The proposed visualization of the communities of modification allows us to observe the different steps that lead to obtaining the final text. The fusion of these communities provides a more precise and concise view of the modifications and the evolution of the text. In addition, the distribution and contribution diagrams synthesize the information contained in each community in order to produce a more concise portrait of the writing process.

The notion of modification index gives an idea of the number of modifications made at a specific place in the text. Thus, the importance of a vertex is evaluated according to the number of modifications made in its neighbourhood, which allows obtaining a new representation of the writing process according to the importance of the vertex.

Finally, examples of possible applications of the concepts in the analysis of the writing process are described, particularly the classification of writing processes, visualization and analysis of modifications. Indeed, we show how the notion of modification community can be used to build a classification model of writing processes using readability measures.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

GLOSSAIRE

Ce glossaire présente les définitions des concepts utilisés dans cette thèse.

Arbre syntaxique : aussi appelé arbre de dépendance ou arbre de constituant, permet de représenter la structure grammaticale d'une phrase. Cette analyse se fait en décomposant la phrase en ses éléments constitutifs (sujets, verbes, compléments, etc) et en illustrant les relations grammaticales entre ces éléments.

Caractère : valeur du clavier lorsque le type de frappe est une insertion.

Chronologie : l'ordre des opérations dans le fichier d'enregistrement des frappes.

Communauté (ou épisode) de modification : est un ensemble de sommets représentant les opérations d'écriture (insertion, suppression) réalisées sans déplacer le curseur. (Confère chapitre 4)

Classification de processus d'écriture : méthode d'analyse qui permet de regrouper les processus d'écriture suivant un critère que l'on choisit.

Groupe nominal : construction grammaticale constituée d'un nom, qui est le noyau du groupe, et de ses dépendants, comme des déterminants (articles, adjectifs) et/ou des compléments (prépositions, subordonnées, etc.).

Indice de modifications : est un nombre entier qui indique le nombre de fois que des modifications ont été réalisées aux alentours d'un sommet représentant un caractère du texte. (Confère chapitre 5)

Insertion : une frappe de type insertion est l'ajout d'un caractère dans le texte.

Modification : suite d'insertions et/ou de suppressions réalisées à une position donnée dans le texte.

Révision : la révision consiste à apporter un changement dans le processus d'écriture. Elle consiste soit à supprimer du texte ou à insérer du texte entre deux portions de texte déjà écrites.

Unité syntaxique : est un élément de base dans la structure d'une phrase. Elle peut être un mot ou un groupe de mots qui remplissent une fonction grammaticale spécifique.

Suppression : une frappe est une suppression si elle vise à effacer un caractère dans le texte en cours d'écriture.

CHAPITRE 1 INTRODUCTION

L'écriture occupe une place centrale dans la société, étant présente dans de nombreux aspects de la vie quotidienne, de l'éducation à la communication professionnelle, en passant par la communication informelle. L'écriture est non seulement un moyen de transmettre des informations, mais aussi un outil puissant pour la réflexion, la créativité et l'expression personnelle. C'est ce qui fait du processus d'écriture l'objet de plusieurs études. Au fil du temps, l'étude des processus d'écriture a connu une évolution remarquable. De l'analyse des manuscrits anciens à l'utilisation des technologies modernes, les recherches ont pour but essentiel de comprendre comment les rédacteurs créent et développent leurs œuvres.

Avec l'essor de l'intelligence artificielle (IA), l'écriture, comme de nombreux domaines, a connu une révolution importante. Les chercheurs et les rédacteurs utilisent désormais des outils d'IA pour analyser, améliorer et même créer des textes. Cependant, la création de textes par l'IA soulève plusieurs problèmes, notamment la baisse du niveau des étudiants. En effet, comme c'est généralement le texte final qui est évalué, on se retrouve parfois avec un texte qui ne reflète pas le niveau réel de l'étudiant. Il devient donc nécessaire de mettre en place des méthodes permettant de connaître les étapes ayant conduit au texte final. Ainsi, l'enseignant évalue non seulement le texte final, mais aussi le processus qui a permis d'aboutir à ce texte. De plus, cela permet de connaître les modifications effectuées au cours de l'écriture et, de ce fait, appréhender l'effet des différentes révisions sur la qualité du texte final. On peut alors se demander s'il est possible d'identifier le degré d'expertise d'un rédacteur en fonction de ses différentes modifications. Autrement dit, quels peuvent être les éléments caractéristiques du processus d'écriture d'un rédacteur ?

Pour comprendre les façons d'écrire et de réviser des rédacteurs, plusieurs méthodes sont utilisées. Certains chercheurs (psychologues, linguistes) ont étudié les manuscrits des grands rédacteurs (Alamargot & Lebrave, 2010 [1]). Cette méthode d'analyse s'appelle la génétique textuelle. En effet, les chercheurs suivent le cheminement de l'écriture en analysant les traces graphiques des manuscrits pour déterminer le trajet du stylo lors de l'écriture (Doquet, 2003 [35]). Cela permet de faire ressortir les versions intermédiaires et les étapes de la rédaction, de connaître l'évolution des idées, du style et de la structure du texte (pour plus de détails, voir *Éléments de critique génétique : lire les manuscrits modernes* de Grésillon (1994) [51]). D'autres chercheurs, par contre, ont élaboré des modèles traduisant le processus de production d'écriture (Flower & Hayes, 1981 [42]). Toutefois, avec l'évolution des outils informatiques, de nouvelles perspectives d'analyse se sont ouvertes. Par exemple, grâce à l'en-

registrement des différentes opérations effectuées lors de l'écriture, il est désormais possible d'accéder à beaucoup plus d'informations sur le processus d'écriture (Allen et al., 2016 [3], Conijn et al., 2019 [94]). Plusieurs méthodes sont utilisées pour analyser les fichiers log (ou fichier journal) issus de l'enregistrement du processus d'écriture, notamment les méthodes statistiques (Baaijen et al., 2012 [8], Ballier et al., 2019 [9]) ou, plus récemment, les méthodes de traitement du langage naturel (NLP)(Mahlow et al., 2024 [75], Ulasik et al., 2025 [112]). La génétique textuelle s'adapte elle aussi à l'utilisation des fichiers d'enregistrement de frappe pour l'étude de l'écriture. C'est le cas, par exemple, de l'*Étude génétique de l'écriture sur le traitement de texte d'élèves de cours moyen 2 de l'année 1995-96* réalisée par Doquet (2003) [35], qui utilise le logiciel *Genèse du texte* pour étudier la manière d'écrire des élèves. Plus récemment, les travaux de Bekius (2023) [12] permettent de voir comment les fichiers log peuvent être utilisés dans la génétique textuelle. Puisque, l'analyse des fichiers log est fastidieuse et peu aisée ; la théorie des graphes est aussi mise à contribution pour structurer l'information contenue dans ces fichiers et ainsi permettre une étude du processus d'écriture des rédacteurs (Bécotte-Boutin, 2019 [11], Caporossi & Leblay, 2011 [27]). Les graphes obtenus par la théorie des graphes permettent à la fois une visualisation et une analyse plus poussée du processus d'écriture. Cependant, certains de ces graphes n'ont pas fait l'objet d'une analyse sur le plan structurel, c'est-à-dire une étude des liens entre les sommets ; c'est le cas du modèle sans perte (Bécotte-Boutin, 2019 [11]).

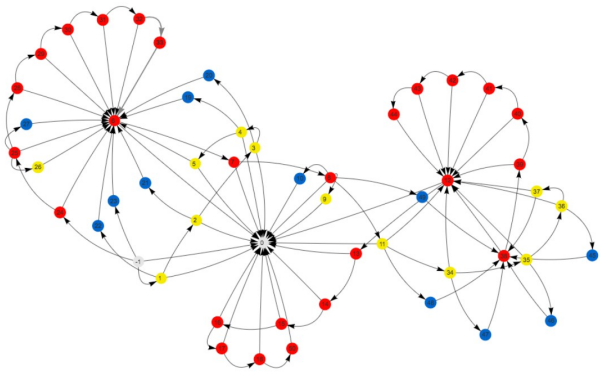


FIGURE 1.1 Exemple du *modèle sans perte* (Bécotte-Boutin, 2019 [11]).

Comme l'illustre la Figure **1.1** représentant un exemple de processus d'écriture par le *modèle sans perte*, chaque caractère est représenté par un sommet ; il est donc très facile de se retrouver avec un graphe comportant plusieurs milliers de sommets. En raison de la taille immense du graphe obtenu, il est nécessaire de disposer d'un moyen pour rechercher l'information sur le processus d'écriture contenu dans ces graphes. Mais comment décider quelle partie du graphe correspond à un aspect particulier du processus d'écriture ? Pour cela, il est

nécessaire de trouver des indicateurs qui mettent en évidence les différents aspects possibles du processus d'écriture. Dans l'étude du processus d'écriture, nous nous intéressons principalement à l'évolution du texte, c'est-à-dire à tout ce qui se passe, de la page blanche au texte final. Ainsi, ce travail se consacre particulièrement à mettre en relief l'importance du graphe de processus d'écriture dans l'analyse des processus d'écriture. En effet, nous cherchons à déterminer les éléments-clés du graphe de processus d'écriture et à déterminer comment les utiliser pour analyser le processus d'écriture.

Dans un autre ordre d'idées, la notion de lisibilité est utilisée pour évaluer la difficulté qu'un lecteur aura lors de la lecture d'un texte. En effet, la lisibilité est un indice qui renseigne sur l'accessibilité d'un texte pour un groupe précis de lecteurs (François, 2011 [47], DuBay, 2004 [37]). La lisibilité est généralement exprimée en termes du niveau d'étude nécessaire à un lecteur pour comprendre un texte donné. Ainsi, pour un texte donné, la lisibilité permet de savoir quel niveau le lecteur doit avoir pour pouvoir lire et comprendre le texte. Pour évaluer la lisibilité, les éléments du texte, tels que le nombre de mots et le nombre de phrases, sont utilisés (DuBay, 2004 [37]). Il existe plusieurs méthodes d'évaluation de la lisibilité, notamment les méthodes classiques ou traditionnelles, les méthodes utilisant le traitement automatique du langage et les méthodes de classification. Toutes ces méthodes prennent en compte des caractéristiques telles que la longueur des mots, le nombre de mots, le nombre de phrases, ainsi que la longueur des phrases. Les méthodes plus sophistiquées prennent en plus en compte la cohérence, la syntaxe, le lexique et la sémantique. En général, la lisibilité permet de classer les textes selon leur degré de difficulté et permet à chaque individu d'un niveau précis de lire et de comprendre le texte qui lui est destiné. Mais la lisibilité est aussi souvent utilisée par les rédacteurs pour améliorer la qualité de leurs textes. En effet, les rédacteurs essaient d'adapter leurs textes en fonction de la valeur de la lisibilité qu'ils souhaitent atteindre (François, 2011 [47]). Cette technique a été largement critiquée, on lui reproche le fait que le texte perde parfois tout son sens et qu'en utilisant des mots courts et des phrases courtes, cela ne rend pas forcément le texte plus compréhensible. En effet, en cherchant à adapter son écrit, de nombreuses informations se perdent. En général, lorsqu'un rédacteur écrit, il utilise des termes qui lui sont familiers et construit des phrases à partir de ses connaissances grammaticales et syntaxiques. Ainsi, en évaluant le texte, on évalue aussi, en quelque sorte, son style d'écriture. Il est alors tout à fait naturel de se demander si l'analyse du processus d'écriture pourrait être améliorée par la combinaison des graphes de processus d'écriture et des mesures de lisibilité.

Objectif de recherche

En partant du graphe de représentation du processus d'écriture du *modèle sans perte* (Bécotte-

Boutin, 2019 [11]), mes recherches visent à identifier des éléments caractéristiques de ce graphe, tels que des structures de communautés et des sommets ayant une importance particulière, qui pourraient permettre d'identifier des traits caractéristiques du processus d'écriture d'un rédacteur. De plus, en se basant sur les communautés détectées et les autres indices identifiés, mes recherches visent à établir un exemple de modèle qui pourra aider à la classification des processus d'écriture en fonction de la variation de la lisibilité durant le processus, et à fournir une autre représentation plus synthétique du processus d'écriture.

Pour répondre à ces différents objectifs, nous avons défini deux nouveaux concepts :

* Les **communautés de modifications** : elles permettent de regrouper les sommets des graphes de processus d'écriture en ensembles qui traduisent les différents épisodes de modifications effectuées avant l'obtention d'un texte final.
* Les **indices de modifications** : ils permettent d'identifier les sommets représentant les parties du texte les plus modifiées au cours du processus d'écriture.

Les notions que nous définissons sont essentiellement basées sur le type de sommet (insertion ou suppression) et/ou le lien entre les sommets (arcs). En effet, nous utilisons rarement les caractères représentés par les sommets. En d'autres termes, nous utilisons essentiellement la structure du graphe et non les caractères représentés par les sommets (le texte). De ce fait, les données utilisées dans cette thèse sont de trois types : données réelles, données simulées et données créées manuellement.

— **Données créées manuellement :** lorsque nous définissons les notions qui ne nécessitent pas de considérer les caractères représentés par les sommets, les processus d'écriture sont alors fabriqués de toute pièce en utilisant le logiciel d'enregistrement de frappes GGXLog (Caporossi et al., 2020 [28]). C'est le cas pour les exemples d'illustrations.

— **Données simulées :** lorsque nous comparons des communautés de modifications ou des indices de modifications, et que cela nécessite des graphes de processus d'écriture avec une multitude de sommets, mais que les caractères des sommets ne sont toutefois pas pris en compte, les processus d'écriture sont alors simulés en utilisant la méthode détaillée au chapitre 3.

— **Données réelles :** lorsque nous utilisons les processus d'écriture pour des applications avec des notions telles que la lisibilité qui nécessitent de considérer les caractères représentés par les sommets, les processus d'écriture proviennent alors d'un ensemble de journaux de frappe intitulé ProTEXT réalisé par Miletić et al.(2022) [78]. Nous utilisons la version en libre accès de la base de données Pro-TEXT. Pro-TEXT est un corpus de

fichiers logs écrits en français dans le cadre du projet interdisciplinaire axé sur le processus d'écriture Pro-TEXT (les processus de textualisation) [78]. Le projet regroupe des psycholinguistes, des linguistes, des spécialistes en traduction automatique du langage (TAL) et des spécialistes de traduction. La version du corpus que nous utilisons contient 45 textes (en français) d'écoliers regroupés en trois groupes d'âge : `CE2` (3^{e} année d'école primaire) environ 8 ans, `CM2` (5^{e} année d'école primaire) environ 10 ans et `C6` (1^{re} année du secondaire) environ 11 ans. Les participants devaient produire deux types de textes : un argumentatif et un narratif. Les textes ont été enregistrés à l'école lors d'une séance d'écriture au clavier [78]. Le tableau suivant présente un résumé des données :

	`Primaire` 3	`Primaire` 5	`Secondaire` 1
`Argumentatif`	9	8	6
`Narratif`	10	5	7

Cette thèse comprend, au chapitre 2, une revue de la littérature sur les études réalisées concernant les processus d'écriture, l'analyse des graphes et les méthodes d'évaluation de texte, telles que la lisibilité. Le chapitre 3 présente les outils et les méthodes utilisés dans cette thèse. Les deux nouveaux concepts de la thèse seront présentés respectivement aux chapitres 4 et 5. Le chapitre 4 portera sur les communautés de modification, leur définition et leurs représentations. Le chapitre 5 traitera de la notion d'indice de modification sous différents angles. Au chapitre 6, il sera question de présenter quelques exemples d'application des communautés de modifications et des indices de modification. Enfin, le chapitre 7 présente les conclusions.

CHAPITRE 2 REVUE DE LITTÉRATURE

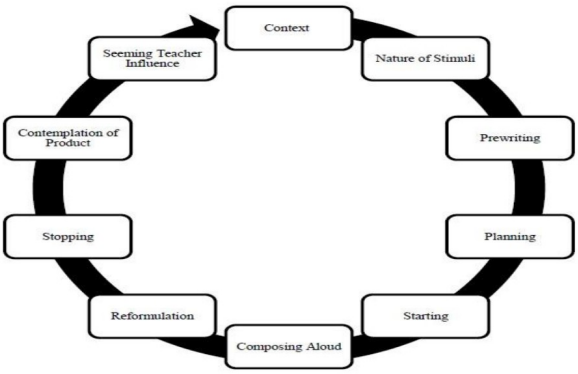
La revue de la littérature portera essentiellement sur les processus d'écriture et l'analyse des graphes. La première partie sera centrée sur les différentes techniques de représentation d'un processus d'écriture et leurs avantages. La deuxième partie portera sur les éléments pris en compte dans l'analyse des graphes et, enfin, la troisième partie donnera un aperçu des différentes façons d'évaluer la lisibilité d'un texte.

2.1 Processus d'écriture

Le processus d'écriture est un processus complexe à étudier. Au départ, le processus d'écriture était décrit comme une activité linéaire. Suivant ce concept, le rédacteur planifie ce qu'il veut écrire, puis rédige un projet qu'il révise ensuite (Hill et al., 1991 [59]). Ainsi, la révision était considérée comme une activité qui se faisait à l'achèvement d'un projet (Murray, 1978 [80], Hill et al., 1991 [59]). Le processus d'écriture est alors assimilé à la croissance d'une plante (Rohman & Wlecke, 1964 [96]), en raison des différentes étapes de la croissance d'une idée d'écriture à partir de pensées et de sensations, pour aboutir à des mots sur le papier (Sharp, 2016 [101]). Le processus d'écriture est alors divisé en trois étapes : la préécriture, qui désigne toute chose qui a lieu avant l'apparition du texte sur le papier ; l'écriture du texte ; et enfin la réécriture, qui consiste à réviser le texte écrit (Rohman & Wlecke 1964 [96]). Ces étapes sont aussi appelées, par Murray (1978) [80], la prévision, la vision et la révision. Cependant, les recherches montrent que le processus d'écriture est plus complexe qu'une présentation linéaire. Les travaux d'Emig (1967 [38], 1971 [39]) ont remis en cause la linéarité du processus d'écriture ; pour elle, l'écriture peut inclure les trois étapes définies précédemment, mais ces étapes se produisaient de manière récursive. Elle définit alors dix étapes à la composition d'un texte et met en évidence l'influence de la parole dans le processus d'écriture (confère Figure **2.1**).

De plus, avec les travaux de Hayes et Flower (1980) [57], le processus d'écriture est considéré comme un processus cognitif structuré hiérarchiquement, dans lequel l'écriture d'un texte est effectuée de manière récursive. Ils ont proposé que, lors de la production écrite, un rédacteur expert utilise trois processus cognitifs : la planification, la textualisation et la révision.

Comme l'illustre la Figure **2.2**, la planification est considérée comme la première étape, suivie de la textualisation et de la révision. Pendant la planification, le rédacteur a recours à ses connaissances et au sujet de rédaction pour élaborer un objectif et un plan d'écriture. Lors-



Contexte : facteur externe qui influence l'écrivain.
Nature of Stimuli (Nature des stimuli) : événement qui active et entretient le processus d'écriture.
Prewriting (Pré-écriture) : le temps pendant lequel est perçue l'idée de composition ainsi que les perceptions de l'acte de composition de cette idée.
Composing Aloud (composer à voix haute) : la période où l'écrivain dit à haute voix son projet. Il réfléchit en parlant.
Reformulation : l'écrivain réécrit autrement son texte.
Stopping (Arrêt) : il arrête d'écrire c'est la fin du brouillon.
Contemplation of production (Réflexion sur la production) : le moment où un écrivain réfléchit à la qualité et au statut de ses écrits, en particulier à la manière dont les lecteurs vont comprendre son texte.
Seeming Teacher Influence (Influence apparente de l'enseignant) : réécriture du texte en fonction des corrections reçues dans le passé.

FIGURE 2.1 Les dix dimensions du processus de composition d'Emig (1971) [39].

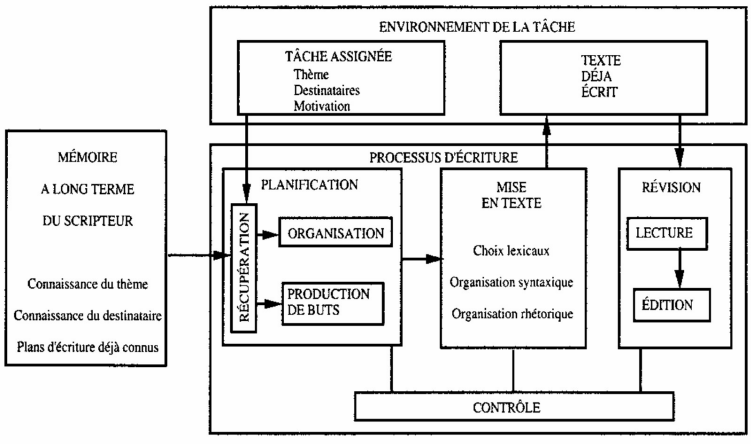


FIGURE 2.2 Modèle de Hayes-Flower (1980)(Doquet C. [35]).

qu'il termine avec la planification, il passe à la textualisation, qui consiste à produire un texte suivant le plan établi. Ensuite, durant la révision, il lit, identifie et corrige les erreurs et les imperfections. Au fil du temps, ce modèle a été amélioré avec l'ajout de la motivation parmi les éléments qui influencent le rédacteur dans la rédaction (Hayes, 1996 [58]) et la suppression de la planification et de la révision (Hayes et al., 2012 [15]). En effet, la planification et la révision sont considérées comme une application particulière du processus d'écriture. Autrement dit, lorsqu'un rédacteur planifie et révise, il écrit un texte, mais de nature différente. Avec l'évolution des technologies, les recherches se sont tournées vers d'autres méthodes d'analyse et de visualisation. Pour la visualisation du processus d'écriture, il existe trois grands types : les représentations linéaires , les représentations avec des courbes et les représentations par des graphes.

Représentations linéaires : La représentation linéaire est une illustration du texte avec des symboles et des polices différents qui traduisent les différentes opérations effectuées lors de l'écriture du texte. D'après Bécotte-Boutin (2019) [11], on distingue la notation-S, la variante de la notation-S, la transcription linéaire génétique, la représentation linéaire de Scriptlog et la représentation TimeLine. La notation-S est une illustration qui décrit le développement d'un texte en représentant l'ensemble du texte et toutes les révisions que le rédacteur a effectuées lors de l'écriture du texte (Kollberg, 1997 [67]). Ci-dessous, un exemple illustre l'écriture d'une phrase avec la notation-S.

`Juste un {petit}`^{3}`|`$_4$`essai pou[e|`$_1$`]`^{1}`r{la}`^{5}`[ma]`^{4}`|`$_5$`revue de littératu[e|`$_2$`]`^{2}`re.|`$_3$

Version finale du texte : `Juste un petit essai pour la revue de la littérature.`

Les symboles utilisés pour la représentation de la notation-S, d'après Kollberg (1997) [67] sont :

$|_i$ pour signifier une pause (interruption) avec le numéro de séquence i ;

{`texte inséré`}^{i} pour une insertion après une interruption i ;

[`texte supprimé`]^i pour une suppression après l'interruption i.

Perrin D. (2003) [91] présente une version plus détaillée de la notation-S. En effet, dans cette version, les sources sont différenciées par des polices différentes et les parties de texte copiées d'une autre source sont soulignées. La Figure **2.3** en donne un exemple.

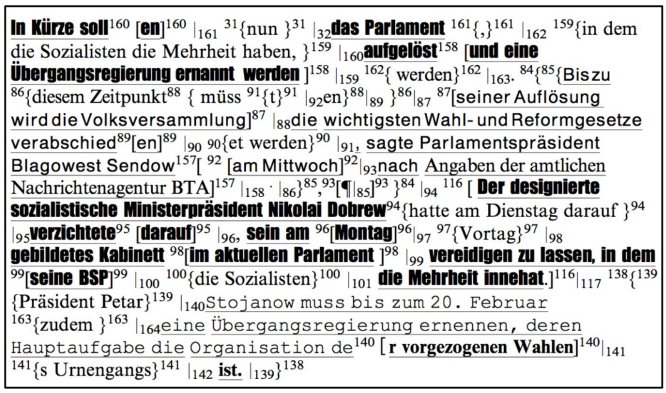


FIGURE 2.3 Version détaillée de la notation-S, Perrin, D. (2003).

Dans la transcription génétique, il y a ajout d'icônes qui signalent les mouvements du curseur. La Figue **2.4** est une illustration réalisée avec le logiciel GGXlog (Caporossi et al., 2020 [28]).

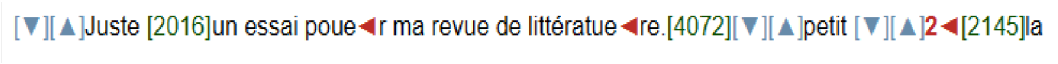


FIGURE 2.4 Transcription linéaire génétique réalisée avec GGXlog (Caporossi et al., 2020 [28]).

Où [▲][▼] est le déplacement du curseur, ►le retour en arrière, ◄la suppression et [durée de la pause en millième de seconde].

La représentation linéaire de scriptlog est une simplification du fichier log. En effet, c'est une représentation des différentes opérations effectuées dans le texte. Elle ne permet pas de bien voir les retours dans le texte (Wengelin et al., 2009, Bécotte-Boutin, 2019 [11]). La Figure **2.4** donne une illustration obtenue avec le logiciel GGxlog (Caporossi et al., 2020 [28]).

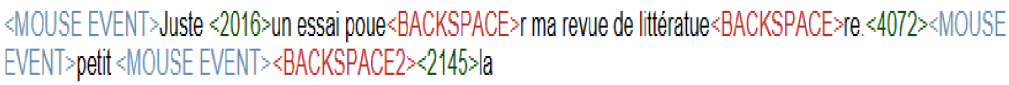


FIGURE 2.5 Représentation linéaire de scriptlog obtenue avec GGxlog (Caporossi et al., 2020 [28]).

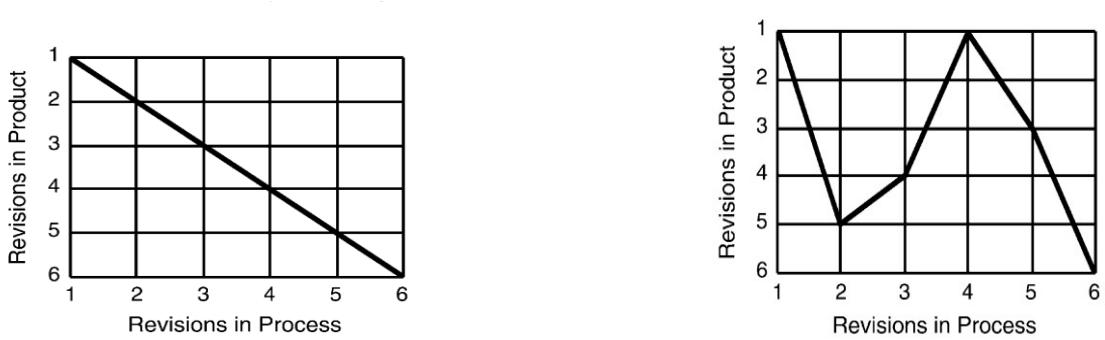
Où < MOUSE EVENT> est le déplacement du curseur, <BACKSPACE> le retour en arrière, <DELETE> la suppression et <durée de la pause>.

En résumé, les représentations linéaires permettent de voir les différentes modifications apportées au texte. Cependant, avec la multitude de symboles utilisés, il n'est pas aisé d'analyser de longs textes.

Représentations par des courbes : Ici, le processus d'écriture est représenté par des courbes qui donnent l'évolution du texte au cours de sa construction. En général, ces courbes mettent en évidence la temporalité, la chronologie et la spatialité présentes dans le processus d'écriture. En d'autres termes, elles permettent d'avoir une idée du temps et de la position relative où des actions sont effectuées dans le texte au cours de sa construction. Pour réaliser ces courbes, différentes variables sont utilisées et, selon ces variables, on a une courbe différente. On peut citer le diagramme de progression, la représentation au fil de la plume, le GrapheLS et les représentations provenant du SIG (système d'information géographique).

— Le `diagramme de progression` représente l'évolution du texte en utilisant les révisions réalisées. Dans le diagramme de progression, chaque point représente une révision, l'axe des abscisses donne l'ordre des révisions dans le processus d'écriture et l'axe des

ordonnées donne l'ordre des révisions dans le produit fini. Par exemple, si un rédacteur corrigeait immédiatement chaque faute de frappe sans revenir en arrière pour supprimer ou insérer quoi que ce soit dans le texte précédent, alors on aurait une ligne droite du coin supérieur gauche au coin inférieur droit, comme le montre la Figure **2.6a**. Par contre, si le rédacteur fait des allers-retours dans le texte pendant les révisions, alors on aura une courbe des variations comme illustrée à la Figure **2.6b**. Dans cette figure, on voit par exemple que la première révision a lieu au début du texte, la deuxième à la fin et la quatrième au début du texte (Perrin D., 2003 [91]).



(a) (b)

FIGURE 2.6 Diagramme de progression, Perrin D. (2003) [91].

— La représentation `au fil de la plume` issue du logiciel Genèse du texte (Foucambert, 1995 [45]) permet d'illustrer le déplacement du curseur sur les différentes lignes du texte en fonction du temps. Ainsi, chaque point de la courbe indique la ligne où se situe le curseur en un instant donné. La figure **2.7** donne un exemple de la représentation d'un processus d'écriture d'un texte de 35 lignes tiré de Doquet C. (1995) [36].

— `Le GrapheLS` est une représentation du processus d'écriture qui ressemble quelque peu au diagramme de progression, car tous deux sont issus de la notation-S. Cependant, le GrapheLS permet d'avoir, en fonction du temps, dans un même graphique, les courbes qui traduisent la progression du nombre de caractères, la position du curseur et la longueur du texte (Bécotte-Boutin, 2019 [11]).

— Les `représentations provenant du SIG` (système d'information géographique) permettent de représenter le processus d'écriture sous forme de plusieurs courbes, tout comme le GrapheLS, mais avec en plus la possibilité d'avoir un aperçu instantané du processus d'écriture permettant de connaître les détails du processus pour un point donné. Il existe aussi deux autres variantes :

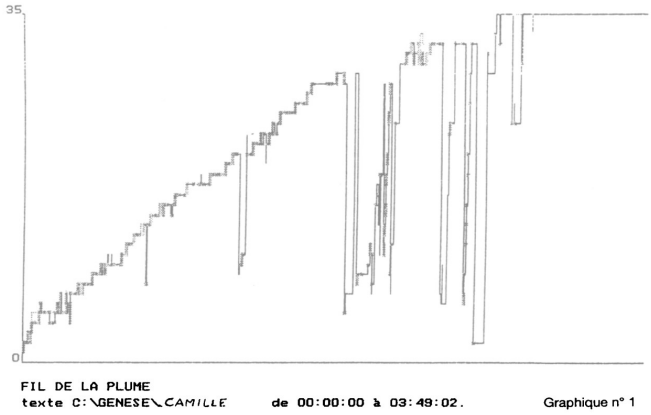


FIGURE 2.7 Représentation au fil de la plume (Doquet C. (1995) [36]).

* la représentation SIG combinée aux sources, qui permet de connaître les sources que le rédacteur a utilisées ;
* la représentation SIG combinée aux pauses, qui permet d'avoir une idée des pauses que le rédacteur a réalisées lors du processus d'écriture.

En résumé, les représentations avec les courbes permettent d'avoir un aperçu général du processus d'écriture en prenant en compte la spatialité et la temporalité. Cependant, avec ces représentations, on n'a pas accès au texte écrit, contrairement à celles des représentations linéaires.

Représentations par les graphes : Ici, la théorie mathématique des graphes est utilisée pour structurer et visualiser l'information. On utilise le fait que le graphe est une illustration qui permet de relier des éléments entre eux. Ainsi, les opérations d'écriture, tels que les insertions et les suppressions, sont reliés entre eux suivant leur ordre ou leur position dans le texte. On peut citer la représentation de Caporossi et Leblay (2011) [27] qui tient compte de la chronologie et des liens spatiaux entre les insertions encore présentes dans le texte final. En effet, les sommets représentent une suite d'insertions ou une suite de suppressions, et les arêtes, les relations spatiales ou chronologiques entre les sommets (Leblay & Caporossi, 2016 [70]). La Figure **2.8** donne un exemple de représentation tiré de Leblay et Caporossi (2016) [70]. Les couleurs des sommets donnent une idée des actions réalisées : le rouge représente les ajouts, le jaune les ajouts supprimés et le bleu les suppressions. La taille des sommets est proportionnelle à la quantité d'ajout ou de suppression effectuée. Les liens chronologiques sont représentés par les lignes continues et les liens topographiques par les lignes discontinues rouges. Nous avons deux types de liens chronologiques :

— Les lignes noires qui relient les opérations consécutives. On peut les retrouver entre :

1. deux sommets rouges,
2. un sommet rouge et un sommet jaune,
3. un sommet rouge et un sommet bleu,
4. un sommet bleu et un sommet jaune.

— Les lignes bleues qui relient les sommets de suppression et les sommets des éléments qu'elles suppriment. On les retrouve essentiellement entre un sommet bleu et un sommet jaune.

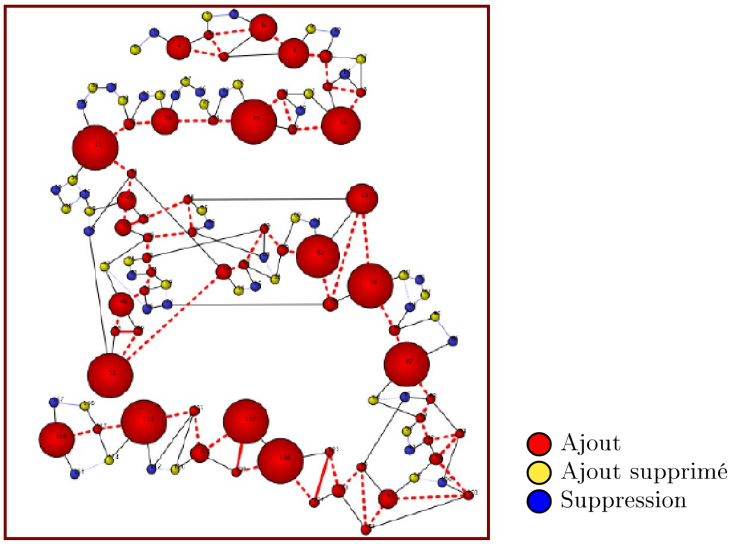


FIGURE 2.8 Représentation de Caporossi et Leblay (2016) [70].

Ce graphe aide à voir la dynamique du processus d'écriture. Cependant, à partir du graphe final, on ne peut pas retracer les différentes versions du texte obtenues durant le processus d'écriture. De plus, un même graphe peut représenter plusieurs processus d'écriture différents (Bécotte-Boutin, 2019 [11]).

Pour pallier à cela, Bécotte-Boutin (2019) [11] présente deux graphes : la *visualisation progressive* et le *modèle sans perte*. Ces graphes sont orientés et prennent en compte la chronologie et la spatialité. Cependant, chaque caractère inséré ou supprimé est représenté par un sommet, la relation spatiale est donnée par les arcs et la relation chronologique est indiquée par les numéros des sommets.

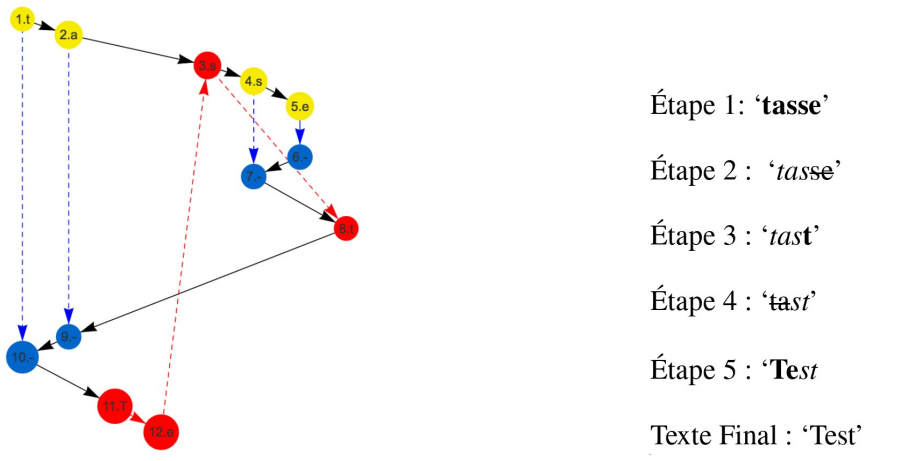
La Figure **2.9** donne un exemple de la *visualisation progressive* :

— les arcs pointillés rouges permettent la lecture du texte final ;

— les arcs pointillés bleus relient les sommets des insertions supprimées (en jaune) et les sommets de suppression (en bleu) qui les suppriment ;

— les arcs noirs indiquent la chronologie des événements, également représentée par les numéros des sommets.

Bien que les axes ne soient pas visibles dans cette représentation, les sommets sont placés selon des coordonnées (x, y) où l'axe des x correspond à la position de l'opération effectuée dans le texte et l'axe des y donne le temps correspondant au moment où l'opération a été réalisée. En résumé, la *visualisation progressive* combine à la fois les avantages des représentations linéaires, des SIG et des graphes ; en effet, elle permet d'avoir toutes les variables utilisées dans les processus d'écriture. La *visualisation progressive* peut être utilisée pour une étude globale et spécifique d'un texte, car elle est dynamique. De plus, elle permet d'avoir un graphe propre à chaque processus.



(a) Graphe (b) Processus d'écriture

FIGURE 2.9 *Visualisation progressive.*

Le *modèle sans perte* est un modèle qui permet de modéliser le fichier log de telle sorte que le graphe obtenu puisse être utilisé pour des analyses structurelles (Bécotte-Boutin, 2019 [11]). Tout comme la *visualisation progressive*, chaque insertion ou suppression est représentée par un sommet. Les arcs représentent le lien spatial entre les insertions, et le lien chronologique est donné par les numéros des sommets. Le *modèle sans perte* comporte moins de variables que la vision progressive, mais toutes les informations du processus d'écriture sont contenues dans sa structure ; c'est pour cette raison qu'on l'appelle « le *modèle sans perte* ».

La Figure **2.10** donne un exemple du *modèle sans perte*. Contrairement à la *visualisation progressive*, les sommets du *modèle sans perte* ne sont pas placés suivant des coordonnées. De

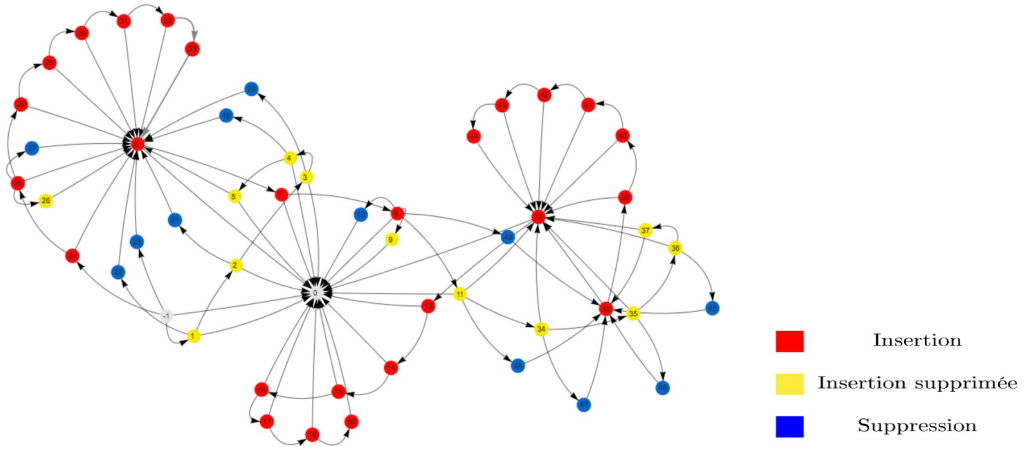


FIGURE 2.10 Modèle sans perte.

plus, il n'y a pas de lien entre un sommet d'une suppression et l'insertion qu'elle supprime. Ceci est dû au fait que la *visualisation progressive* est essentiellement dédiée à l'analyse visuelle, alors que le *modèle sans perte* n'est pas uniquement destiné à la visualisation, mais plutôt à une analyse structurelle du graphe obtenu par le modèle.

Pour résumer, la représentation du processus d'écriture est passée d'un modèle avec des flèches et des rectangles à des modèles permettant de visualiser la spatialité et la chronologie des processus d'écriture. De plus, le graphe obtenu par Bécotte-Boutin (2019) [11] contient toutes les informations du processus d'écriture dans sa structure, ce qui rend son analyse d'autant plus intéressante, car analyser sa structure revient à analyser le processus d'écriture.

2.2 Analyse de graphes

Un graphe est une représentation mathématique dans laquelle des points ou sommets sont reliés par des lignes. Les graphes sont utilisés pour modéliser des problèmes dans différents domaines, notamment les réseaux sociaux (Girvan & Newman, 2002 [50]), les problèmes de transport, et bien d'autres. Les graphes obtenus sont très souvent de grande taille. On utilise alors différentes techniques pour extraire de l'information de ces graphes. Ces techniques d'analyses dépendent du type de graphe et de l'information que l'on souhaite extraire. Le but général de l'analyse d'un graphe est soit de détecter des motifs particuliers dans ce graphe, soit de trouver les éléments (sommets, arêtes ou arcs) les plus importants du graphe selon l'information que l'on veut extraire. Les motifs sont généralement appelés des communautés ou clusters, mais reposent sur le même principe : mettre ensemble des sommets qui sont

fortement liés entre eux et faiblement avec les autres sommets (Girvan & Newman, 2002 [50]). Les indices de centralité, de leur côté, permettent de quantifier une intuition selon laquelle, dans la plupart des graphes, certains sommets ou arêtes sont plus importants que d'autres.

Nous divisons l'analyse des graphes en deux parties. La première partie est consacrée aux méthodes de détection des communautés et la seconde partie, aux indices de centralité.

2.2.1 Méthodes de détection de communautés

En général, détecter des communautés dans des graphes, consiste à trouver des sous-ensembles de sommets connectés de telle sorte que les sommets d'un même sous-ensemble aient beaucoup de liens entre eux et peu avec les sommets des autres sous-ensembles. Ces liens peuvent prendre la forme d'arêtes, d'arcs ou d'attributs, selon que nous soyons dans le cas d'un graphe orienté ou non, étiqueté ou non. La plupart des méthodes de détection de communautés sont conçues pour les graphes sociaux et les graphes dans le domaine de la biologie. Il n'existe aucune méthode de détection de communautés se rapportant directement au graphe de processus d'écriture que nous étudions. Nous analyserons donc toutes les méthodes de détection de communautés susceptibles d'être utilisées dans le cadre des graphes du *modèle sans perte*. Pour faciliter notre revue de la littérature, nous divisons les méthodes de détection de communautés en deux groupes : les méthodes pour les graphes simples et les méthodes pour les graphes multidimensionnelles.

Graphes simples

Les graphes simples sont des graphes dans lesquels il y a au plus une arête (arc) entre les sommets (Bondy & Murty, 2008 [18]). L'un des plus importants regroupements des méthodes de détection de communautés existant dans ce domaine est l'article de Fortunato (2010) [44]. La détection des communautés est reliée à la notion de densité. En effet, si un graphe est peu dense, alors il sera possible de détecter une structure de communauté. La densité d'un graphe est vue en termes d'arêtes (arcs) entre les sommets, ainsi un graphe peu dense est un graphe dont le nombre d'arêtes (arcs) est du même ordre de grandeur que le nombre de sommets. Une communauté n'a pas une définition universelle (Fortunato, 2010 [44]) ; en effet, suivant le (ou les) critère(s) choisi(s), on peut trouver des communautés dans les graphes. Une communauté peut, par exemple, être définie, comme l'un des ensembles de sommets suivant :

— Tous les sommets sont reliés les uns aux autres (clique) ;

— Tous les sommets d'un sous-graphe à n sommets sont reliés par des arêtes les uns aux

autres (n-cliques). (Lucf 1950 [74], Alba 1973 [2]) ;
— Tous les sommets sont reliés par des chemins ne dépassant pas un nombre n fixé d'arêtes (n-clan) (Mokken 1979 [79]) ;
— Chaque sommet est lié à au moins k sommets du sous-graphe (k-core). (Seidman, 1983 [99]) ;
— La densité intra-communauté δ_{int} est supérieure à un seuil, avec

$$\delta_{int} = \frac{\texttt{nombre d'arêtes de la communauté}}{\texttt{nombre d'arêtes possible de la communauté}}.$$

Si les sommets peuvent être représentés dans l'espace euclidien en fonction de leurs positions, alors la distance entre les sommets dans cet espace est considérée pour la construction des communautés. La distance la plus utilisée est la *similarité cosinus* (Fortunato, 2010 [44]). Cependant, d'autres distances, telles que la distance euclidienne, la L_2-norme ou la L_∞-norme sont également utilisées. Il existe plusieurs autres distances qui sont directement liées à la structure du graphe, telles que la distance associée à la matrice d'adjacence du graphe (Fortunalo, 2010 [44] ; Lorrain & White, 2022 [73]) qui part du principe que deux sommets différents sont équivalents s'ils ont les mêmes voisins. En utilisant ces différentes définitions de communautés, Fortunato a classé les méthodes de détection de communautés en huit catégories [44], entre autres :

1. les méthodes traditionnelles
2. les algorithmes de divisions
3. les méthodes basées sur la modularité
4. les algorithmes de partitionnement spectral
5. les algorithmes dynamiques
6. les méthodes basées sur l'inférence statistique
7. l'extraction des communautés recouvrantes
8. les méthodes multi-résolutions

Les méthodes traditionnelles sont essentiellement des méthodes de partitionnement de graphes, qui consistent à diviser les sommets en un nombre spécifique de groupes de taille prédéfinie, de sorte que le nombre d'arêtes reliant les groupes soit minimal. L'algorithme de Kernighan et Lin (1970) [64] est l'une des premières procédures pour le partitionnement de graphes. Le principal inconvénient de cette méthode est le fait d'imposer la taille et le nombre de communautés. Pour pallier cela, une variante est utilisée : le partitionnement hiérarchique qui peut être ascendant ou descendant, selon que l'on débute le partitionnement avec chaque sommet seul dans sa communauté ou avec tous les sommets dans une même communauté. Le

partitionnement hiérarchique se base sur des fonctions de similarité entre les sommets pour soit les regrouper (méthode ascendante), soit les séparer (méthode descendante). L'avantage des méthodes hiérarchiques est qu'on n'a pas besoin de préciser à l'avance ni la taille ni le nombre de communautés.

Les méthodes avec algorithmes de division sont basées sur le principe qu'une manière simple d'identifier les communautés dans un graphe consiste à détecter les arêtes qui relient les sommets des différentes communautés et de les enlever. L'algorithme le plus utilisé dans ce domaine est celui de Girvan et Newman (2002 [50], 2004 [86]), car il a marqué le début d'une nouvelle ère dans la détection de communautés. Les méthodes de Girvan et Newman sont centrées autour du concept de centralité d'intermédiarité, qui exprime la fréquence de participation des arêtes à un processus. Ce concept est défini de trois manières différentes :

— La centralité d'intermédiarité des arêtes : L'intermédiarité des arêtes est le nombre de chemins les plus courts entre toutes les paires de sommets qui passent par l'arête. Il s'agit d'une extension aux autres arêtes du concept populaire d'interdépendance des sites, introduit par Freeman en 1977 [48], qui exprime l'importance des arêtes dans des processus tels que la diffusion de l'information, où l'information passe généralement par les chemins les plus courts.

— Centralité d'intermédiarité par marche aléatoire : L'intermédiarité de la marche aléatoire est la fréquence des passages sur l'arête d'un marcheur aléatoire parcourant le graphe (Newman & Girvan, 2004 [83] ; Newman, 2003 [86]). En effet, le marcheur se déplace à partir d'un sommet, suit chaque arête adjacente avec la même probabilité.

— Centralité d'intermédiarité par flux : L'intermédiarité des flux de courant est définie en considérant le graphe comme un réseau de résistances, où les arêtes ont une résistance unitaire. Si une différence de tension (potentiel) est appliquée entre deux sommets, chaque arête transporte une certaine quantité de courant, qui peut être calculée en résolvant les équations de Kirchoff (Newman & Girvan, 2004 [83] ; Newman, 2003 [86]).

Les méthodes basées sur la modularité : La modularité (Newman & Girvan, 2004 [86]) est définie par

$$Q = \sum_{i=1}^{n_c} \left(m_{ii} - a_i^2 \right) \tag{2.1}$$

où

- n_c est le nombre de communautés
- $M = (m_{ij})_{1 \leqslant i,j \leqslant n_c}$ est une matrice symétrique avec m_{ij} représentant la proportion d'arêtes du graphe qui relient les sommets de la communauté i à ceux de la communauté j ;

- $a_i = \sum_{j=1}^{n_c} m_{ij}$ représente la proportion d'arêtes du graphe qui relient les sommets de la communauté i aux sommets des autres communautés.

La notion de modularité est aussi souvent définie en fonction des degrés des sommets (Fortunato, 2010 [44]). Le degré d'un sommet est le nombre de liens reliant ce sommet aux autres sommets.

$$Q = \sum_{i=1}^{n_c} \left[\frac{l_i}{e} - \left(\frac{k_i}{2e} \right)^2 \right] \tag{2.2}$$

où

- n_c est le nombre de communautés
- l_i est le nombre d'arêtes à l'intérieur de la communauté i ;
- k_i la somme des degrés des sommets de la communauté i.
- e est le nombre d'arêtes du graphe ;

Avec le temps, la modularité a subi plusieurs modifications et extensions pour tenir compte, par exemple, de l'orientation des graphes et des poids des arêtes (Fortunato, 2010 [44]).
Dans la plupart des cas, des valeurs élevées de la modularité présagent de bonnes partitions. Ainsi, la partition correspondant à la valeur maximale de la modularité sur un graphe donné devrait être la meilleure. C'est sur ce principe que plusieurs algorithmes utilisent la modularité comme fonction d'évaluation des communautés détectées ou une version modifiée de la modularité. Parmi ces méthodes, on trouve la méthode gloutonne (Newman, 2004 [84]) et le recuit simulé (Fortunato, 2010 [44]) qui déterminent des partitions d'ensembles de sommets en utilisant la modularité comme fonction de qualité pour évaluer les partitions obtenues.

Parmi toutes les méthodes existantes, l'une des plus utilisées est la modularité de Newman et Girvan (2004) [86], car elle est à la fois un critère de détection de communautés et une fonction d'évaluation des communautés détectées. Bien que la modularité soit fortement utilisée, elle a sa limite de résolution. En effet, elle n'arrive pas à bien distinguer les communautés lorsqu'il y a des communautés de tailles différentes dans un même graphe.

Graphes multidimensionnels

Un graphe multidimensionnel ou attribué est un graphe dans lequel il peut exister plus d'un lien entre deux sommets. Pour détecter les communautés dans ces graphes, deux grands groupes de méthodes existent (Atzmueller et al., 2021) [5]. Certaines méthodes considèrent le graphe comme unidimensionnel en prenant une dimension de référence et utilisent les autres dimensions (attributs) pour améliorer les communautés obtenues. D'autres, en revanche, considèrent toutes les dimensions lors de la définition des communautés.

Cas 1 : Utiliser les attributs pour améliorer les communautés obtenues. Dans cette catégorie, la structure générale des méthodes, très souvent utilisées, est la suivante : rendre le graphe unidimensionnel, puis appliquer une méthode de détection de communautés des graphes unidimensionnels, et enfin évaluer la pertinence des communautés. L'article de Bothorel et al. (2015) [22] présente une subdivision des méthodes en six groupes suivant les différentes approches utilisées. Il y a, par exemple, la réduction du graphe en un graphe pondéré, où le poids de chaque arête correspond à la similarité des attributs des sommets de l'arête. La similarité des attributs peut être définie de plusieurs façons. Par exemple, Neville et al.(2003) [82] utilisent comme similarité des attributs entre deux sommets le nombre d'attributs qui ont les mêmes valeurs pour deux sommets. Une autre approche consiste à supprimer tout le réseau et à définir une distance entre les différents sommets. La distance parfois définie prend en compte à la fois les attributs et la structure du graphe. C'est le cas, par exemple, de Combe et al. (2012) [30] qui prend comme distance la combinaison linéaire de la similarité des attributs et de la distance définie par les arêtes du graphe. (Pour plus de distance, voir [22]).

Cas 2 : Considérer les attributs lors de la définition des communautés. Ici, la définition d'une communauté prend en compte tous les attributs du graphe. L'article d'Atzmueller et al. (2021) [5] présente l'essentiel des méthodes utilisant cette approche de détection de communautés. D'une part, il existe des méthodes qui considèrent les attributs lors de la formulation du modèle de détection des communautés, sans tenir compte de leurs valeurs. Les communautés résultantes de cette étape seront ensuite traitées en fonction des valeurs des attributs pour obtenir des communautés répondant aux critères souhaités. Dans cette catégorie, ce qui différencie les méthodes les unes des autres est la première étape. En effet, dans cette étape, certains utilisent la notion de cohésion des attributs (Moser et al. 2009, Gunnenman et al. 2010 [53], Gunnenman et al. 2013c [54]), d'autres, la notion de densité pour des valeurs d'attributs limitées par des intervalles (Gunnenman et al. (2011) [52]) ou encore les distances entre attributs (Du et al. (2017) [106], Sun et al. [107]).
D'autre part, il existe des méthodes qui prennent simultanément en compte l'attribut et la valeur des attributs des sommets lors de la formulation du modèle. Cette façon de faire permet d'éviter de parcourir le modèle pour identifier les valeurs pertinentes. La manière de traiter les attributs et leurs valeurs diffère d'un rédacteur à l'autre. Par exemple, Kalofolias et al. (2019) [61] proposent une méthode qui consiste à considérer les communautés comme des sous-groupes de sommets obtenus à partir de la description de la communauté ; Pool et al. (2014) [92] présentent un modèle en deux étapes. La première consiste à construire une communauté comme un groupe de sommets, et la deuxième détermine les communautés ayant des scores élevés.

2.2.2 Mesures de centralité

Les mesures de centralité sont des mesures destinées à capturer la notion d'importance dans un graphe en déterminant les sommets ou les arêtes (arcs) les plus significatifs. Elles trouvent leurs origines dans l'analyse des réseaux sociaux. En effet, lors de l'analyse de ces réseaux, les chercheurs ont constaté que certains acteurs jouent un rôle plus « important » que d'autres.

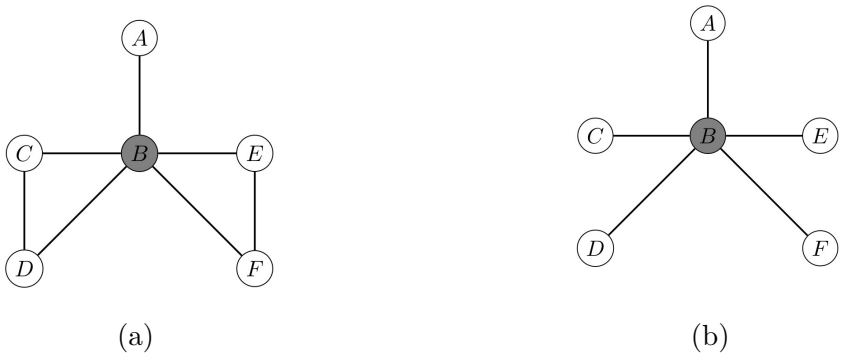


FIGURE 2.11 Illustration de la mesure de centralité. Le sommet le plus important est coloré en gris.

Par exemple, si le graphe de la Figure **2.11a** représente un réseau social, alors B a plus de contacts dans le réseau que les autres ; par contre, A est celui avec le moins de contacts. Ce phénomène se retrouve aussi dans d'autres types de graphes. Par exemple, dans les réseaux de communication, il existe des sommets qui sont primordiaux pour la communication et d'autres qui n'ont pas d'influence sur le réseau. Si l'on considère le graphe de la Figure **2.11b** comme un réseau de communication, alors B joue le rôle le plus important, car s'il tombe en panne, la communication est interrompue entre les autres éléments du réseau ; par contre, si A tombe en panne, la communication entre les sommets n'est pas affectée. Cela permet de constater que les sommets de ces réseaux n'ont pas les mêmes rôles. Pour quantifier cette notion d'importance d'un sommet dans un graphe, plusieurs définitions ont été proposées et regroupées sous le nom de mesure de centralité (Koschützki et al., 2005 [68]). En général, les mesures de centralité se présentent sous la forme d'une fonction réelle des sommets d'un graphe dont les valeurs permettent de classer des sommets suivant leur importance. La notion d'importance peut être vue de plusieurs façons différentes, par exemple, par rapport au flux de circulation dans un graphe (Borgatti, 2005 [20]) ou encore par rapport au degré de participation à la cohésion du graphe. La grande majorité des mesures de centralité sont issues des recherches en sciences sociales, dans le cadre de l'analyse des réseaux sociaux, ce qui fait que les termes sont généralement très ancrés dans les sciences sociales. En effet, il n'existe pas de mesure de centralité définie explicitement pour les graphes de processus d'écriture. Cette

revue de la littérature vise donc à chercher des mesures de centralité qui pourraient avoir du sens dans le domaine de l'analyse de processus d'écriture et à déterminer dans quelle mesure il est possible d'adapter certaines mesures existantes ou de créer de nouvelles mesures qui répondent aux critères de mesure d'importance dans le domaine de l'écriture. Bien que les mesures de centralité puissent aussi être définies pour les arêtes ou arcs des graphes, nous nous restreignons simplement ici aux mesures de centralité des sommets, car les principales informations des graphes de processus d'écriture se trouvent dans les sommets. Koschützki et al. (2005) [68] présentent un aperçu général des mesures de centralité. Les mesures de centralité sont définies selon que le graphe soit orienté ou non. Dans le cas des graphes orientés, chaque mesure de centralité est souvent définie pour chaque sommet relativement à ses arcs entrants ou sortants. Ainsi, on a une mesure de centralité entrante pour les arcs entrant à un sommet et une mesure de centralité sortante pour les arcs sortant d'un sommet. Koschützki et al. [68] classent les mesures de centralité des sommets en cinq groupes, suivant le problème d'importance que résout la mesure.

Distance et voisinages : Dans ce groupe, la centralité est mesurée par rapport à la « joignabilité » du sommet. Ainsi, un sommet est central s'il est facilement atteignable à partir des autres sommets du graphe. Les mesures de centralité de cette catégorie sont essentiellement basées sur la notion de distances au sein du graphe. Nous pouvons citer :

1. la centralité de degré [93] (l'une des plus simples des mesures de centralité), où la centralité est donnée par le nombre de voisins du sommet, c'est-à-dire les sommets à distance 1 du sommet ;
2. la centralité basée sur l'excentricité (Hage & Harary, 1995 [56]), qui est déterminée par rapport à l'inverse de la distance maximale entre le sommet considéré et les autres sommets du graphe. Ainsi, plus la distance qui sépare un sommet des autres est petite, plus le sommet est central ;
3. la centralité de proximité (Bavelas, 1950 [10]), qui est calculée par rapport à l'inverse de la somme totale des distances entre le sommet considéré et les autres sommets. Avec cette mesure, plus le total des distances est petit, c'est-à-dire que le sommet est proche des autres sommets, plus le sommet est central.

Plus court chemin : Ici, la centralité est basée sur l'ensemble des chemins les plus courts dans un graphe. Nous avons trois mesures dans cette catégorie, entre autres la centralité de contrainte, la centralité d'intermédiarité et la centralité d'intermédiarité modifiée. Les deux premières sont essentiellement basées sur le nombre de chemins les plus courts qui contiennent le sommet considéré, tandis que la troisième ne prend en compte que l'ensemble des chemins entre le sommet considéré et les autres sommets du graphe.

Vitalité : Les mesures de centralité de cette catégorie sont basées sur la notion de vitalité du sommet dans le graphe. En effet, étant donnée une fonction à valeurs réelles arbitraire sur un graphe, une mesure de vitalité quantifie la différence entre la valeur de la fonction sur un graphe avec et sans le sommet (Koschützki et al., 2005 [68]). En d'autres termes, l'importance d'un sommet est déterminée en fonction du manque à gagner que son absence peut causer au graphe. Ainsi, plus le manque est grand, plus le sommet est important. En remarquant que, dans un graphe de communication, il n'y a aucune raison que l'information passe par le plus court chemin et que d'autres chemins peuvent aussi être considérés (Stephenson & Zelen, 1989 [105]), Freeman et al. (1991) [49] introduisent trois mesures de centralité, qui sont une amélioration des trois mesures définies par rapport au plus court chemin. Ces méthodes sont chacune, en plus de la notion de vitalité, basées sur l'une des notions suivantes : le flot maximum de Ford et Fulkerson (1956) [43], l'indice de Wiener (Wiener 1947 [117]), et le nombre de plus courts chemins. Nous avons, entre autres, la proximité de vitalité et la centralité de contrainte de vitalité (Koschützki et al., 2005 [68]).

Flux de courant : Ici, le graphe est considéré comme un réseau électrique dans lequel circule un courant (Koschützki et al., 2005 [68]). En effet, le graphe est transformé en circuit électrique en supposant que sur chaque arête est placée une unité de résistance. Le premier à faire cette considération est Klein Raindić (1993) [65]. Newman (2003) [83] définit la centralité d'intermédiarité de flux de courant comme la quantité de courant qui circule à travers un sommet dans le réseau électrique. Pour la centralité de proximité de flux de courant, Bandes et Fleischer (2005) [24] proposent une mesure de centralité qui évalue la distance entre deux sommets comme la différence de potentiel entre les deux sommets.

Marche aléatoire : Dans certains cas, déterminer les chemins les plus courts entre deux sommets peut être impossible. Dans ce cas, la centralité basée sur les plus courts chemins peut être non fondée. La notion de marche aléatoire est alors introduite pour pallier à ce manque. En effet, on suppose qu'un « message » part d'un sommet source s à un sommet cible t en suivant les arêtes du graphe sans savoir où se trouve le sommet t ; ainsi, lorsqu'il arrive à un sommet, il choisit une arête au hasard pour poursuivre son chemin jusqu'à ce qu'il atteigne t. Avec cette notion, Newman (2003) [83] définit la mesure de centralité d'intermédiarité de marche aléatoire d'un sommet i comme le nombre de fois qu'une marche aléatoire passe par le sommet i durant son trajet, divisé par le nombre total de marches aléatoires de s à t. De la même manière, on définit également une forme de centralité de proximité de marche aléatoire, appelée centralité de Markov [116]. Dans cette centralité, le plus court chemin est remplacé par le temps moyen du premier passage par le sommet considéré. Ainsi, la centralité de Markov est l'inverse de la moyenne des temps moyens des premiers passages par le sommet

considéré.

Rétroaction : Dans cette catégorie, la notion de centralité d'un sommet est définie par rapport à la centralité de ses voisins. En d'autres termes, un sommet est d'autant plus central que ses voisins le sont également. L'une des premières mesures de centralité utilisant ce principe est l'indice de Katz (Katz 1953 [62]). Cette mesure est basée sur le fait que si un individu i vote pour un individu j et si l'individu i a plusieurs électeurs, alors cela revient à dire que les électeurs de i ont aussi voté indirectement pour j. L'idée de Katz est alors de compter également tous les votes indirects, où le nombre d'intermédiaires peut être arbitrairement grand. Dans le même ordre d'idées, pour les réseaux sociaux, nous avons, par exemple, la centralité des vecteurs propres de Bonacich [16], l'indice Hubbell [60] et la centralité de la négociation selon Bonacich [17]. Dans la catégorie des centralités de rétroaction, on retrouve aussi les mesures de centralité du web. Nous avons, entre autres, le PageRank (Page 1999 [88], Berkhin 2005 [14]) qui ne prend en compte que la structure topologique du graphe. Les pôles et autorités (Kleinberg 1999 [66]) qui prennent en compte à la fois la topologie et le contenu des pages web, et SALSA (Lempel 2000 [71]) qui est une combinaison des deux autres mesures de centralité.

En résumé, les mesures de centralité permettent de capturer la notion d'importance d'un sommet suivant l'information que l'on souhaite extraire du graphe. Ainsi, elles constituent un outil que l'on peut utiliser pour extraire des informations des graphes des processus d'écriture.

2.3 Lisibilité

En général, lorsqu'on parle de mesure de lisibilité d'un texte, on a l'impression qu'on fait référence à l'aspect physique du texte, c'est-à-dire la mise en page, la typographie, telle que la taille ou la couleur des polices, ou encore la présence des graphiques. Mais, cela relève d'un autre domaine que Richaudeau (1969) [47, 95] a dénommé « lisibilité typographique ». En effet, dans la langue anglaise, « lisibilité » et « lisibilité typographique » sont désignées par deux mots différents : « readability » pour « lisibilité » et « legibility » pour « lisibilité typographique ». À la lumière de cette distinction, des auteurs, tels que Louis Timbal-Duclaux [110], suggéreront l'emploi du terme « lisabilité » pour « readability » et du terme « lisibilité » pour « legibility », mais cette proposition n'a pas eu le succès escompté.

Pour Edgar Dale et Jeanne Chall (1949) [32], la lisibilité est la somme des éléments d'un texte, y compris les interactions, qui affectent la compréhension et la vitesse de lecture d'un groupe de lecteurs. Pour Henry (1975), c'est le degré de difficulté éprouvé par un lecteur essayant

de comprendre un texte, et pour Klare (1963), la facilité de compréhension ou d'assimilation due au style d'écriture (François 2011 [47], DuBay 2004 [37]). Ainsi, en prenant en compte ces éléments, la lisibilité est un indice qui renseigne sur le niveau de difficulté d'un texte en utilisant les aspects lexicaux, syntaxiques, de cohérence et de cohésion présents dans le texte. En d'autres termes, c'est un indicateur du niveau de compétence nécessaire pour être en mesure de comprendre un texte, au vu des termes qui le constituent. La lisibilité a ses origines vers la fin du XIX^e siècle aux États-unis (DuBay 2004 [37]). Elle est née d'un souci d'évaluation de textes destinés aux élèves des écoles américaines. En effet, les premières études de lisibilité sont essentiellement tournées vers la langue anglaise. Au début, pour mesurer la lisibilité d'un texte, on se basait sur le jugement d'experts ou sur des résultats obtenus sur des échantillons de la population [47], mais cette méthode était très coûteuse. De ce fait, dès le début du XX^e siècle, des recherches ont été menées pour établir des formules mathématiques qui puissent prédire la lisibilité. Le principal problème de ces études est de déterminer les aspects du texte qui influencent le niveau de compréhension et ceux qui doivent être pris en compte dans la lisibilité. L'idée de base est que, plus un texte contient des mots ignorés par le lecteur, plus le texte est difficile [13]. En 1921, Edward Thorndike établit une liste de mots anglais fréquemment utilisés [37]. Le premier indice de lisibilité, l'œuvre de Lively et Pressey (1923) [72], est simplement le décompte des mots du texte absents de la liste de Thorndike [37]. Par la suite, en 1928, Vogel et Washbourne établissent la première véritable formule obtenue par régression linéaire qui prend en compte la difficulté du vocabulaire et la syntaxe [37]. Depuis lors, les formules de lisibilité ont nettement évolué, avec l'ajout de nouvelles variables linguistiques et aussi de nouvelles méthodes utilisées pour établir ces formules.

De plus, l'utilisation de ces formules ne se limite plus simplement au cadre scolaire, mais s'étend à bien d'autres domaines.

Formules de lisibilité traditionnelles ou classiques : Traditionnellement, les formules de lisibilité sont obtenues en utilisant la régression linéaire. On obtient alors des formules simples et compréhensibles qui se présentent sous la forme :

$$L = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n \tag{2.3}$$

où L est une estimation du niveau de lisibilité du texte ; $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ les paramètres du modèle et $X_1, X_2, \ldots, X_n$ les variables linguistiques.

Les variables linguistiques sont les variables telles que : le nombre de mots, la longueur des mots, le nombre de syllabes, la longueur moyenne des phrases et le nombre de phrases.

Plusieurs formules de lisibilité sont établies en utilisant la méthode traditionnelle, mais les formules de Dale et Chall(1948a) [31], Flesch(1948) [41] et Gunning(1952) [55] sont les trois qui sortent du lot. Elles utilisent toutes trois la même variable pour mesurer la difficulté syntaxique des textes, mais la différence majeure réside dans l'évaluation de la difficulté lexicale. Dale-Chall utilise une liste de référence, Flesch se base sur la longueur des mots et Gunning sur les mots difficiles (mots comptant plus de 2 syllabes). Les autres formules de type traditionnel sont généralement des améliorations des formules précédentes, en remplaçant ou en supprimant certaines variables ou en recalculant les paramètres. Constatant que les formules classiques ne peuvent pas s'appliquer de la même manière dans tous les domaines, des recherches sont effectuées pour spécialiser les formules classiques. Ce qui donne lieu à l'apparition des formules spécialisées, telles que des formules destinées aux textes scientifiques [47], des formules pour évaluer l'oral [40, 47] et pour les matériels de l'armée [47, 104]. En raison de leur simplicité, du nombre restreint de variables et de leur facilité d'application, ces formules sont largement utilisées jusqu'à présent. Elles se retrouvent dans des logiciels de traitement de texte comme Word, dans le domaine de la santé (pour évaluer, par exemple, les notices des médicaments), ainsi que sur le web (pour évaluer les pages des sites). Toutefois, les formules classiques sont largement critiquées à cause de cette simplicité. Pour Bormuth [21], la structure de la formule classique n'est pas exacte, car les variables sont plutôt liées par des courbes et non des droites. Les formules de lisibilité classiques sont aussi critiquées sur le fait qu'elles reposent uniquement sur les facteurs textuels et du fait qu'elles ne peuvent pas être généralisées, ou même utilisées pour améliorer la qualité d'écriture d'un texte [26, 47, 100]. Plusieurs chercheurs ont décidé d'intégrer d'autres variables dans l'évaluation de la lisibilité, notamment les variables cognitives, les variables lexico-sémantiques et les variables syntaxiques.

Lisibilité avec variables cognitives : Les variables cognitives comprennent, par exemple, l'organisation du texte, la structure rhétorique, la charge inférentielle [47]. En effet, la structure rhétorique désigne la manière d'organiser ses arguments ou preuves pour convaincre, informer ou émouvoir son lecteur. La charge inférentielle, quant à elle, est l'effort mental requis pour relier les informations implicites ou sous-entendues contenues dans un texte. Ces variables sont organisées autour de deux notions : la cohérence et la cohésion. Van Dijk [114] définit la cohérence comme la propriété sémantique du texte, basée sur l'interprétation de chaque phrase, par rapport à l'interprétation d'autres phrases du texte. La cohésion est définie comme du texte représenté par des liens grammaticaux formels explicites (connecteurs de discours) et des liens lexicaux qui signalent comment les énoncés qui ont des parties de texte plus importantes sont liés entre eux. Kintch et Vipond élaborent le premier modèle de

lisibilité avec variables structuro-cognitives [47]. Ils utilisent l'approche de l'analyse propositionnelle en se basant sur la notion de cohérence définie par Van Dijk (1977) [114]. En effet, pour eux, un texte peut être vu comme une liste de propositions. Ils observent que le temps de lecture varie en fonction du nombre de propositions dans le texte. Pour Susan Kemper (1983) [63], comprendre un texte revient à organiser temporellement ou causalement une suite d'événements. Elle se base sur cela pour proposer une formule de lisibilité qui évalue la charge inférentielle d'un texte. Pour la cohésion, Bormuth utilise des variables telles que la longueur moyenne des groupes nominaux et des noms propres. Pour mesurer la cohésion lexicale, Sheehan et al (2014) [103] proposent de calculer la fréquence de répétition des mots contenus dans des phrases adjacentes. Landauer (2013) [69] propose l'analyse sémantique latente (LSA) pour mesurer la similarité des mots et des passages entre eux. En effet, LSA est une méthode statistique et linguistique qui repose sur le principe que les mots utilisés dans des contextes similaires tendent à avoir des significations similaires. Flor, Klebonav et Sheehan (2013) [77, 102] suggèrent une mesure appelée Lexical Tightness (LT), qui évalue la cohésion entre les parties du texte à l'aide du ratio des pronoms et des articles des parties du texte. Bien que les formules de lisibilité avec variables structuro-cognitives soient un peu plus complexes, des travaux tels que ceux de Tordirascu et al. [111] ont montré que les variables structuro-cognitives ne contribuent généralement pas beaucoup au pouvoir prédictif des formules qui l'utilisent, par rapport aux formules de lisibilité traditionnelle ou aux statistiques simples.

Lisibilité avec variables lexico-sémantiques : Les variables lexico-sémantiques permettent de mesurer les difficultés dues au vocabulaire. Le rapport type-token est le plus fréquemment utilisé. Il mesure le rapport entre le nombre de mots uniques et le nombre total de mots dans un texte [77].

Lisibilité avec variables syntaxiques : Les variables syntaxiques servent à mesurer la complexité grammaticale du texte. Elles peuvent être regroupées en fonction du type d'analyse grammaticale effectuée. Ainsi, pour une analyse par arbre syntaxique, on considère la hauteur moyenne de l'arbre, le nombre moyen de phrases nominales et le nombre moyen de phrases verbales. Pour l'analyse des relations grammaticales, on utilise la distance entre les ensembles de relations grammaticales. Pour la complexité des unités syntaxiques, on prend en compte la longueur d'une unité syntaxique au niveau de la phrase, ainsi que la longueur de toute structure avec un sujet et un verbe.

Lisibilité avec traitement automatique du langage : Avec l'utilisation du traitement automatique du langage, il y a une amélioration du calcul des formules de lisibilités existantes et l'avènement d'autres types de formules de lisibilité. La lisibilité avec modèle de langage en est un exemple. En effet, le modèle de langage est la prédiction d'une distribution de probabilité des mots du vocabulaire. Pour évaluer la lisibilité suivant un modèle de langage, on utilise la notion de perplexité (PPL). Pour un modèle de langage donné m, la perplexité est définie par :

$$PPL = 2^{-\frac{1}{N}\sum_{t=1}^{N}\log_2(m(w_t))} \tag{2.4}$$

où N est le nombre total de mots du vocabulaire ; w_t est le mot numéro t ; $m(w_t)$ désigne la probabilité d'occurrence du mot w_t dans le vocabulaire étudié.

Lisibilité avec approche classification : De nos jours, le problème d'évaluation de la lisibilité d'un texte n'est plus vu comme un problème de régression linéaire, mais comme un problème de classification. En d'autres termes, évaluer la difficulté d'un texte revient à lui attribuer une classe. Schwarm et Ostendorf (2005) [98] sont parmi les premiers à développer des modèles de classification pour évaluer la lisibilité des textes. Ils utilisent les machines à vecteurs de support (SVM) combinées aux mesures traditionnelles de lisibilité et d'autres modèles statistiques de langue pour produire un classificateur. Dès lors, de nombreux autres modèles de classification sont créés, avec des objectifs et des variables particulières d'un modèle à l'autre. Vajjala et Meurers [118] adaptent des mesures lexicales et syntaxiques élaborées à l'origine pour évaluer le degré d'assimilation d'une seconde langue pour créer un classificateur. Suivant les expériences effectuées, le classificateur obtenu atteint 93% de précision et surpasse les autres approches sur la classification de la lisibilité. Xia, Kochmar et Briscoe [119] utilisent, en plus des variables lexicales et syntaxiques, le modèle de langage et la variable de cohésion pour créer un classificateur SVM. D'autres travaux illustrent la possibilité de créer des modèles qui prennent en compte plusieurs types de documents ou plusieurs langues. C'est le cas de Sheehan et al. [102] qui proposent un modèle de classification en deux étapes, qui combine les caractéristiques lexicales, syntaxiques, de cohésion et de cohérence. Ils trouvent que chaque document doit être classé selon la lisibilité de son type et que l'utilisation d'un modèle de classification pour prédire la lisibilité des documents sans tenir compte du genre n'est pas faisable. Madrazo Azpiazu et Pera [6] présentent un modèle de classification de lisibilité multilingue qui prend en compte les variables traditionnelles, lexico-sémantiques, de cohésion et de cohérence. Ce modèle produit de bons résultats et illustre le fait qu'il est possible de former un modèle prenant en compte plusieurs langues. Avec les résultats satisfaisants des méthodes de classification, certains chercheurs approfondissent les recherches en utilisant la classification neuronale. C'est le cas de Yuxuan et al.(2020) [108]

qui proposent une méthode à base de réseau de neurones récurrent, permettant de créer des modèles d'évaluation de lisibilité d'un texte. Le modèle obtenu peut prédire la lisibilité de tout genre de texte ; de plus, il permet d'évaluer la lisibilité des phrases et des paragraphes. Dans le même ordre d'idée, Nadeem et Ostendorf [81] suggèrent une approche neuronale pour remédier au fait que les lisibilités traditionnelles ne produisent pas de bons résultats sur les textes scientifiques. Ils examinent alors plusieurs architectures de réseaux de neurones, et le résultat montre qu'il est possible d'obtenir une grande précision sur de petits textes et que certains modèles peuvent être utilisés pour évaluer des textes d'autres domaines que leur domaine initial. D'autre part, pour résoudre le problème de l'utilisation d'une même formule de lisibilité pour plusieurs langues, Azpiazu et Pera (2019) [7] présentent une stratégie d'évaluation automatique de la lisibilité, Vec2Read. Vec2Read est basé sur un réseau de neurones récurrent qui permet de prédire le niveau de difficulté d'un texte indépendamment de la langue. Les méthodes neuronales ont tendance à éliminer l'utilisation des variables linguistiques utilisées dans les modèles traditionnels de lisibilité, Deutsch et al (2020) [34] proposent un modèle qui combine à la fois les caractéristiques linguistiques et les modèles d'apprentissage profond. Ils constatent que l'ajout des variables linguistiques n'a pas amélioré le modèle. Martinc, Pollak et Robnik-Skonja(2020) [77] présentent un ensemble de nouvelles approches supervisées et non supervisées pour déterminer la lisibilité des documents à l'aide de réseaux de neurones profonds. Ils démontrent que le modèle de langage neuronal peut être utilisé dans une classification non supervisée.

Après près d'un siècle d'existence, l'évaluation de la lisibilité a nettement évolué. Elle est passée d'une simple liste de mots à des modèles permettant de classer des textes dans différentes langues et dans des domaines variés, en tenant compte de plusieurs paramètres linguistiques. Bien que les méthodes aient changé, certains éléments restent toujours à considérer. Par exemple, certaines méthodes de classification ont montré qu'il est possible de généraliser les formules de lisibilité à plusieurs langues. Une avenue possible est d'établir des modèles pouvant évaluer la lisibilité, quelles que soit la langue et la nature du document. Nous avons aussi constaté que la lisibilité est utilisée par certains rédacteurs pour améliorer la qualité de leur texte, ce qui permet de remarquer qu'en évaluant un texte, on évalue aussi, sans s'en rendre compte, la manière d'écrire de la personne qui l'a rédigé. Ainsi, on peut en conclure qu'il est possible d'adapter la lisibilité pour qu'elle évalue aussi les processus d'écriture.

2.4 Conclusion

En résumé, la revue de la littérature a permis, tout d'abord, de s'imprégner des techniques de visualisation et de représentation du processus d'écriture, ainsi que d'observer leur évolution

au fil du temps. Nous avons ainsi constaté que le graphe du *modèle sans perte* représente fidèlement le processus d'écriture et que toutes les informations du processus sont contenues dans sa structure. Il ressort également que le *modèle sans perte* n'a jamais fait l'objet d'une analyse structurelle (des liens entre ses sommets). Ensuite, la revue de la littérature a permis de passer en revue toutes les méthodes d'analyse de graphes susceptibles d'être utilisées dans le graphe du processus d'écriture, notamment les méthodes de détection de communautés et les méthodes de détection des sommets les plus importants du graphe. Nous avons remarqué que chacune des méthodes a été créée pour répondre à des problèmes et dans des contextes bien précis. Ainsi, les appliquer directement aux graphes de processus d'écriture, sans modification préalable, peut produire des résultats inexplicables. Enfin, la revue de la littérature a révélé que les mesures de lisibilité servent à évaluer la qualité d'un texte et qu'elles sont devenues maintenant des modèles de classification de texte ; dans certains cas, elles aident à l'amélioration de la rédaction des textes. Forts de ces constats, nous tenterons de proposer des outils qui permettent d'analyser le graphe de processus d'écriture en tenant compte de la structure du graphe et, parfois aussi, du texte écrit.

CHAPITRE 3 OUTILS ET MÉTHODES

Dans ce chapitre, nous présentons les différents outils et méthodes que nous utiliserons pour définir les notions abordées dans les chapitres suivants.

Dans la section 3.1, il sera question de définir la notion de graphe ainsi que les différentes notions qui y sont liées. Ensuite, la section 3.2 présente la construction du *modèle sans perte*, qui sera utilisé pour définir les communautés et les indices servant à l'analyse des processus d'écriture. La section 3.3, quant à elle, présente les méthodes de détection des communautés ainsi que certains éléments permettant de comparer ces méthodes. Et enfin, la section 3.4 met en évidence les méthodes de mesure des degrés d'importance des sommets et les outils qui permettent leur comparaison.

3.1 Théorie des graphes

Dans cette section, nous définissons les termes et propriétés relatifs aux graphes qui seront utilisés dans les autres chapitres. Nous utilisons principalement les définitions et propriétés tirées de Fournier [46] et de Bondy Murty [19] [18].

Comme mentionné précédemment, un graphe est une représentation mathématique dans laquelle des points ou sommets sont reliés par des lignes. Ces diagrammes permettent de modéliser plusieurs problèmes suivant la signification donnée aux sommets et aux arêtes. Par exemple, dans les graphes sociaux, les sommets peuvent représenter des personnes ; et les lignes, des paires d'amis ; dans les graphes de communication, les sommets peuvent être des centres de communication, les lignes, les liens entre les centres. En général, dans les graphes, on s'intéresse principalement à l'existence ou non d'un lien entre les sommets.

Définition 3.1.1. Un graphe G est un couple (V, E) composé d'un ensemble V de sommets et d'un ensemble $E \subseteq V \times V$ d'arêtes.

Dans le cas où le graphe a une orientation, il est souvent appelé digraphe et noté (V, A) où A est l'ensemble des arcs.

En général, les arêtes entre deux sommets sont notées par les crochets $[x, y]$ et les arcs par les parenthèses (x, y).

Quelques définitions et propriétés des graphes :

- Deux sommets sont dits `adjacents` ou `voisins` s'ils sont reliés par une arête ou un arc. Lorsqu'on a un arc (x, y), le sommet y est appelé `successeur` de x et le sommet x est

(a) Cas non orienté (b) Cas orienté

FIGURE 3.1 Exemple de graphe

appelé `prédécesseur` de y.

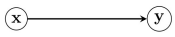
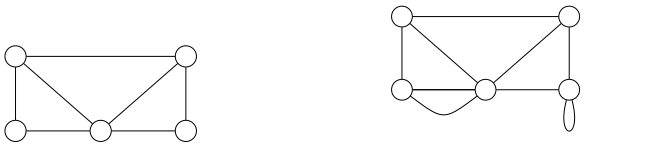


FIGURE 3.2 Arc

- Un graphe est dit `simple` s'il n'a ni boucle ni arêtes parallèles.



(a) Simple (b) Boucle et arête parallèle

FIGURE 3.3 Graphe.

Un graphe orienté est dit `strict` s'il n'a pas de boucles et pas d'arcs multiples (mais il peut avoir des arcs opposés).

- Un graphe est dit `complet` s'il est simple et chaque paire de sommets est adjacente.
- Un graphe `vide` est un graphe dans lequel il n'y a pas de sommets adjacents.
- Le `degré` de v est le nombre de sommets voisins à v.

$$deg(v) = |\{[u, v] \in E | u \in V\}|.$$

Lorsque le graphe est orienté, on a deux notions de degré :

Le degré `entrant` en v qui est le nombre d'arcs entrant dans v

$$d^{\text{in}}(v) = |\{(u, v) \in A | u \in V\}|.$$

Le degré `sortant` de v est le nombre d'arcs sortant de v

$$d^{\text{out}}(v) = |\{(v,u) \in A | u \in V\}|.$$

- Une `chaîne` dans un graphe $G = (V, E)$ est une suite $[x_0, e_1, x_1, e_2, x_2, \ldots, e_k, x_k]$ où $k \geqslant 0$, les $x_i (i = 0, 1, \ldots, k)$ sont des sommets de G et les $e_i (i = 1, 2, \ldots, k)$ sont des arêtes de G telles que pour $i = 0, 1, \ldots, k-1$, l'arête e_{i+1} relie les sommets x_i et x_{i+1}. L'entier k représente la longueur de la chaine qui correspond au nombre d'arêtes de la chaîne. Dans le cas où G est un graphe simple, on peut définir une chaine simplement par la séquence de ses sommets $[x_0, x_2, \ldots, x_k]$.

  On a un `cycle` lorsque les deux extrémités de la chaine coïncident. (C'est-à-dire $x_0 = x_k$).
- Un `chemin` dans un graphe orienté $G = (V, A)$ est une suite $(x_0, a_1, x_1, a_2, x_2, \ldots, a_k, x_k)$ où $k \geqslant 0$, les $x_i (i = 0, 1, \ldots, k)$ sont des sommets de G et les $a_i (i = 1, 2, \ldots, k)$ sont des arcs de G telles que pour $i = 0, 1, \ldots, k-1$, l'arc a_{i+1} relie les sommets x_i et x_{i+1}. L'entier k représente la longueur du chemin qui correspond au nombre d'arcs du chemin. Un chemin de longueur k est appelé un $k - chemin$. Dans le cas où G est un graphe simple, on peut définir un chemin simplement par la séquence de ses sommets $(x_0, x_1, \ldots, x_k)$.

  On a un `circuit` lorsque les deux extrémités du chemin sont identiques. (C'est-à-dire $x_0 = x_k$).
- Un chemin est dit `élémentaire` s'il ne passe pas deux fois par le même sommet. (Fournier [46])
- Un chemin est dit `hamiltonien` s'il passe par tous les sommets du graphe une et une seule fois.
- Un graphe est dit `hamiltonien` s'il possède un cycle hamiltonien.
- Un graphe est dit `connexe` s'il existe un chemin entre toutes les paires de sommets. Dans le cas contraire, le graphe est `non connexe`.
- La `distance` entre deux sommets dans un graphe orienté (non orienté) est la longueur du chemin (de la chaîne) le plus court qui les relie, s'il existe au moins un chemin (une chaîne) entre eux. Cette distance est le nombre minimal d'arcs (arêtes) dans un chemin (une chaîne) reliant les deux sommets.

Définition 3.1.2. Un graphe attribué est un graphe G dans lequel chaque sommet $v \in V$ est associé à un vecteur d'attributs $x = (x_1, x_2, \ldots, x_d)$ et chaque arête $e \in E$ a un vecteur $y = (y_1, y_2, \ldots, y_t)$.

Ce sont les notions sur les graphes qui sont utilisées dans le reste des chapitres. Certaines autres notions seront définies lors de leur utilisation.

3.2 Modèle sans perte

Dans cette section, nous présentons le *modèle sans perte* (Bécotte-Boutin 2019 [11]) : sa construction et un exemple.

Le *modèle sans perte* est un modèle qui permet de transformer un fichier log de processus d'écriture en un graphe. Le modèle est dit sans perte car, à partir du graphe, on peut retrouver le fichier log qui a permis sa construction. Chaque ligne du fichier log est représentée par un sommet dans le graphe. Le graphe du *modèle sans perte* est un graphe attribué. Chaque sommet du graphe comporte trois attributs (Bécotte-Boutin 2019 [11]) : un numéro, un type et une position.

Numéro : le numéro du sommet représente le numéro de la ligne du fichier auquel le sommet correspond.

Type : le type est soit une insertion, soit une suppression. Pour un effet visuel, les sommets de même type sont représentés par une même couleur. Ainsi, dans le *modèle sans perte*, la couleur rouge est utilisée pour les insertions et le bleu pour les suppressions (Bécotte-Boutin 2019 [11]). En plus, la couleur jaune est utilisée pour les sommets des insertions supprimées. En effet, la couleur du sommet passe du rouge au jaune lorsque le caractère du sommet est supprimé dans le texte.

Position : la position donne l'emplacement du caractère représenté par le sommet dans le texte. L'attribut position d'un sommet est mis à jour chaque fois qu'un sommet est ajouté au graphe. L'attribut position est -1 si le sommet représente une suppression ou un caractère supprimé. Par contre, si c'est une insertion, alors l'attribut position est un entier supérieur ou égal à 0.

En plus de ces trois attributs principaux des sommets, deux attributs complémentaires sont automatiquement assignés à chaque sommet lors de sa création, notamment le temps de frappe et le caractère que représente le sommet [11].

La position du curseur au moment de l'exécution de la ligne i du fichier log est donnée par $Pos(i)$.

Principe de construction

Soit λ le nombre de lignes dans le fichier log.

Première étape: On commence par créer un graphe à deux sommets. Un sommet -1 qui représente le début du texte et un sommet 0 pour la fin du texte.

Deuxième étape: Ensuite, pour chaque ligne i du fichier log représentant une frappe, on crée un sommet avec les attributs $type(i)$, $temps(i)$ et $c(i)$ qui sont respectivement le type de frappe (insertion ou suppression), le temps de l'insertion et le caractère correspondant à la frappe. Pour chaque sommet x, $p_i(x)$ est la position du caractère représenté par le sommet x dans le texte après avoir créé le sommet de la ligne i du fichier log. Lorsque le sommet x est une suppression, puisqu'il ne représente aucun caractère dans le texte, alors on fixe $p_i(x) = -1$.

- Lorsque le sommet i est une insertion, le sommet i est relié aux sommets j et k qui correspondent aux caractères à la position avant le caractère dans le texte ($p_{i-1}(j) = p_i(i) - 1$) et après lui dans le texte ($p_{i-1}(k) = p_i(i)$). Et puis tous les sommets avec attribut à position supérieure ou égale à $p_i(i)$ sont mis à jour en ajoutant $+1$ à leurs attributs.
- Lorsque le sommet i est une suppression, le sommet i est relié aux sommets j et k correspondant aux caractères à la position avant le caractère supprimé dans le texte ($p_{i-1}(j) = Pos(i) - 1$) et après ce caractère ($p_{i-1}(k) = Pos(i) + 1$). Et puis tous les sommets avec attribut de position supérieure ou égal à $p_i(i)$ sont mis à jour en ajoutant -1 à leur attribut. Par contre, pour le sommet du caractère supprimé, l'attribut est fixé à -1 et ne changera plus, car il ne sera plus jamais mis à jour.

L'**Algorithme 1** présente la méthode formelle de construction du modèle :

Notations :

$Pos(i)$:	attribut position de la ligne i du fichier log qui donne la position du curseur au moment de la création de la ligne i
$p_i(x)$:	la valeur de l'attribut position du sommet x à la fin de l'ajout du sommet i dans le graphe.
$Type(x)$:	nature du sommet x (insertion (I), suppression (S) ou insertion supprimée (IS))
λ :	nombre de lignes du fichier log

Algorithme 1 : Modèle sans perte

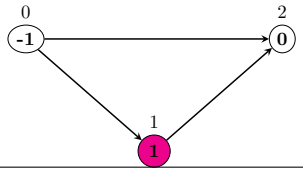
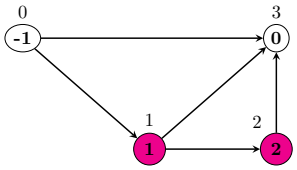
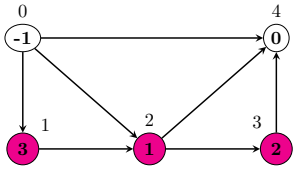
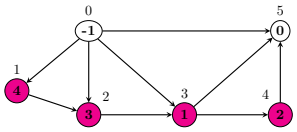
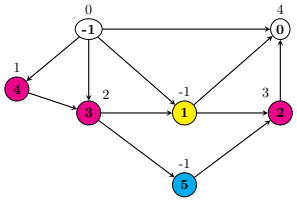
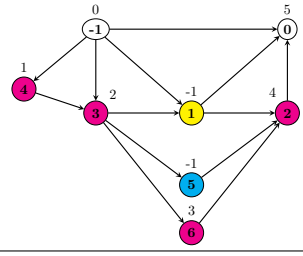
Entrées : un fichier log
Sorties : un graphe $G = (N, A)$ représentant un processus d'écriture
Créer un graphe G qui contient les sommets -1 et 0 ; et l'arc $(-1, 0)$;
Poser $p_0(-1) = 0$ et $p_0(0) = 1$;
pour *toute ligne du fichier* $i = 1$ *jusqu'à* λ **faire**
  Trouver le sommet j tel que $p_{i-1}(j) = Pos(i) - 1$;
  Trouver le sommet k tel que $p_{i-1}(k) = Pos(i)$;
  si *Type(i)* $=$ `I` **alors**
    Poser $p_i(i) = Pos(i)$;
    pour *tous les sommets* x *dans le graphe* **faire**
      si $p_{i-1}(x) \geqslant Pos(i)$ **alors**
        Poser $p_i(x) = p_{i-1}(x) + 1$
      sinon
        Poser $p_i(x) = p_{i-1}(x)$
      fin
    fin
  fin
  Créer le sommet i et l'ajouter dans G
  si *Type(i)* $= S$ **alors**
    Poser $p_i(i) = -1$;
    pour *tous les sommets* $x \neq i$ *dans le graphe* **faire**
      si $p_{i-1}(x) > Pos(i)$ **alors**
        si $p_{i-1}(x) = Pos(i) + 1$ **alors**
          Poser $p_i(x) = -1$ et $k = x$
        sinon
          Poser $p_i(x) = p_{i-1}(x) - 1$
        fin
      sinon
        Poser $p_i(x) = p_{i-1}(x)$
      fin
    fin
  fin
  Ajouter les arcs (j, i) et (i, k)
fin
retourner G

Exemple 3.2.1. *Construction du graphe sans perte.*

Considérons le fichier log suivant :

i	*Temps*	*Type(i)*	*Position*	*Caractère*
1	*1*	*I*	*1*	*F*
2	*20*	*I*	*2*	*E*
3	*56*	*I*	*1*	*A*
4	*100*	*I*	*1*	*C*
5	*150*	*S*	*3*	*-*
6	*170*	*I*	*3*	*K*

on a alors

i	$Pos(i)$	j	k	$Type(i)$	$p_i(x)$	*Graphes*
1	1	-1	0	I	$p_1(-1) = 0$ $p_1(0) = 2$	
2	2	1	0	I	$p_2(-1) = 0$ $p_2(1) = 1$ $p_2(0) = 3$	
3	1	-1	1	I	$p_3(-1) = 0$ $p_3(1) = 2$ $p_3(2) = 3$ $p_3(0) = 4$	
4	1	-1	3	I	$p_4(-1) = 0$ $p_4(1) = 3$ $p_4(2) = 4$ $p_4(3) = 2$ $p_4(0) = 5$	
5	3	3	2	S	$p_5(-1) = 0$ $p_5(1) = -1$ $p_5(2) = 3$ $p_5(3) = 2$ $p_5(4) = 1$ $p_5(0) = 5$	
6	3	3	2	I	$p_6(-1) = 0$ $p_6(1) = -1$ $p_6(2) = 3$ $p_6(3) = 2$ $p_6(4) = 1$ $p_6(5) = -1$ $p_6(0) = 6$	

Pour plus de détails, consulter [11]. En résumé, le *modèle sans perte* permet de visualiser le processus d'écriture à partir du fichier log. En effet, lorsque l'on écrit, chaque caractère inséré ou supprimé est représenté par un sommet qu'on met en lien avec les arcs suivant leurs positions dans le texte.

Dans la suite, nous essaierons de montrer, à travers la notion de communauté et d'indice de

modification, que le *modèle sans perte* peut aussi être utilisé pour une analyse plus détaillée du processus d'écriture. Pour une harmonisation des couleurs dans le reste des chapitres, nous modifions un peu les couleurs en adoptant les couleurs suivantes : `Bleu` pour les insertions, `Rouge` pour les insertions supprimées et `Jaune` pour les suppressions. Ainsi, la visualisation que nous proposons au chapitre 4 et le graphe utiliseront des codes couleurs quasiment identiques.

3.3 Détection des communautés

Dans cette section, nous présentons quelques méthodes de détection de communauté qui seront utilisées au chapitre **4** pour une comparaison avec les communautés de notre méthode. Les méthodes que nous présentons sont essentiellement les méthodes implémentées dans le logiciel Python. Nous donnons également des outils d'évaluation des méthodes de détection. (Nous utilisons les définitions tirées de Fortunato (2010) [44], de Brandes (2005) [23] et de Newman (2010) [87])

3.3.1 Méthode de Clauset-Newman-Moore (MCNM)

Elle est la première méthode conçue pour maximiser la modularité [44]. Dans la méthode de Clauset-Newman-Moore, les communautés sont définies comme des ensembles de sommets qui sont plus connectés entre eux qu'avec le reste du réseau. En effet, il s'agit d'une méthode gloutonne, basée sur un regroupement hiérarchique ascendante dans laquelle des groupes de sommets sont successivement réunis pour former des communautés plus larges, de sorte que la modularité augmente après la fusion. On part avec une partition où chaque sommet est seul dans sa communauté, les arêtes ne sont pas initialement présentes, elles sont ajoutées une à une au cours de la procédure. On choisit à chaque étape, l'arête qui entraine la plus grande augmentation (ou la plus petite diminution) de la modularité. Pour rappel, la modularité est donnée par la formule **2.1** [44, 85].

3.3.2 Méthode de la circulation des fluides (MCF)

Dans cette méthode, la définition des communautés est basée sur l'interaction des fluides dans un environnement [90]. En effet, les communautés sont considérées comme des fluides qui s'étendent et se contractent en fonction des relations qu'elles ont avec les autres communautés au sein du graphe. Le nombre de communautés (k) à obtenir est fixé à l'avance, ce qui permet à la méthode d'identifier un nombre variable de communautés. L'algorithme de la méthode de la circulation des fluides est basé sur la propagation des étiquettes [90]. L'algorithme

débute alors en assignant de manière aléatoire un sommet à chacune des k communautés. Ensuite, l'algorithme itère sur tous les sommets dans un ordre aléatoire, en mettant à jour la communauté de chaque sommet sur la base de sa propre communauté et des communautés de ses voisins (sommets adjacents). La règle de mise à jour est basée sur la densité de la communauté, qui est l'inverse du nombre de sommets de la communauté. Ainsi, le sommet est assigné à la communauté ayant la somme des densités maximale au sein du voisinage du sommet v en tenant aussi compte de sa communauté. Ce processus est effectué plusieurs fois jusqu'à ce qu'aucun sommet ne change de communauté lors d'une itération complète. La méthode est asynchrone, car chaque mise à jour de sommet est calculée en utilisant le dernier état partiel du graphe [90].

3.3.3 Méthode des k-cliques (Mkcliques)

Dans cette méthode, la définition de la communauté est basée sur le principe qu'une communauté type se compose de plusieurs sous-graphes complets (c'est-à-dire deux sommets quelconques sont toujours adjacents) qui ont des sommets en commun. [89]. Une communauté, ou plus précisément une communauté de k-cliques, est comme une union de toutes les k-cliques (sous-graphes complets de tailles k) qui peuvent être atteintes les unes à partir des autres via une série de k-cliques. Autrement dit, une communauté de k-cliques est l'union de toutes les cliques de tailles k qui partagent $k-1$ sommets [89].

3.3.4 Propagation d'étiquettes (MPL ou LPA)

La propagation d'étiquettes permet de trouver des communautés en utilisant uniquement la structure du graphe. Deux variantes de la méthode sont programmées dans Python.

La méthode asynchrone (MPLa) L'algorithme débute en attribuant une unique étiquette à chaque sommet, puis, met à jour l'étiquette d'un sommet en donnant à ce sommet l'étiquette qui apparait le plus dans son voisinage. L'opération est répétée jusqu'à ce que tous les sommets possèdent l'étiquette le plus présent dans son voisinage. L'algorithme est asynchrone, car chaque sommet est mis à jour sans attendre l'actualisation des étiquettes des autres sommets.

La méthode semi-synchrone (MPLs) : Ici, chaque sommet est mis à jour en tenant compte des changements d'étiquettes effectués sur les autres sommets lors de l'itération. La propagation d'étiquettes est complète lorsque tous les sommets ont l'étiquette le plus fréquent dans son voisinage.

3.3.5 Outils d'évaluation des méthodes de détections des communautés

Dans cette partie, nous donnons les outils pour évaluer les communautés obtenues à partir des différentes méthodes ainsi que des techniques pour comparer ces méthodes. En général, une fonction de qualité est utilisée pour évaluer les différentes partitions en communautés obtenues. Une fonction de qualité est une fonction qui attribue un score à chaque partition d'un graphe. Ainsi, la partition avec le plus grand score est considérée comme la meilleure. La fonction de qualité se présente parfois sous une forme additive [44] :

$$Q_{\text{lte}}(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} q(\mathcal{C}) \tag{3.1}$$

où $\mathcal{C}$ est une communauté de la partition $\mathcal{P}$ et q une fonction qui évalue chaque communauté en attribuant un score selon le degré de conformité aux critères de construction des communautés.

Dans notre cas, nous utilisons principalement la performance, la couverture et la modularité.

La performance : La performance est une mesure de qualité qui compte le nombre de paires de sommets correctement interprétées [44], c'est à dire le nombre de paires de sommets connectés par une arête et appartenant à la même communauté plus le nombre de paires de sommets non connectés par une arête et appartenant à deux communautés différentes. Pour une partition $\mathcal{P}$, la performance est définie par :

$$P_{\text{erf}}(\mathcal{P}) = \frac{|\{(i,j) \in E, \mathsf{C}_i = \mathsf{C}_j\}| + |\{(i,j) \notin E, \mathsf{C}_i \neq \mathsf{C}_j\}|}{n(n-1)/2} \tag{3.2}$$

$0 \leqslant P_{\text{erf}}(\mathcal{P}) \leqslant 1$.

où

- n est le nombre de sommets du graphe.
- E l'ensemble des arêtes du graphe.
- C_i est la communauté du sommet i.

L'inconvénient principal de la performance est la gestion de graphes très clairsemés, c'est-à-dire avec peu d'arêtes. En effet, dans ces cas, l'écart entre le nombre d'arêtes réalisables et le nombre maximal d'arêtes est également énorme, et la performance a tendance à être plus petite. En général, les méthodes avec de bonnes performances ont tendance à générer de nombreuses communautés [23].

La couverture : La couverture est une mesure de qualité qui est basée sur le nombre d'arêtes entre les sommets d'une même communauté [44]. En effet, la couverture est le rapport du nombre d'arêtes intra-communautaires au nombre total d'arêtes. Pour une partition $\mathcal{P}$, la couverture est définie par :

$$C_{over}(\mathcal{P}) = \sum_{C \in \mathcal{P}} \frac{l_C}{e} \tag{3.3}$$

où l_C est le nombre d'arêtes à l'intérieure de la communauté C et e le nombre total d'arêtes ; la couverture est égale à 1 s'il y a une seule communauté dans la partition. Par conséquent, la couverture est utilisée lorsqu'on est certain qu'on n'a pas des partitions comportant qu'une seule communauté. Dans ce cas, la meilleure partition est celle qui maximise la couverture. C'est pour cette raison que la couverture n'est généralement pas utilisée comme seule mesure de qualité, mais qu'on l'associe à d'autres mesures de qualité pour une bonne interprétation [23].

La modularité : La modularité de Newman et Girvan (2004) est la fonction de qualité la plus utilisée. Elle est basée sur l'idée qu'un graphe aléatoire n'est pas censé avoir une structure de communauté. L'existence possible de communautés est alors révélée par comparaison entre la densité réelle des arêtes dans un sous-graphe et la densité que l'on s'attendait à avoir dans le sous-graphe si les sommets du graphe étaient reliés indépendamment de la structure communautaire. Cette densité d'arêtes attendue dépend du modèle sans structure de communauté choisi. Le modèle sans structure de communauté est appelé modèle nul ; en effet, c'est une copie du graphe original conservant certaines de ses propriétés structurelles [44] telles que :

— Le même nombre d'arêtes que le graphe initial.
— La même distribution des degrés que le graphe initial.

La modularité peut alors être écrite comme suit :

$$Q = \frac{1}{2e} \sum_{i=1}^{n} \sum_{j=1}^{n} (B_{ij} - P_{ij})\, \delta(\mathsf{C}_i, \mathsf{C}_j) \tag{3.4}$$

où la somme porte sur toutes les paires de sommets, n le nombre de sommets, $B = (B_{ij})_{1 \leqslant i,j \leqslant n}$ est la matrice d'adjacence, e le nombre total d'arêtes du graphe, et P_{ij} représente la probabilité d'avoir une arête entre les sommets i et j dans le modèle nul. La fonction de Kronecker $\delta(\mathsf{C}_i, \mathsf{C}_j)$ vaut 1 si les sommets i et j sont dans la même communauté et 0 sinon.

Le graphe du modèle nul est choisi de manière arbitraire. Il existe plusieurs méthodes possibles, mais il est préférable d'opter pour un modèle nul avec la même distribution de degrés

que le graphe d'origine [44]. On obtient alors la formule de modularité comme définie à la sous-section 3.3.1.

3.4 Mesure d'importance d'un sommet dans un graphe

Dans cette section, nous présentons les mesures de centralité qui ont été considérées à la base de nos réflexions et qui seront utilisées au chapitre **6** pour une comparaison avec les méthodes que nous aurons élaborées. Il sera aussi question de présenter quelques outils de comparaison des méthodes.

3.4.1 Quelques mesures de centralité

`Centralité de degré (CD) :` La centralité de degré, Freeman (1979) [48], représente la forme la plus simple et la plus intuitive de la notion de centralité. Selon cette mesure, déterminer l'importance d'un sommet dans un graphe revient donc à calculer le nombre de ses sommets voisins, ou de manière équivalente, à calculer le nombre de liens qui lui sont incidents. En théorie de graphe, ce nombre est appelé `degré` du sommet, d'où l'appellation de `centralité de degré`.

`Centralité de Katz (CK) :` La centralité de Katz, présentée par Leo Katz [62] en 1953, est l'une des premières idées concernant les centralités de rétroactions. Pour rappel, les centralités de rétroaction (parfois appelées centralité du vecteur propre) sont des centralités basées sur le principe qu'un sommet est d'autant plus important que ses voisins le sont. La centralité de Katz calcule la centralité d'un sommet en fonction de la centralité de ses voisins. En effet, elle calcule l'influence relative d'un sommet au sein du réseau en mesurant le nombre de voisins immédiats (sommets du premier degré) ainsi que tous les autres sommets du réseau qui sont connectés au sommet considéré via ses voisins immédiats. La centralité de Katz pour un sommet i, en fonction de celle de ces voisins, est donnée par la formule :

$$CK(i) = \alpha \sum_{j=1, j\neq i}^{n} B_{ij} CK(j) + \beta_i \tag{3.5}$$

où

- n est le nombre de sommets du graphe.
- $B = (B_{ij})_{1\leqslant i\leqslant n, 1\leqslant j\leqslant n}$ est la matrice d'adjacence du graphe avec λ_{max} pour valeur propre maximale.
- Le paramètre β_i contrôle la centralité initiale, ce qui permet d'accorder un poids

supplémentaire aux voisins immédiats du sommet i.

- Le paramètre $\alpha < \frac{1}{\lambda_{max}}$ est un facteur d'atténuation qui pénalise les connexions établies avec des voisins éloignés, ce qui permet que la centralité de Katz soit calculée correctement.

`Centralité d'intermédiarité (CI) :` La centralité d'intermédiarité mesure à quel point un sommet se trouve sur des chemins entre d'autres sommets. L'idée d'intermédiarité a été introduite par Freeman en 1977 [48] et quelques années avant par Anthonisse [4]. En effet, si, dans un graphe de communication, on a par exemple des messages qui circulent de sommet en sommet le long des arêtes, et si chaque paire de sommets du graphe, qui est connectée par un chemin, échange un message avec une probabilité égale par unité de temps, et que, de plus, les messages empruntent toujours le chemin le plus court à travers le graphe (ou un tel chemin, choisi au hasard, s'il y en a plusieurs), alors le nombre de messages passant par chaque sommet est proportionnel au nombre de plus courts chemins sur lesquels le sommet se trouve. C'est ce nombre qui est appelé la centralité d'intermédiarité. Plus formellement :

Définition 3.4.1. La centralité d'intermédiarité d'un sommet v est la somme de la fraction de tous les chemins les plus courts de toutes les paires qui passent à travers v :

$$CI(v) = \sum_{s,t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{3.6}$$

où V est l'ensemble des sommets, σ_{st} est le nombre de chemins les plus courts de s à t, $\sigma_{st}(v)$ est le nombre de ces chemins passant par un sommet v autre que s, t.

Si $s = t$, $\sigma_{st} = 1$, et si $v \in \{s, t\}$, $\sigma_{st}(v) = 0$.

`Centralité de proximité (CP) :` La centralité de proximité mesure à quel point un sommet est proche des autres sommets du graphe [23]. Plus formellement,

Définition 3.4.2. La centralité de proximité d'un sommet v est l'inverse de la distance moyenne du chemin le plus court vers v sur tous les $n-1$ sommets atteignables :

$$CP(v) = \frac{n-1}{\sum_{u \in V} d(v,u)} \tag{3.7}$$

où $d(v,u)$ est la distance du chemin le plus court entre v et u, et $n-1$ est le nombre de sommets accessibles à partir de v. Notons que la fonction de distance de proximité calcule la distance entrante vers u pour les graphes orientés.

Bien que la version non orientée de la centralité de proximité soit applicable aux graphes connexes orientés, Valente et Foreman [113] proposent une version orientée de la mesure. Cette variante est définie par :

$$CP_o(v) = \frac{\sum_{u \in V}(\Delta_G + 1 - d(v,u))}{n-1} \tag{3.8}$$

où V est l'ensemble des sommets du graphe, Δ_G est le diamètre du graphe et n le nombre de sommets.

`PageRank (PgR)` : PageRank calcule un classement des sommets d'un graphe en fonction de la structure des liens entrants des sommets. Il a été conçu à l'origine par Brin et Page [25] comme algorithme pour classer les pages web. Au lieu de tenir simplement compte de la centralité de pages qui pointent vers une page pour calculer la centralité de la page considérée, comme le fait Katz [62], ils tiennent aussi compte du nombre de pages qui pointent vers la page. Ainsi, le PageRange est une mesure de centralité d'une page web dépendant du nombre et de la centralité des pages web qui renvoient à cette page. Nous avons la définition suivante :

Définition 3.4.3 (PageRank simplifié)**.** Soit u une page web. Soit F_u l'ensemble des pages où u pointe. B_u l'ensemble des pages qui pointe vers u. Soit $N_u = |F_u|$ le nombre de liens sortant de u. Soit $c < 1$ le facteur utilisé pour la normalisation (pour que le total des rangs de toutes les pages soit constant). La version simplifiée de PageRank pour le sommet u, en fonction des pages qui pointent vers lui, est donnée par la formule suivante :

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} \tag{3.9}$$

L'équation est récursive, mais elle peut être calculée en commençant par n'importe quel ensemble de rangs et en itérant le calcul jusqu'à ce qu'il converge [88].

Lorsque deux pages pointent l'une vers l'autre, mais qu'aucune autre page ne pointe vers l'une d'elles, alors, au cours de l'itération, cette boucle accumulera le rang, mais ne distribuera jamais de rang (puisqu'il n'y a pas d'arêtes sortantes). La boucle forme une sorte de piège que nous appelons un puits de rang. Pour surmonter ce problème de puits de rang, Brin et Page [25] introduisent une source de rang c'est-à-dire une page qui pointe vers d'autres sites mais qu'aucune page ne pointe vers elle, ce qui donne la définition suivante :

Définition 3.4.4 (PageRank)**.** Soit $E(u)$ un vecteur sur les pages web qui correspond à une source de rang. Alors, le PageRank d'un ensemble de pages web est une affectation,

R', aux pages web qui satisfait :

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u) \tag{3.10}$$

tel que c soit maximisé et $\|R'\|_1 = 1$ ($\|R'\|_1$ dénote la norme L_1 de R') [88].

3.4.2 Comparaison des mesures de centralité

Les mesures de centralité sont généralement nées dans le but de résoudre un problème précis dans le graphe. Ainsi, les mesures de centralité peuvent être divisées en deux groupes, d'une part, celles qui sont basées sur la cohésion du graphe, et d'autre part, celles qui sont basées sur le flux de circulation. Suivant ces bases, il existe différentes méthodes pour évaluer la performance de ces mesures de centralité. Pour la cohésion du graphe, on peut citer, par exemple, la sensibilité [76] et la robustesse [33]. Pour la circulation, on a la notion d'efficacité [115].

Sensibilité : La sensibilité (S_e) permet de connaître le nombre de changements de connectivité dans un graphe lors de la suppression des sommets. La sensibilité est définie par :

$$S_e = \sum_{k \in \sigma} \frac{n_k k^2}{N} \tag{3.11}$$

où n_k représente le nombre de composantes connexes dont la taille est égale à k, N représente le nombre de sommets du graphe et σ l'ensemble des tailles des différentes composantes connexes du graphe.

La suppression progressive des sommets du réseau fragmente le graphe en plusieurs parties déconnectées. Lors du processus de suppression, les valeurs de la sensibilité, s, présentent généralement un pic correspondant à une proportion spécifique, ps, à laquelle le graphe n'est plus connexe. Plus spécifiquement, si le graphe est fragmenté plusieurs fois pendant le processus de suppression progressive des sommets, alors il y a plusieurs pics. Plus la valeur de ps est petite, meilleure est la méthode de mesure de centralité utilisée pour réaliser le classement des sommets.

Le taux de déclin de l'efficacité : En général, en communication, lorsque deux sites sont proches, il est plus facile de transférer une information d'un site à l'autre. Ce principe est utilisé pour définir l'efficacité d'un graphe [115].

$$\text{Eff} = \frac{1}{n(n-1)} \sum_{i \neq j} \frac{1}{d_{ij}} \tag{3.12}$$

Où n est le nombre de sommets total du graphe et d_{ij} est le plus court chemin entre

les sommets i et j. Puisque, lorsqu'on supprime un sommet du graphe, le chemin le plus court moyen entre les sommets devient plus grand, il y a alors une détérioration de l'efficacité du graphe. Pour mesurer cette détérioration, on utilise la formule suivante :

$$TD_{\text{eff}} = 1 - \frac{\text{Eff}}{\text{Eff}_0} \tag{3.13}$$

où Eff_0 est l'efficacité du graphe d'origine.

Le taux de déclin permet ainsi de quantifier la diminution de l'efficacité du graphe lorsque des sommets sont supprimés. Plus le taux est proche de 1, plus la détérioration est grande ; plus le taux est proche de 0, plus la détérioration est faible.

Robustesse : Une composante connexe dans un graphe est un sous-graphe dans lequel toutes les paires de sommets sont connectées. Lorsqu'un graphe est constitué de plusieurs composantes connexes, la composante contenant le plus grand nombre de sommets est la plus grande composante connexe ou parfois la composante connexe maximale [120]. Le plus grand coefficient de connectivité est alors défini comme le rapport de la taille de la plus grande composante connexe (n_{gcc}) sur le nombre total de sommets du graphe (N). Soit

$$r = \frac{n_{gcc}}{N} \tag{3.14}$$

La robustesse d'une méthode [120] est définie comme étant la moyenne des plus grands coefficients de connectivité obtenus en enlevant un sommet du graphe.

$$R = \sum_{i=1}^{N} r_i \tag{3.15}$$

où r_i est le plus grand coefficient de connectivité en enlevant le sommet i. Plus le R est petit, plus la méthode de mesure de centralité est meilleure.

3.5 Mesures de lisibilité

Dans cette section, nous présentons quelques mesures de lisibilité utilisées dans la partie application.

Flesch reading ease (`Fre`) : Le score de facilité de lecture de Flesch est une mesure conçue pour indiquer la difficulté de compréhension d'un texte. Sa formule prend en compte la longueur moyenne des phrases et le nombre moyen de syllabes par mot [41].

$$\text{Fre} = 206,835 - 1,015\frac{W}{S} - 84,6\frac{Y}{W} \tag{3.16}$$

où W est le nombre de mots, S le nombre de phrases et Y le nombre de syllabes.

En générale, le score de Flesch est compris entre 0 et 100. Plus le score est élevé, plus le texte est facile à lire.

Indice de Coleman-Liau (`Cli`) : L'indice de Coleman-Liau est un test de lisibilité conçu par Meri Coleman et T.L. Liau pour évaluer la compréhensibilité d'un texte [29]. La formule de l'indice de Coleman-Liau prend en compte le nombre moyen de lettres et le nombre moyen de phrases.

$$\text{Cli} = 0,0588L - 0,296S - 15,8 \tag{3.17}$$

où L est le nombre moyen de lettres pour cent mots, S le nombre moyen de phrases pour cent mots.

Indice de Gunning Fog (`Gfog`) : L'indice de Gunning Fog, développé par Robert Gunning [55], estime le nombre d'années d'étude nécessaires pour comprendre un texte. Sa formule prend en compte le nombre de mots par phrase et le pourcentage de mots complexes. En général, les mots complexes sont ceux comportant plus de deux syllabes.

$$\text{Gfog} = 0.4\frac{W}{S} + 40\frac{W_{syl\geq 3}}{W} \tag{3.18}$$

où W est le nombre de mots, S le nombre de phrases et $W_{syl\geq 3}$ le nombre de mots comportant plus de deux syllabes.

Indice automatisé de lisibilité (`Ari`) : L'indice automatisé de lisibilité évalue la lisibilité d'un texte en fonction de la longueur des mots et des phrases du texte [104].

$$\text{Ari} = 4,71\frac{C}{W} + 0,5\frac{W}{S} - 21,43 \tag{3.19}$$

où W est le nombre de mots, S le nombre de phrases et C le nombre de caractères.

L'`Ari` donne un score qui correspond approximativement au niveau scolaire nécessaire pour comprendre le texte.

Indice de Dale-Chall (`Dcr`) : L'indice de Dale et Chall est une formule de lisibilité créée par Edgar Dale et Jeanne Chall [31]. Il estime le niveau d'étude nécessaire pour lire et comprendre un texte. Sa formule est :

$$\text{Dcr} = 3.6365 + 0.0496\frac{W}{S} + 15.79\frac{W_d}{W} \tag{3.20}$$

où W est le nombre de mots, S le nombre de phrases et W_d le nombre de mots difficiles.

Pour Dale, les mots difficiles sont des mots absents de la liste des trois mille mots les plus couramment utilisés qu'il a établie. Mais établir une liste est parfois fastidieux, alors la difficulté d'un mot est évaluée parfois par le nombre de syllabes qu'il compte. Dans cette thèse, nous adoptons la deuxième façon de faire.

3.6 Déformation temporelle dynamique (DTW)

La déformation temporelle dynamique (en anglais *Dynamic Time Warping*) [97] est une mesure de similarité entre des séries chronologiques de tailles potentiellement différentes. Par exemple, pour deux séries chronologiques $X = (X_0, X_1, \ldots, X_{n-1})$ et $Y = (Y_0, Y_1, \ldots, Y_{m-1})$ de longueurs respectives n et m, la déformation temporelle dynamique entre X et Y est donnée par :

$$DTW(x, y) = \min_{\pi} \sqrt{\sum_{(i,j)\in\pi} d(x_i, y_j)^2} \tag{3.21}$$

où d est une distance (généralement la distance euclidienne) et $\pi = [\pi_0, \pi_1, ..., \pi_k]$ respecte les propriétés suivantes :

— π est une liste de paires d'indices telles que $\pi_k = (i_k, j_k)$ avec $0 \leqslant i_k \leqslant n$ et $0 \leqslant j_k \leqslant m$. De plus, $\pi_0 = (0, 0)$ et $\pi_k = (n-1, m-1)$.

— Pour tout $k > 0$, $\pi_k = (i_k, j_k)$ est relié à $\pi_{k-1} = (i_{k-1}, j_{k-1})$ comme suit :

- $i_{k-1} \leqslant i_k \leqslant i_{k-1} + 1$
- $j_{k-1} \leqslant j_k \leqslant j_{k-1} + 1$.

3.7 Simulation du graphe de processus d'écriture

Dans cette section, nous présentons la méthode que nous utilisons pour simuler les fichiers logs permettant de réaliser certaines comparaisons dans le chapitre des applications (6).

Les graphes de processus d'écriture issus des fichiers log simulés sont essentiellement utilisés pour les études structurelles. En effet, certains attributs des sommets du graphe, tels que le caractère inséré et le temps d'insertion, ne sont pas utilisés. Les trois principaux attributs utilisés sont : le numéro du sommet, le type du sommet et la position à laquelle l'événement du sommet est effectué. Nous nous concentrons alors à simuler ces trois constituants du fichier log.

En utilisant les quelques exemples à notre disposition, nous avons remarqué que la distribution de ces trois attributs suit parfois des lois particulières et parfois non. Étant donné que l'échantillon des processus d'écriture utilisé n'est pas grand, certaines façons d'écrire peuvent

être absentes de cet échantillon. Par conséquent, certaines considérations peuvent sembler peu intuitives, car elles ne tiennent pas compte de tous les styles de rédactions possibles. En d'autres termes, certaines considérations risquent de ne pas refléter fidèlement la réalité de l'écriture.

Type : Le type (insertion ou suppression) suit une loi de Bernoulli de paramètre $p = 0.8$, où p est la probabilité que le type soit une insertion et $1 - p$ la probabilité que le type soit une suppression.

Position : La position ne suit pas une loi particulière connue a priori. En effet, les numéros des positions peuvent être consécutifs jusqu'à un certain niveau avant de changer vers un autre numéro. En d'autres termes, on obtient plutôt des séquences de numéros de position consécutifs de taille variée. Ainsi, pour simuler la liste des positions, nous effectuons deux tirages aléatoires : le premier pour trouver la position de départ de la série et le deuxième pour trouver la taille de la série. Pour simplifier, nous utilisons une distribution uniforme pour déterminer le début et la taille des séries. Ainsi, la position de début de la série est choisie aléatoirement entre les nombres entiers compris entre 1 et la longueur du texte courant. Et la taille de la série est tirée aléatoirement entre les nombres entiers compris entre 1 et le nombre de lignes restant pour compléter la taille du fichier log voulu.

Numero : Correspond au numéro de ligne et n'est pas aléatoire, car c'est une suite consécutive de numéros de 1 à la taille du fichier log.

Plus formellement, on a le pseudo-algorithme suivant :

`Notations` :

Textlength :	`longueur du texte`
Posi :	`position du curseur au début d'une série de positions consécutives`
Posilength :	`longueur d'une série de positions consécutives`
Type :	`type du sommet` I `pour une insertion et` S `pour une suppression.`
Num :	`numéro de la ligne.`
logFile :	`fichier log (`*logFileNum* `colonne des numéros,` *logFileType* `colonne des types,` *logFilePosi* `colonne des positions)`
nlig :	`taille (nombre de lignes) du fichier log`
nbren :	`nombre de lignes déjà produites`

Algorithme 2 : Simulation Fichier Log

Entrées : nlig
Sorties : logFile
$Textlength = 0$;
$nbren = 0$;
tant que $nbren < nlig$ **faire**
  si $Textlength = 0$ **alors**
    $Posi = 1$;
    $Posilength =$ choisir un entier entre $[1, n - nbren]$;
    pour i *de* 0 *à* $Posilength$ **faire**
      $logFileNum = logFileNum + [nbren + i + 1]$; `Ajouter un numéro à la ligne.`
      $logFileType = logFileType + [I]$; `Ajouter un type à la ligne.`
      $logFilePosi = logFilePosi + [posi + i]$; `Ajouter une position à la ligne.`
    fin
    $Textlength = Textlength + Posilength$; `Ajouter la longueur du texte.`
  sinon
    $Posi =$ choisir un entier entre $[1, Textlength]$;
    $Type =$ choisir entre `'I'` et `'S'` suivant une loi de Bernoulli de paramètre $p = 0.8$ (p pour une insertion et $1 - p$ pour une suppression) ;
    si $Type =$ `'I'` **alors**
      $Posilength =$ choisir un entier entre $[1, n - nbren]$;
      pour i *de* 0 *à* $Posilength$ **faire**
        $logFileNum = logFileNum + [nbren + i + 1]$;
        $logFileType = logFileType + [I]$;
        $logFilePosi = logFilePosi + [posi + i]$;
      fin
      $Textlength = Textlength + Posilength$
    sinon
      $Posilength =$ choisir un entier entre $[1, min(n - nbren, Posi)]$;
      pour i *de* 0 *à* $Posilength$ **faire**
        $logFileNum = logFileNum + [nbren + i + 1]$;
        $logFileType = logFileType + [S]$;
        $logFilePosi = logFilePosi + [posi - i]$;
      fin
      $Textlength = Textlength - Posilength$
    fin
  fin
  $nbren = nbren + Posilength$
fin
$logFile = [logFileNum, logFileType, logFilePosi]$;
retourner $logFile$

3.8 Conclusion

En partant de la notion de graphe, ce chapitre a présenté, tour à tour, le *modèle sans perte*, des algorithmes pour la détection des communautés, des mesures de centralité, des mesures de lisibilité et une méthode de création d'un fichier log simulant un processus d'écriture, qui sont les différents outils et méthodes qui seront utilisés dans les chapitres suivants. Si une notion n'est pas définie dans ce chapitre, alors elle sera définie lors de son utilisation.

CHAPITRE 4 COMMUNAUTÉS DE MODIFICATION

Certaines parties de ce chapitre font partie de l'article : Visualising the stages of a writing process [109].

Ce chapitre présente la notion de communauté de modification d'un graphe de processus d'écriture qui est une adaptation de la notion générale de communauté dans un graphe, appliquée au cas particulier du graphe de processus d'écriture. Bien que plein de méthodes de détection de communautés existent, nous avons décidé de développer notre propre méthode de détection, car, comme nous allons le voir à la section 4.4, les autres méthodes ne produisent pas toujours les communautés qui répondent à nos critères de communauté.

Lorsqu'on écrit un texte, dans la majeure partie des cas, ce qu'on écrit en premier est absent du texte final. En effet, nous sommes parfois amenés à changer ce que l'on écrit à cause des fautes ou de nouvelles idées. Mais malheureusement, ces changements n'apparaissent pas dans le texte final. Ainsi, personne n'a connaissance des modifications effectuées. Certes, avec le développement des outils informatiques, plusieurs techniques sont utilisées pour analyser le processus d'écriture à travers les fichiers d'enregistrement des frappes, mais il reste toutefois très difficile à analyser. Partant du *modèle sans perte*, ce chapitre définit la notion de communauté de modification, qui se veut être un ensemble de sommets qui traduit à la fois la structure du graphe et le texte écrit lors d'une modification.

La section 4.1 présente comment subdiviser le processus d'écriture en communautés de modifications et propose aussi une visualisation du processus d'écriture à l'aide de ces communautés. Ensuite, la section 4.2 présente les méthodes pour regrouper des communautés afin d'obtenir des communautés plus larges, offrant ainsi une vision plus concise du processus d'écriture. La section 4.3 introduit des diagrammes qui permettent de synthétiser l'information contenue dans les communautés. Enfin, la section 4.4 donne des avantages et des inconvénients des communautés de modifications.

Dans ce chapitre, un *graphe de processus d'écriture* G est défini par un ensemble $W = \{\text{début, fin}\} \cup V$ de sommets avec $V = \{1, 2, 3, ..., n\}$, par un ensemble A d'arcs reliant ces sommets et par une fonction $\mathcal{C} : V \longrightarrow \{Bleu, Jaune, Rouge\}$ qui attribue l'une des trois couleurs à chaque sommet de V. Le graphe contient l'arc (début, fin) ainsi que deux arcs (x, v) et (v, y) avec $x \in \{\text{début}\} \cup V$ et $y \in \{\text{fin}\} \cup V$ pour tout $v \in V$. Les sommets `début` et `fin` représentent respectivement les sommets -1 et 0 de la description du *modèle sans perte* du chapitre 3 (Outils et méthodes). Le graphe a donc exactement $2|V| + 1$ arcs.

4.1 Subdivision du processus d'écriture en communautés de modification

Cette section porte sur la définition de communautés de modifications et la visualisation du processus d'écriture à l'aide de ces communautés.

La tâche de la détection des communautés dans un graphe est de trouver des sous-ensembles de sommets connectés, tel que les sommets d'un même sous-ensemble ont beaucoup de liens entre eux et peu de lien avec les sommets des autres sous-ensembles. Dans le graphe de processus d'écriture (*modèle sans perte*), on a deux types de liens : le lien spatial donné par les arcs et le lien chronologique donné par les numéros des sommets. Ainsi, trouver les communautés dans le graphe de processus d'écriture revient à trouver un ensemble de sommets qui ont des liens à la fois chronologiques et spatiaux entre eux. En plus de cela, les communautés identifiées doivent traduire au mieux les modifications effectuées dans le texte lors du processus d'écriture. Il nous faut alors définir des critères pour la construction des communautés qui respectent toutes ces exigences.

Dans la sous-section 4.1.1, il sera question de présenter les différentes actions effectuées au cours d'un processus d'écriture. Ensuite, la sous-section 4.1.2 portera sur la définition des communautés de modifications. Et enfin, la sous-section 4.1.3 présente une visualisation du processus d'écriture à l'aide de ces communautés.

4.1.1 Les types d'actions effectuées lors d'un processus d'écriture

Le processus d'écriture est très complexe à étudier, car plusieurs paramètres entrent en jeu. En effet, l'évolution des outils informatiques d'enregistrement de l'écriture fait en sorte que plusieurs événements qui se produisent au cours d'un processus d'écriture peuvent être pris en compte lors de l'enregistrement du processus. Alors que les premiers logiciels ne considéraient que les frappes effectuées au clavier, maintenant il est possible de prendre en compte plusieurs autres actions, comme le déplacement du curseur, le copier-coller, et même parfois les événements externes, comme la position des yeux lors de l'écriture. Mais le *modèle sans perte* que nous utilisons ne prend en compte que les insertions, les suppressions et le déplacement du curseur. Ainsi, en utilisant les notations définies à la section 3.2, nous considérons trois différents types d'actions effectuées par le rédacteur :

Type 1 **:** Le rédacteur peut déplacer le curseur de place dans le texte.

$Pos(i) \neq p_{i-1}(i-1)$ et $Pos(i) \neq p_{i-1}(i-1)+1$, avec $i > 0$ et $p_0(0) = 0$.

Type 2 **:** Le rédacteur peut insérer un caractère où le curseur est placé.

$Pos(i) = p_{i-1}(i-1)+1$, avec $i > 0$ et $p_0(0) = 0$.

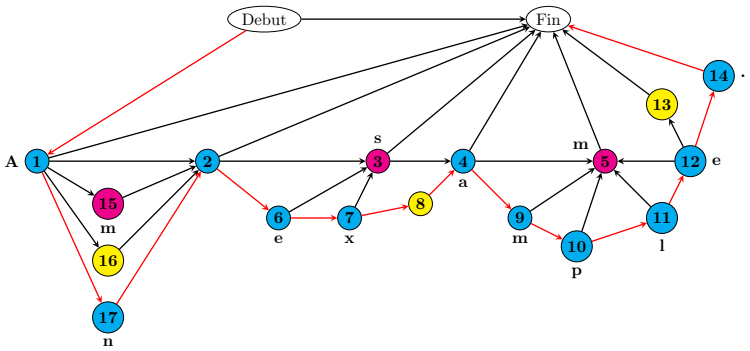
Type 3 **:** Le rédacteur peut supprimer un caractère adjacent au curseur. La suppression peut se faire à gauche du curseur (type 3.1)

$Pos(i) = p_{i-1}(i-1)$, avec $i > 0$ et $p_0(0) = 0$

ou à droite du curseur (type 3.2)

$Pos(i) = p_{i-1}(i-1) + 1$, avec $i > 0$ et $p_0(0) = 0$.

Une illustration des différentes actions est donnée par la Figure **4.1**.



(a) Graphe modèle sans perte

T_1 : 'A sam'

T_2 : 'A **ex**~~*s*~~*am*'

T_3 : 'A *exa***mple**~~*m*~~.'

T_4 : '*A*~~**mn**~~ *example.*'

Texte Final : 'An example.'

(b) Processus d'écriture

FIGURE 4.1 Graphe de processus d'écriture.

Dans l'exemple, les insertions sont représentées avec les lettres en gras et les lettres supprimées sont barrées. Le texte final produit par le rédacteur est « `An example.` ». On remarque que le texte n'est pas écrit d'un seul jet. Initialement, le rédacteur écrit « `A sam` ». Après, il déplace le curseur avant le « `s` » (action type 1), insert « `ex` » (action type 2) et supprime « `s` » situé à droite du curseur (action type 3.2). Le curseur est ensuite déplacé devant le « `m` » (action type 1) et le rédacteur insère « `mple` » (action type 2) et efface le dernier « `m` » (action type 3.2) et le remplace par le point. Finalement, le rédacteur bouge à nouveau le curseur, cette fois-ci pour se positionner après la lettre « `A` » du début et écrit « `m` » (action type 2) et supprime le « `m` » qui se trouve à gauche du curseur (action type 3.1) et le remplace par « `n` ».

Dans le *modèle sans perte*, les trois types d'actions sont représentés comme suit :

Type 1 **:** Déplacement du curseur.

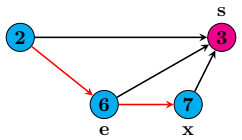
Un sommet d'insertion de numéro x, $(x > 1)$, représente une insertion après déplacement du curseur s'il n'existe pas d'arc $(x-1, x)$.

Un sommet de suppression de numéro x, $(x > 1)$, représente une suppression après déplacement du curseur, s'il n'existe pas de sommet y tel qu'on ait soit $(x-1, y)$ et (x, y) ou soit $(y, x-1)$ et (y, x).

N.B. : Lorsque le déplacement du curseur est directement suivi d'une suppression, qu'elle soit à droite ou qu'elle soit à gauche du curseur, elle est considérée comme une suppression à gauche.

Type 2 : Insertion à la place où se situe le curseur.

Un sommet de numéro x, ($x > 1$), représente une insertion à la place où se situe le curseur si l'arc $(x-1, x)$ existe. De plus, les sommets $x-1$ et x ont un même successeur dans le graphe. Par exemple, l'insertion de « `x` » avant le « `s` » à la suite de « `e` » donne :

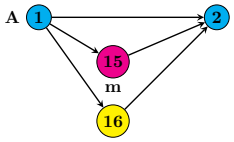


Type 3 : Suppression à gauche ou à droite du curseur.

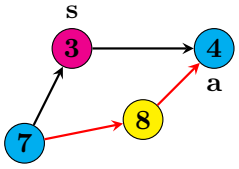
Soit y le numéro du sommet correspondant au caractère à gauche du curseur.

Soit z le numéro du sommet correspondant au caractère à droite du curseur.

— Le sommet de suppression de numéro x représente une suppression à gauche du curseur, s'il existe l'arc (x, z). C'est le cas de la suppression du « `m` » au début du texte, on a $x = 16$, $y = 15$ et $z = 2$.



— Le sommet de suppression de numéro x représente une suppression à droite du curseur si l'arc (y, x) existe. C'est le cas de la suppression de « `s` », on a $x = 8$, $y = 7$ et $z = 3$.



De plus, les sommets de suppression sont représentés en jaune.

En résumé, nous avons défini 3 types d'actions effectuées lors d'un processus d'écriture et donné leur correspondance dans le *modèle sans perte*. Il reste maintenant à donner la définition de ce que nous entendons par « `communauté de modifications` ».

4.1.2 Communauté dans le *modèle sans perte*

Le terme communauté est généralement utilisé pour faire référence au groupe de personnes ou d'objets qui ont des caractéristiques particulières en commun dans un domaine spécifique. Nous utilisons le terme « communauté » ici comme synonyme des phases d'un processus d'écriture que nous définissons comme un ensemble maximal d'actions consécutives effectuées sans déplacer le curseur. Cet ensemble d'actions est appelé une communauté de modifications ou encore épisode de modifications. En d'autres mots, c'est l'ensemble maximal d'actions consécutives de type 2 ou 3. Dans l'exemple 4.1b, chaque étape constitue une communauté de modifications lorsque le curseur est déplacé à une nouvelle position à la fin de chaque étape. Pour définir de manière plus formelle les communautés de modification, considérons le principe suivant :

Pc : Un sommet $x \in V$ vérifie `Pc (Principe de communauté)`, si x vérifie au moins un des cas suivant :

$\mathbf{Pc}_1$: $(x, x+1) \in A$

$\mathbf{Pc}_2$: x est un sommet de suppression et les sommets x et $x+1$ ont un même prédécesseur ou un même successeur. En d'autres termes, il existe un sommet $y \in W$ tel que $(x, y) \in A$ et $(x+1, y) \in A$ ou $(y, x) \in A$ et $(y, x+1) \in A$.

Définition 4.1.1 (Communauté de modifications)**.** Soit un graphe de processus d'écriture G. On dit qu'un ensemble de sommets de numéros consécutifs $C = \{x_1, x_2, \ldots, x_k\}$ de G est une communauté de modifications, lorsque pour tout sommet $x_i \in C$ on a un des cas suivants :

1. $x_i + 1 \in C$ et x_i vérifie `Pc`
2. $x_i - 1 \in C$ et $x_i - 1$ vérifie `Pc`

Autrement dit, deux sommets de numéros consécutifs x et $x+1$ sont dans la même communauté si on est dans l'un des cas suivants :

1. Il existe l'arc $(x, x+1)$.
2. Les deux sommets sont des suppressions et ils ont le même plus grand prédécesseur ou le même plus grand successeur.
3. L'un des sommets est un sommet d'insertion et l'autre est un sommet de suppression, et ils ont le même plus grand prédécesseur.

N.B. : Le plus grand prédécesseur (successeur) est le prédécesseur (successeur) avec le plus grand numéro.

Il en résulte, alors l'**Algorithme 3** de détection des communautés de modification ou épisodes de modifications dans le graphe de processus d'écriture.

`Notations` :

$Predg(x)$:	`le plus grand prédécesseur du sommet` x
$Sucg(x)$:	`le plus grand successeur du sommet` x
$Type(x)$:	`nature du sommet` x `(insertion(I), suppression(S) ou insertion supprimée(IS))`
C_j :	`Communauté de modification numéro` j

Algorithme 3 : Création de communautés de modification

Entrées : G de processus d'écriture
Sorties : $C = \{C_1, \ldots, C_m\}$ ensemble des communautés de modifications

```
n ← |V|;
j ← 1;
C_j ← {1};
i ← 2;
tant que i ⩽ n faire
    si arc (i − 1, i) existe alors
        Ajouter le sommet i à C_j
    sinon
        si Type(i) =S et (Predg(i) = Predg(i − 1) ou Sucg(i) = Sucg(i − 1)) alors
            Ajouter le sommet i à C_j
        sinon
            si Type(i − 1) =S et (Predg(i) = Predg(i − 1) ou Sucg(i) = Sucg(i − 1)) alors
                Ajouter le sommet i à C_j
            sinon
                j ← j + 1;
                C_j ← {i};
            fin
        fin
    fin
    i ← i + 1;
fin
retourner C
```

En appliquant l'**Algorithme 3** au graphe de l'exemple **4.1a** et attribuant des couleurs différentes pour chaque communauté, on obtient le graphe de la Figure **4.2**.

On remarque qu'avec le *modèle sans perte*, on peut se retrouver avec un nombre de sommets très élevé. Dans ce cas, la visualisation des communautés n'est pas facile. De plus, il n'est pas

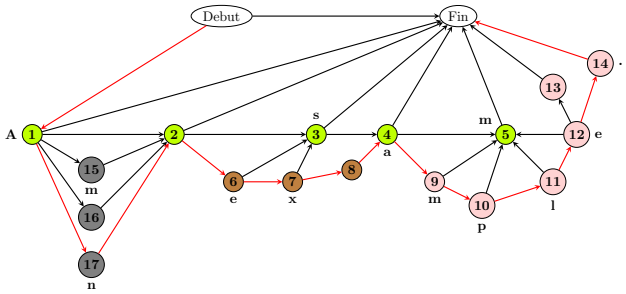


FIGURE 4.2 Graphe de processus d'écriture avec communauté de modifications.

évident de faire directement le rapprochement entre les phases d'écriture et les communautés. Pour pallier à ce manque, on essaie de trouver une visualisation qui est plus intuitive.

4.1.3 Visualisation du processus à l'aide des communautés de modification

Cette sous-section porte sur la visualisation des communautés de modifications, par le même fait sur la visualisation des processus d'écriture.

Comme l'illustre la Figure **4.3**, nous utilisons les barres, les pointeurs et les flèches pour représenter à la fois l'évolution du texte et la constitution des communautés. En effet, chaque élément utilisé renvoie à un événement effectué dans le texte.

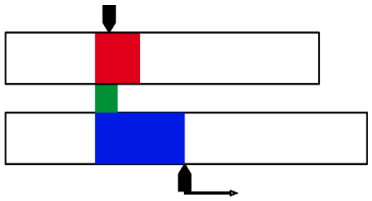


FIGURE 4.3 Visualisation d'une communauté de modifications.

— Les deux barres horizontales : la première est une illustration du texte au début de la modification, la seconde, le texte à la fin de la modification.
— Les pointeurs : sur la première barre, indiquent la position au début de la modification faite dans la communauté, et en bas de la seconde barre, indiquent la position du curseur à la fin de la modification.
— La flèche en dessous du second pointeur représente le mouvement du curseur vers le début de la prochaine modification, c'est-à-dire vers le prochain pointeur.
— L'espace blanc dans les barres horizontales est la partie du texte qui n'est pas changée durant la modification.

— La portion rouge dans la première barre horizontale indique la partie du texte qui a été supprimée. La suppression peut se faire à gauche comme à droite du curseur, c'est ce qui explique pourquoi le pointeur qui indique le début de la modification peut être au milieu de la zone rouge.

— L'espace bleu dans la seconde barre horizontale indique le nouveau texte visible à la fin de la modification. D'autres caractères d'insertions peuvent avoir été effectués pendant la modification, mais ont été immédiatement supprimés ; de ce fait, ils n'apparaissent pas dans la modification finale.

— Le rectangle vert entre les deux barres horizontales indique le nombre de caractères qui ont été insérés et supprimés durant cette phase de modification.

En résumé, le nombre de caractères que le rédacteur insère dans une communauté de modifications est égal à la somme des longueurs des rectangles bleu et vert, alors que le nombre de caractères supprimés est égal à la somme des longueurs des rectangles rouge et vert.

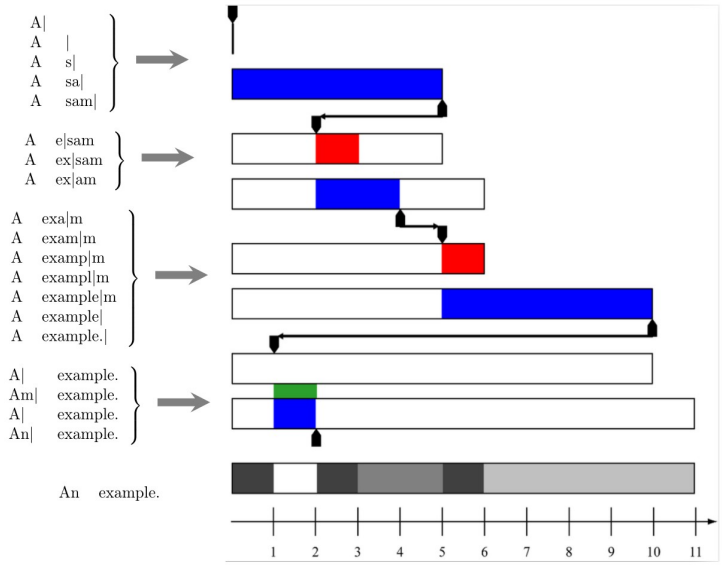


FIGURE 4.4 Visualisation des communautés de modifications de l'exemple 4.1a.

Pour illustration, la Figure **4.4** nous montre les quatre communautés de l'exemple. En plus des communautés, nous indiquons à gauche l'évolution du texte et chaque étape représente une communauté. De plus, en dessous de toutes les communautés, nous ajoutons un rectangle dont la longueur est celle du texte final. L'espace occupé par un caractère est coloré avec une nuance de gris. Nous utilisons k différentes nuances, où k est le nombre total de communautés. La première communauté a la couleur noire et la dernière la couleur blanche (sauf s'il n'y a qu'une communauté). Chaque partie dans le texte final a la couleur de la communauté dans

laquelle le caractère qui occupe l'emplacement a été inséré. Si les couleurs de nuances très différentes sont adjacentes l'une à l'autre dans le rectangle du texte final, cela signifie que la partie du texte où on l'observe a été changée longtemps après que la première version a été écrite. Comme on peut le voir dans l'illustration de notre exemple, au début du texte, un espace blanc est entouré par deux espaces noirs. Cela s'est produit par ce que le « `n` » dans la seconde position a été inséré à la toute fin, alors que le « `A` » à la gauche et l'espace blanc à la droite ont été écrits au tout début.

La représentation des communautés que nous avons choisie a des avantages variés. Elle permet la visualisation de l'étendue des modifications dans chaque phase du processus d'écriture, sa position dans le texte aussi bien que sa composition, en plus d'indiquer la direction du mouvement du curseur durant le processus d'écriture. Le nombre de communautés correspond au nombre de phases d'écriture, alors que la taille des zones rouge et bleue des communautés nous donne l'étendue et la composition des modifications. Plus la partie de la communauté rouge et bleue est grande, plus grand est le changement effectué. S'il n'y a que la partie bleue dans la communauté, cela signifie que le rédacteur a simplement ajouté de nouveaux textes. Par contre, s'il n'y a que la partie rouge, cela signifie que le rédacteur a uniquement supprimé le texte existant. La présence du rectangle vert indique que le nouveau texte inséré a été immédiatement supprimé. La figure **4.5** présente l'ensemble des cas possibles de représentation d'une communauté, en termes de présence ou d'absence d'insertions et de suppressions.

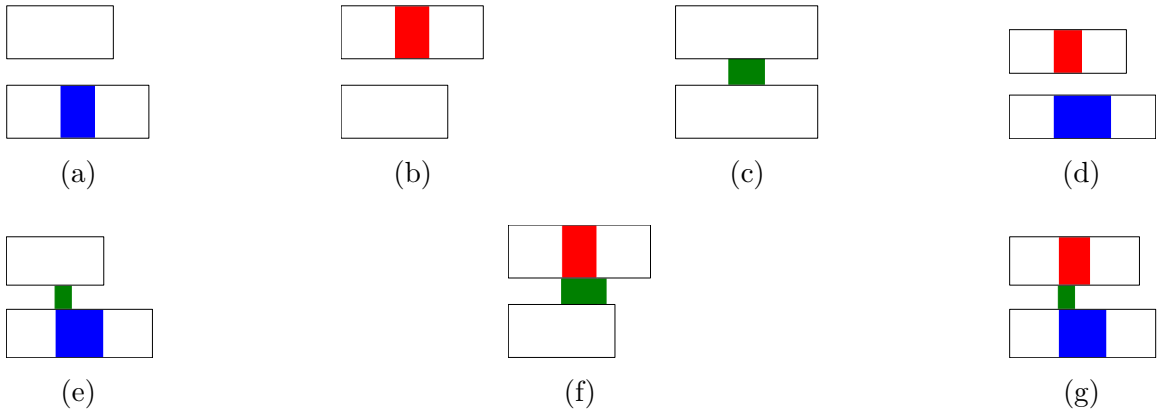


FIGURE 4.5 Les configurations possibles d'une communauté.

En résumé, en observant le diagramme représentant les communautés, nous pouvons avoir une idée précise des modifications effectuées par le rédacteur, de son étendue et de sa position dans le texte final.

4.2 Fusion des communautés et processus d'écriture

Lors de la révision d'un texte, il peut arriver qu'on fasse de petits changements çà et là dans le texte. Ces changements touchent parfois des parties très proches dans le texte, mais le fait de déplacer le curseur un peu à droite ou à gauche fait en sorte que ces modifications apparaissent dans des communautés différentes. De plus, ces petits changements peuvent interrompre temporairement le processus d'écriture. C'est le cas, par exemple, lorsque le rédacteur remarque de petites fautes de frappe dans différentes parties du texte non nécessairement proches les unes des autres. Ainsi, le rédacteur peut déplacer le curseur dans différentes parties pour effectuer de petites modifications. Ainsi, nous avons deux notions qui en découlent, la proximité des modifications et la taille des modifications. En suivant ce genre de raisonnement, on peut se retrouver avec plusieurs critères qui peuvent permettre de regrouper des communautés. Dans cette section, nous nous intéressons à deux types de fusion basés sur deux différents critères, la proximité des communautés et la taille des communautés.

La sous-section 1 porte sur le critère de proximité des modifications et la sous-section 2 porte sur le critère de la taille des communautés.

Pour illustrer la fusion des communautés, nous considérons un exemple de texte un peu plus grand, composé de 18 communautés (Figure **4.6**).Ce texte est construit de toutes pièces pour l'exemple en utilisant le logiciel GGXLog [28]. Supposons qu'il s'agit de la visualisation des 18 communautés de la Figure **4.7**

4.2.1 Proximité des modifications

Le premier type de fusion concerne les communautés successives qui nécessitent un petit déplacement du curseur pour aller d'une communauté à la suivante.

Le principe est alors de fusionner deux communautés successives si le mouvement du curseur entre deux communautés est plus petit qu'un seuil. L'**Algorithme 4** permet d'effectuer ces fusions.

Par exemple, dans la Figure **4.7**, on peut remarquer que la longueur du déplacement du curseur entre deux communautés varie de 2 à 309 caractères. La majeure partie de ces déplacements est de longueur plus petite que 20 caractères, alors que seulement 3 sont plus grands que 80 caractères. Comme le montre la Figure **4.8a**, si nous fusionnons deux communautés consécutives quand le mouvement du curseur est plus petit que 20 caractères de long, le nombre de communautés diminue à 11. En effet, les communautés $C_6, C_7, ..., C_{11}$ fusionnent pour former une communauté, tandis que les communautés C_{12}, C_{13} et C_{14} en forment une autre. Cela signifie que ces modifications sont effectuées à moins de 20 caractères d'écart.

Texte produit par le rédacteur :

A community of modification is a series of successive insertions and/or deletions at a specific point in the text. For example, if you write "am" between "x" and "p" in the word "Exple", you get "Example". So "am" represents the community of modifications of the word "Exple". In graph terms, the community of modification is the set of nodes with successive numbers (chronological links) and arcs (spatial links). However, due to the high number of nodes in very large graphs, it is sometimes not easy to observe communities.

Décomposition en communautés C_i de modifications :

C_1 : **A modification are series of successive insertions.**

C_2 : *A* **community of** *modification are series of successive insertions.*

C_3 : *A community of modification are series of successive insertions* **and/or deletions**.

C_4 : *A community of modification* ~~are~~**is a** *series of successive insertions and/or deletions.*

C_5 : *A community of modification is a series of successive insertions and/or deletions.* **For example, if you write "xyz" between "b" and "c", "xyz" represents the community of modification of "abcd".**

C_6 : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "*~~xyz~~**am***" between "b" and "c", "xyz" represents the community of modification of "abcd".*

C_7 : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "am" between "*~~b~~**x***" and "c", "xyz" represents the community of modification of "abcd".*

C_8 : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "am" between "x" and "*~~p~~**ep***", "xyz" represents the community of modification of "abcd".*

C_9 : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "am" between "x" and "p",* **in the word** *"xyz" represents the community of modification of "abcd".*

C_{10} : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "am" between "x" and "p", in the word "*~~xyz~~**Exple***" represents the community of modification of "abcd".*

C_{11} : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "am" between "x" and "p", in the word "Exple" ,***you get "Example". So "am"** *represents the community of modification of "abcd".*

C_{12} : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "am" between "x" and "p", in the word "Exple" ,you get "Example". So "am" represents the community of modification of* **the word** *"abcd".*

C_{13} : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "am" between "x" and "p", in the word "Exple" , you get "Example". So "am" represents the community of modification of the word "*~~abcd~~**Exple**".

C_{14} : *A community of modification is a series of successive insertions and/or deletions. For example, if you write "am" between "x" and "p", in the word "Exple", you get "Example". So "am" represents the community of modification of the word "Exple".* **In graph terms, the community is the set of nodes. However, due to the high number of nodes it is sometimes difficult to observe communities.**

C_{15} : *A community of modification is a series of successive insertions and/or deletions* **at a specific point in the text** *. For example, if you write "am" between "x" and "p", in the word "Exple" , you get "Example". So "am" represents the community of modification of the word "Exple". In graph terms, the community is the set of nodes. However, due to the high number of nodes it is sometimes difficult to observe communities.*

C_{16} : *A community of modification is a series of successive insertions and/or deletions at a specific point in the text. For example, if you write "am" between "x" and "p", in the word "Exple" , you get "Example". So "am" represents the community of modification of the word "Exple". In graph terms, the community is the set of nodes* **with successive numbers (chronological links) and arcs (spatial links)**. *However, due to the high number of nodes it is sometimes difficult to observe communities.*

C_{17} : *A community of modification is a series of successive insertions and/or deletions at a specific point in the text. For example, if you write "am" between "x" and "p", in the word "Exple" , you get "Example". So "am" represents the community of modification of the word "Exple". In graph terms, the community is the set of nodes with successive numbers (chronological links) and arcs (spatial links). However, due to the high number of nodes* **in very large graphs,** *it is sometimes difficult to observe communities.*

C_{18} : *A community of modification is a series of successive insertions and/or deletions at a specific point in the text. For example, if you write "am" between "x" and "p", in the word "Exple" , you get "Example". So "am" represents the community of modification of the word "Exple". In graph terms, the community is the set of nodes with successive numbers (chronological links) and arcs (spatial links). However, due to the high number of nodes in very large graphs, it is sometimes* ~~difficult~~**not easy** *to observe communities.*

FIGURE 4.6 Exemple de processus d'écriture avec 18 communautés.

Ainsi, comme on peut le voir à la figure **4.8a**, à l'épisode 6, toutes les modifications sont réalisées sur la portion du texte écrit à l'épisode de modifications 5, ce qui semble indiquer une phase de révision. Lorsque le seuil passe à 60 (Figure **4.8b**) au lieu de 20, nous obtenons seulement quatre communautés. Quand le seuil augmente, le nombre de communautés diminue, ce qui nous donne une vision plus concise du processus d'écriture et des révisions effectuées. Ainsi, on peut observer que de petites révisions sont effectuées à la fin du texte.

En bref, en appliquant la fusion basée sur la proximité des communautés, nous obtenons des communautés qui donnent des informations sur tous les changements effectués successivement dans une localisation spécifique avec un seuil choisi pour le déplacement du curseur. Ce qui rend possible la détection des phases de révisions basée sur la proximité des modifications.

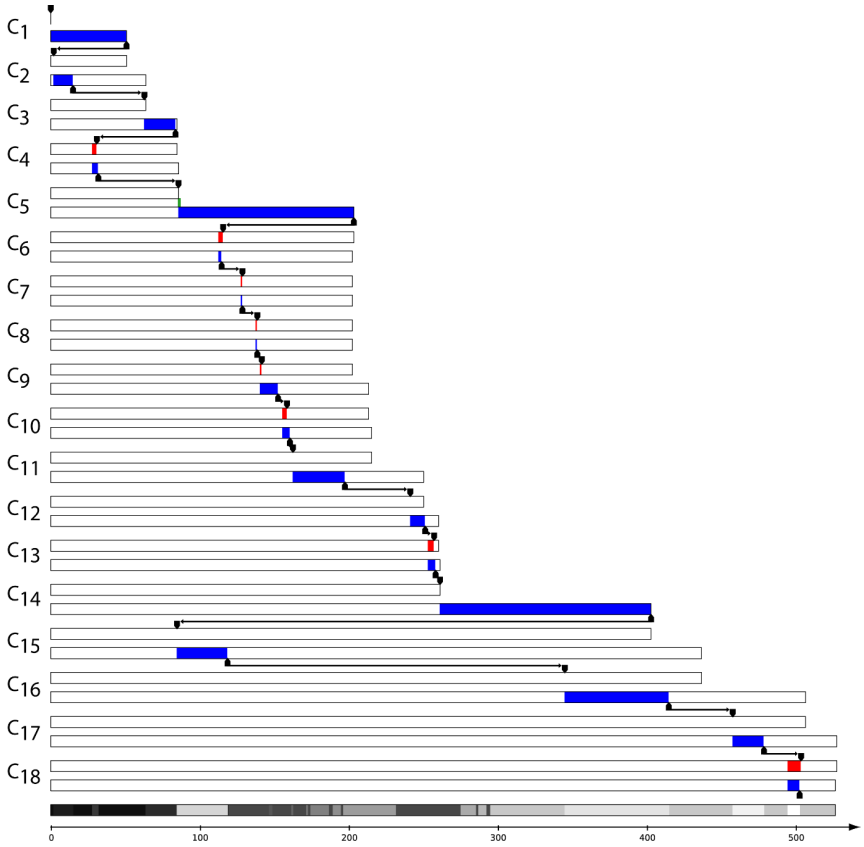
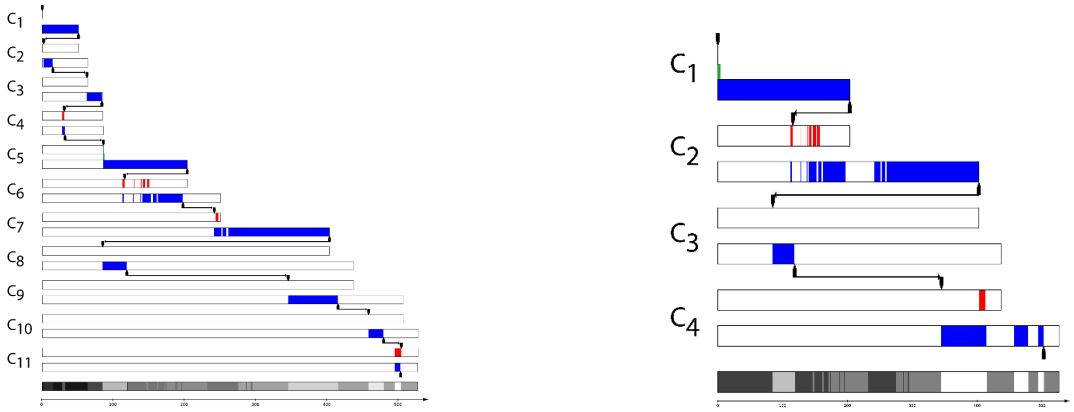


FIGURE 4.7 Visualisation des 18 communautés de modifications de l'exemple 4.6



(a) Seuil = 20

(b) Seuil = 60

FIGURE 4.8 Fusion par proximité.

Algorithme 4 : Fusion par Proximité des insertions

Entrées : $C = \{C_1, C_2, \ldots, C_m\}$ ensemble des communautés de modifications
Sorties : $C' = \{C'_1, C'_2, \ldots, C'_k\}$ communautés de modifications fusionnées

$m \leftarrow |C|$;
$i \leftarrow 1$;
$j \leftarrow 1$;
$C'_j \leftarrow C_i$;
$i \leftarrow 2$;
tant que $i \leqslant m$ **faire**
  $x \leftarrow$ `position du curseur à la fin de` C_{i-1};
  $y \leftarrow$ `position du curseur au début de` C_i;
  si $|y - x| \leqslant S_0$ **alors**
    $C'_j \leftarrow C'_j \cup C_i$;
  sinon
    $j \leftarrow j + 1$;
    $C'_j \leftarrow C_i$;
  fin
  $i \leftarrow i + 1$;
fin
retourner $C' = \{C'_1, \ldots, C'_k\}$

4.2.2 Taille des changements

Le second type de fusion concerne l'étendue des changements effectués dans la communauté. Quand deux communautés ont seulement de petites modifications, nous pouvons décider de les regrouper.

Posons

Ins(c) : la longueur de la partie bleue de la communauté c.

Ste(c) : la longueur de la partie rouge de la communauté c.

n_{mod_c} : le nombre de modifications de la communauté c.

Définition 4.2.1. Le `nombre de modifications` d'une communauté est la somme de la longueur de la partie rouge et bleue. En effet, il représente les modifications visibles à la fin des modifications de la communauté.

$$n_{mod_c} = \text{Ins}(c) + \text{Ste}(c)$$

Ce qui correspond au nombre de nouveaux caractères insérés qui sont visibles à la fin de la communauté, auquel on ajoute le nombre de caractères qui étaient présents au début de la

communauté et qui ont été supprimés. Le principe est alors de fusionner deux communautés successives si leur nombre de modifications est inférieur à un seuil donné.

L' **Algorithme 5** permet d'obtenir les fusions des communautés en fonction de la taille des communautés.

`Notations` :

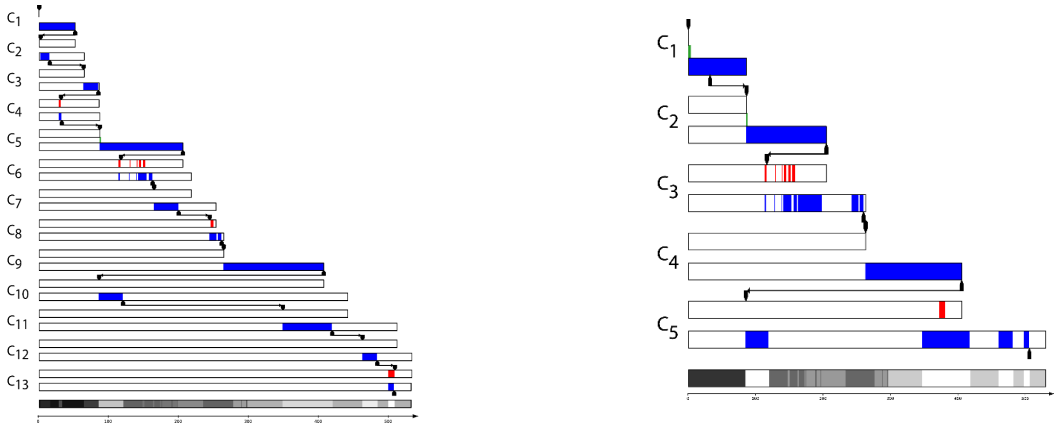
`Ins`(i) :	`Nombre d'insertions non supprimées immédiatement de la communauté` i
`Ste`(i) :	`Nombre de suppression du texte existant avant le début de la communauté` i
C_i :	`Communauté de modification numéro` i

Algorithme 5 : Fusion par Taille des modifications

Entrées : $C = \{C_1, C_2, \ldots, C_m\}$ ensemble des communautés de modifications ;
S_1 : seuil de l'ampleur du changement
Sorties : $C' = \{C'_1, C'_2, \ldots, C'_k\}$ communautés de modifications fusionnées
$m \leftarrow |C|$;
$i \leftarrow 1$;
$j \leftarrow 1$;
$C'_j \leftarrow C_i$;
$i \leftarrow 2$;
tant que $i \leqslant m$ **faire**
  si $Ins(i) + Ste(i) \leqslant S_1$ **alors**
    $C'_j \leftarrow C'_j \cup C_i$;
  sinon
    $j \leftarrow j + 1$;
    $C'_j \leftarrow C_i$;
  fin
  $i \leftarrow i + 1$;
fin
retourner $C' = \{C'_1, \ldots, C'_k\}$

Par exemple, dans la Figure **4.7**, la majeure partie des communautés ont une taille supérieure à 100 caractères. Si on considère un seuil de 20 et 70 caractères pour la fusion, nous obtenons les figures de la Figure **4.9**. Pour un seuil égal à 20, nous obtenons 13 communautés, alors qu'en augmentant à 70, nous obtenons 5 communautés.

En utilisant la fusion des communautés basées sur la taille, nous obtenons des communautés qui donnent des informations sur les changements successifs effectués avec une taille infé-



(a) Seuil = 20 (b) Seuil = 70

FIGURE 4.9 Fusion par taille.

rieure à un seuil choisi. Ce qui permet d'observer des épisodes de révision basés sur l'étendue des changements effectués. Nous pouvons donc relier deux modifications différentes qui sont distantes l'une de l'autre dans le texte final. Par exemple, quand nous comparons les figures **4.9a** et **4.9b**, nous observons que les dernières communautés obtenues avec un seuil égal à 70 incluent la $15^{ième}$ communauté en plus de la $16^{ième}$, $17^{ième}$ et $18^{ième}$. Ainsi, on a la modification 15 effectuée au début du texte, tandis que les trois autres sont effectuées à la fin. Ce critère peut être utilisé si l'on s'intéresse à l'agrégation des petites modifications qui sont successives dans le temps.

4.3 Diagramme de synthèse des communautés

Les communautés de modifications sont définies dans le but que l'on puisse avoir accès à des informations supplémentaires sur le processus d'écriture. Dans cette optique, cette section présente des diagrammes qui permettent de synthétiser l'information contenue dans les communautés. Pour illustrer clairement notre point, nous considérons l'exemple composé de 18 communautés de la section précédente. Dans la sous-section 4.3.1, nous présentons un diagramme de distribution qui donne une synthèse des insertions et suppressions effectuées. Ensuite, la section 4.3.2 nous présente un diagramme qui donne les contributions des différentes communautés à la formation du texte final.

4.3.1 Diagramme de distribution

Une communauté de modification est faite d'insertions et de suppressions. Lorsqu'un nouveau caractère est inséré dans le texte, il peut être supprimé immédiatement après son insertion (c'est le cas par exemple lors d'une faute de frappe) ou il peut aussi être supprimé bien après (par exemple, lors d'une relecture plus approfondie). S'il n'est pas supprimé, le nouveau texte écrit dans la communauté va devenir une partie du texte final. Nous pouvons ainsi classer les insertions en trois groupes :

— Insertions supprimées immédiatement (I_1)
— Insertions supprimées dans une communauté ultérieure (I_2)
— Insertions jamais supprimées et qui vont apparaître dans le texte final (I_3)

Nous avons alors la définition suivante :

Définition 4.3.1. Soit G un graphe de processus d'écriture. Soit x un sommet de G, soit C_x une communauté de modifications contenant x.

1. x est un sommet d'insertion supprimé immédiatement ($x \in I_1$), s'il existe un sommet $y \in G$, tel que $y \in C_x$ et y `supprime` x.
2. x est un sommet d'insertion supprimé ultérieurement ($x \in I_2$), s'il existe un sommet $y \in G$, tel que $y \notin C_x$ et y `supprime` x.
3. x est un sommet d'insertion jamais supprimé ($x \in I_3$), s'il n'existe pas de sommet qui `supprime` x.

Nous pouvons aussi diviser les suppressions effectuées dans une communauté en deux groupes :

— Suppression du texte qui vient tout juste d'être écrit dans la même communauté (D_1).
— Suppression d'une pièce de texte présent au début de la communauté (D_2).

On a aussi la définition suivante :

Définition 4.3.2. Soit G un graphe de processus d'écriture. Soit y un sommet de G, soit C_y une communauté de modification contenant y.

1. y est un sommet de suppression d'un nouveau texte ($y \in D_1$), s'il existe un sommet $x \in G$, tel que $x \in C_y$ et y `supprime` x.
2. y est un sommet de suppression d'un texte existant ($y \in D_2$), s'il existe un sommet $x \in G$, tel que $x \notin C_y$ et y `supprime` x.

La distribution est illustrée par un diagramme constitué d'un cercle et de deux barres. Le cercle est divisé en deux parties respectivement proportionnelles aux insertions réalisées (partie bleue), soit

$$\frac{|I_1|+|I_2|+|I_3|}{|I_1|+|I_2|+|I_3|+|D_1|+|D_2|}\%$$

du cercle et aux suppressions qui représente le reste du cercle. Les barres indiquent respectivement la proportion des différents types d'insertions et de suppressions. Comme le diagramme

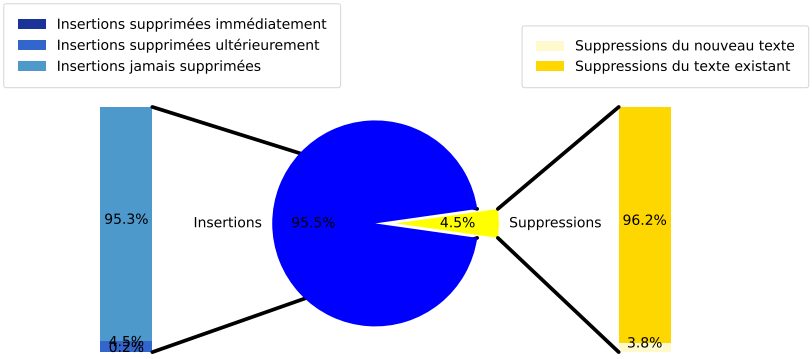


FIGURE 4.10 Diagramme de distributions.

de distribution (Figure **4.10**) des insertions et suppressions pour l'exemple 4.7 le montre, 95.5% des actions réalisées par le rédacteur sont des insertions et 4.5% sont des suppressions. Plus encore, 96.2% des suppressions concernent le texte écrit dans les communautés précédentes. Nous pouvons aussi voir que 95.3% des insertions restent jusqu'à la version finale du texte et que moins de 1% sont supprimés immédiatement. Ce qui signifie que, lorsque le rédacteur écrit quelque chose, il est plus probable que cela reste jusqu'à la fin.

En bref, le diagramme peut être utile pour comprendre comment le rédacteur révise son texte. Un rédacteur qui révise de manière très intensive aura par exemple un pourcentage plus élevé d'insertions supprimées (insertions supprimées immédiatement et/ou insertions supprimées ultérieurement). Lorsque le pourcentage d'insertions supprimées immédiatement est élevé, cela peut signifier que le rédacteur a effectué plus de révisions à la fin du texte courant. Lorsque le pourcentage d'insertions supprimées ultérieurement est élevé, cela peut signifier que le rédacteur a effectué plus de révisions sur le texte déjà existant.

4.3.2 Diagramme de contribution

Il est question ici d'évaluer la contribution de chaque communauté à la formation du texte. En effet, lorsqu'une insertion effectuée dans une communauté reste jusqu'à la fin, il fait partie intégrante du texte formé. Ainsi, nous évaluons la contribution en fonction du nombre d'insertions de la communauté présentes jusqu'à la fin.

Notons N le nombre de caractères du texte final et N_i le nombre de caractères du texte final issu de la $i^{ème}$ communauté, autrement dit N_i est le nombre de sommets d'insertion de type

I_3 de la communauté i.

$$N_i = |\{x \in V \text{ tel que } x \in C_i \cap I_3\}|$$

La contribution de la $i^{ème}$ communauté est donnée par :

$$CT_i = \frac{N_i}{N}\%$$

La contribution est donnée en termes de pourcentage du nombre de caractères présent dans le texte final résultant d'une modification donnée. Pour représenter cette contribution, on utilise un diagramme à bande, comme l'illustre la Figure **4.11** obtenue avec notre exemple.

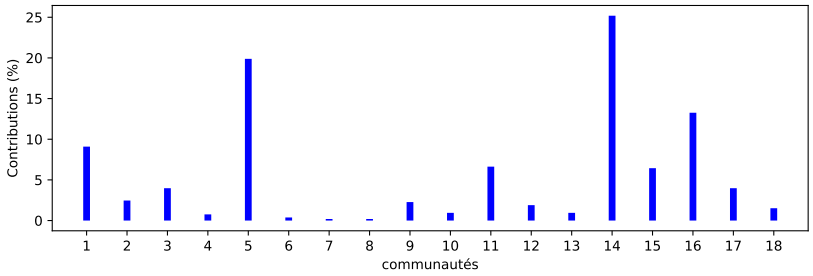


FIGURE 4.11 Diagramme de contribution.

On observe que près de 25% du texte final vient de la $14^{iéme}$ communauté, 20% vient de la $5^{iéme}$ et 15% vient de la $16^{ième}$. Les insertions réalisées dans ces trois communautés couvrent plus de 60% du texte final, alors que la contribution combinée des $4^{ième}$, $6^{ième}$,$7^{ième}$ et $8^{ième}$ communautés représente moins de 5%. En d'autres termes, 60% des caractères du texte proviennent de ces trois communautés et seulement 5% des caractères proviennent de ces quatre autres communautés.

En résumé, les diagrammes ainsi présentés permettent d'avoir accès à des informations un peu plus synthétiques.

4.4 Avantages et inconvénients des communautés de modifications

Nous avons défini la notion de communauté dans le but de permettre l'analyse des processus d'écriture, mais cette notion peut, dans certains cas, avoir ses limites.

Dans cette section, il est question de donner quelques avantages et inconvénients des communautés de modifications. La sous-section **4.4.1** est consacrée aux avantages et inconvénients liés à l'analyse du processus d'écriture. La sous-section 4.4.2 porte essentiellement sur les

avantages et inconvénients par rapport aux autres méthodes de détection de communautés.

4.4.1 Avantages et inconvénients pour l'analyse des processus d'écriture

Les communautés de modifications permettent de diviser le processus d'écriture en phases d'écriture qui pourront aider à l'analyse du processus d'écriture. Comme on peut le constater, la représentation qui découle des communautés permet de comprendre visuellement comment le texte a été écrit. De plus, avec les fusions des communautés, cette représentation permet d'observer les phases de révisions effectuées dans le texte. Plus encore, les diagrammes de distribution et de contribution permettent d'avoir accès à des informations à la fois structurée, synthétiques et riche en contenu. Dans un tout ordre d'idées, les communautés de modifications peuvent, avec certaines manipulations, être définies directement à partir du fichier d'enregistrement des frappes, ce qui fait des communautés de modifications une notion un peu plus générale pour l'étude des processus d'écriture.

Le principal inconvénient de notre définition de communauté de modifications réside dans le fait qu'on considère simplement trois types d'actions, ce qui peut limiter l'interprétation de nos communautés. En effet, en incluant d'autres types d'actions, tels que les copier-coller et les pauses, les communautés pourront être encore plus représentatives des processus d'écriture.

4.4.2 Avantages et inconvénients par rapport à d'autres méthodes de détection de communautés

Dans cette partie, nous comparons les communautés de modifications avec les communautés obtenues par d'autres méthodes connues, telles que la méthode de Clauset-Mewman-Moore (`MCNM`), la méthode de circulation des fluides (`MCF`), les k-cliques (`Mkcliques`) et la propagation de labels (`MPL`). Ces méthodes sont choisies, car elles ont quelques similitudes avec la méthode des communautés de modifications, notamment :

— la manière gloutonne d'ajouter les sommets dans les communautés (Clauset-Newman-Moore) ;
— la manière de circuler dans le graphe (circulation des fluides) ;
— la manière de considérer les voisinages d'un sommet (propagation de labels).

Pour faire les comparaisons avec la méthode des circulations des fluides, comme le nombre de communautés est fixé à l'avance, nous choisissons de fixer le nombre de communautés égal au nombre de communautés de modifications obtenu sur le même graphe. Pour effectuer les comparaisons, nous simulons 100 graphes de processus d'écriture de taille variant de 5 à 2000 sommets. La répartition du nombre de graphes par taille est donnée dans le tableau suivant :

Taille	5	10	25	50	100	250	500	1000	1500	2000
Nombre de Graphes	10	10	10	10	10	10	10	10	10	10

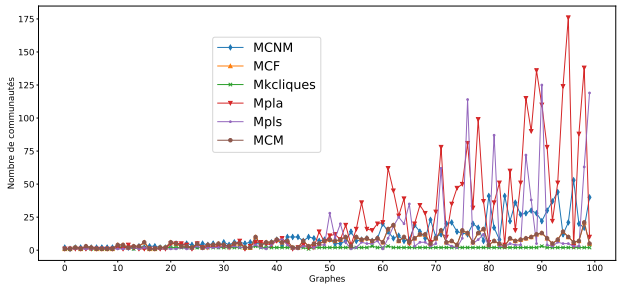
Dans un premier temps, nous comparons le nombre de communautés obtenu, la performance, la couverture et la modularité des méthodes. Ensuite, nous comparons la composition des communautés entre elles.

Premières comparaisons : Pour réaliser nos comparaisons, nous évaluons le nombre de communautés, la performance, la couverture et la modularité de chaque méthode sur chacun des graphes. Les tableaux **4.1** regroupent les résultats obtenus en termes de pourcentage du nombre de graphes où chaque méthode s'est démarquée. On remarque que, pour le nombre de communautés, la méthode de modularité de Clauset, Newman et Moores (`MCNM`) obtient le plus grand nombre de communautés dans 46% des graphes. Elle est suivie de la méthode de propagation de labels asynchrone (`MPLa`) avec 36%. La méthode des communautés, quant à elle, a eu dans 10% des cas un nombre élevé de communautés (en particulier pour des graphes de petites tailles) et dans 9% des cas un petit nombre de communautés. On remarque que la méthode des circulations de fluide (`MCF`) et la méthode des communautés de modifications (`MCM`) ont le même nombre de communautés. Ce qui est dû au fait que l'on a imposé que les deux méthodes aient le même nombre de communautés. Par contre, on note une différence entre les nombres d'arcs intercommunauté et les nombres d'arcs intracommunauté des deux méthodes, ce qui signifie que les communautés obtenues sont bien différentes. En termes de performance, la `MCM` est la seule avec une performance différente ; sinon, toutes les autres méthodes ont des performances identiques pour chacun des graphes et elles sont toutes supérieures à celle de la `MCM`. La méthode `MCM` a la plus petite couverture pour tous les graphes. Pour la modularité, la méthode `MCNM` obtient la valeur la plus élevée pour tous les graphes, ce qui est tout à fait normal, car `MCNM` est basée sur la maximisation de la modularité. Par contre, la méthode des k-cliques a la moins bonne modularité dans 88% des cas ; dans certains de ses cas, la modularité est même négative.

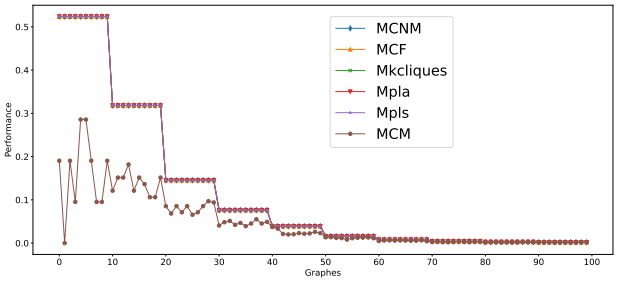
En résumé, la méthode des communautés de modification produit des communautés en nombre parfois égal aux autres méthodes dans certains graphes, mais avec très peu de couverture, de performance et de modularité par rapport aux autres méthodes. Ce qui laisse présager que les partitions en ensemble de sommets obtenues par la méthode `MCM` sont bien différentes des partitions des autres méthodes. Pour nous rassurer de ce résultat, nous effectuons une autre comparaison.

TABLEAU 4.1 Pourcentage de graphes pour lesquels chaque méthode a obtenu le meilleur score.

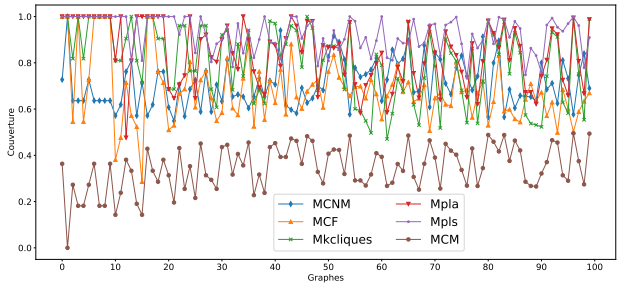
Maximum (%)						
Méthodes	n_c	e_{intra}	e_{inter}	**Performance**	**Couverture**	**Modularité**
MCNM	46	7	25	100	7	100
MCF	10	8	37	100	8	0
Mkcliques	0	10	2	100	10	0
MPLa	37	21	1	100	21	0
MPLs	7	54	0	100	54	0
MCM	10	0	35	0	0	0
Minimum (%)						
Méthodes	n_c	e_{intra}	e_{inter}	**Performance**	**Couverture**	**Modularité**
MCNM	2	0	7	0	0	1
MCF	9	0	8	0	0	9
Mkcliques	70	0	10	0	0	88
MPLa	10	0	21	0	0	8
MPLs	9	0	54	0	0	10
MCM	9	100	0	100	100	15



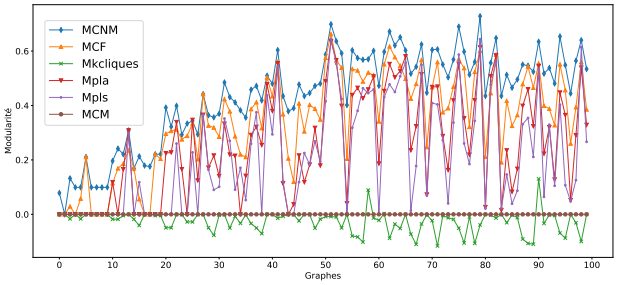
(a) Nombre de communautés



(b) Performances



(c) Couvertures



(d) Modularité

FIGURE 4.12 Comparaisons des méthodes suivant chaque critère.

Deuxièmes comparaisons : Ici, nous comparons la concordance des communautés issues des autres méthodes à celles issues de notre méthode. Pour cela, on considère toutes les paires de sommets possibles et on évalue la proportion des paires qui sont à la fois dans une même communauté pour les deux méthodes (la méthode des communautés de modification et la méthode considérée) ou qui ne sont pas dans une même communauté pour les deux

méthodes. Les résultats obtenus sont présentés à la Figure **4.13**. Les graphes sont classés par ordre croissant suivant leur taille. On remarque que les communautés obtenues par les autres méthodes concordent quelques fois avec celles de la méthode `MCM`, c'est le cas dans les graphes de petite taille, mais sinon, dans la majeure partie des graphes, la concordance est différente de 1. Ainsi, la `MCM` obtient des partitions, en ensembles de sommets, bien différentes des autres méthodes.

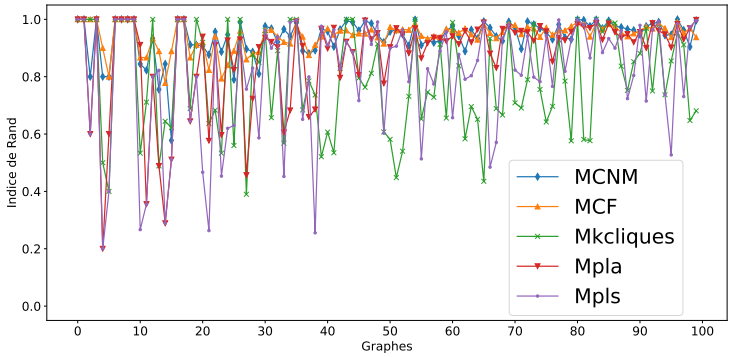


FIGURE 4.13 Mesure de concordance des communautés des différentes méthodes avec les communautés de modifications.

En résumé, la méthode `MCM` fournit des partitions en ensembles de sommets tout à fait différentes des partitions obtenues par les autres méthodes. Ce qui veut dire que les règles que nous avons définies ne sont pas prises en compte par les autres méthodes de détection de communautés.

4.5 Conclusion

Dans ce chapitre, nous avons défini la notion de « communauté de modifications », aussi appelée « épisode de modifications », ce qui permet de diviser le processus d'écriture en épisodes. Nous avons également présenté comment représenter ces communautés afin de comprendre visuellement la manière dont un rédacteur écrit et révise son texte. Cette méthode de visualisation permet de voir l'étendue des changements effectués dans chaque communauté (épisode) de modifications, ainsi que la position de la modification dans le texte et sa composition (insertion et suppression). De plus, la fusion et les diagrammes additionnels permettent de synthétiser l'information contenue dans le processus d'écriture. Nous avons également remarqué qu'en plus de permettre l'analyse du processus d'écriture, cette méthode produit parfois des partitions en ensembles de sommets semblables à celles obtenues par d'autres méthodes (surtout sur les graphes de petite taille). Toutefois, en général, ces partitions sont

nettement différentes, ce qui permet d'avoir une autre vision du graphe du processus d'écriture par rapport à celle qu'on pourrait avoir en utilisant les méthodes de partitionnement de graphe existantes. Ce qui veut dire alors que nos critères de communauté sont plus pertinents que les autres critères pour les graphes de processus d'écriture.

Les sommets d'une même communauté sont censés faire partie d'une même modification ; autrement dit, les sommets de communautés différentes font partie de modifications distinctes. Il est alors possible d'avoir des sommets ayant des voisins dans plusieurs communautés différentes. On peut dès lors se demander s'il existe des sommets plus importants que les autres dans le graphe du processus d'écriture et à quoi ils peuvent faire référence dans le texte, ce qui fera l'objet du chapitre suivant.

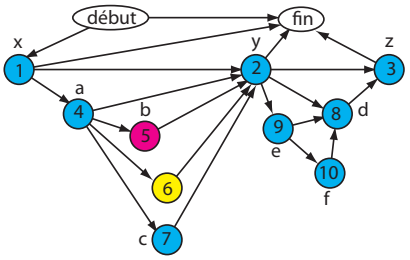
CHAPITRE 5 INDICE DE MODIFICATIONS

Ce chapitre présente la notion d'indice de modification qui renseigne sur le nombre de modifications effectuées en un endroit précis du texte. En effet, lorsque nous écrivons un texte, nous sommes parfois amenés à revenir plusieurs fois à une même partie de ce texte. L'indice nous permettra d'avoir une idée des places les plus modifiées. Certes, certains outils permettent d'avoir une idée des modifications, mais cet indice se veut être un chiffre tout simple qui donne une importance à un endroit suivant la quantité de modifications effectuées. L'indice de modification est calculé à partir du graphe de processus d'écriture issu du *modèle sans perte*. Il permet de donner de l'influence à un sommet suivant le nombre de fois que des modifications sont effectuées aux alentours du sommet. Pour définir l'indice de modification, nous aurons besoin de la notion de voisinage d'un sommet pour localiser l'endroit précis où a lieu la modification. Nous aurons également besoin de repréciser la notion de modification en ajoutant de nouveaux éléments.

Nous commençons le chapitre avec la section 5.1 qui donne plusieurs définitions de voisinages qui sont des ensembles de sommets correspondants à des insertions ou suppressions effectuées proches du sommet considéré. Ensuite, dans la section 5.2, ces voisinages seront partitionnés en blocs de modifications correspondant à un ensemble d'actions qu'on peut considérer comme formant une seule modification. Contrairement au chapitre précédent où les modifications étaient essentiellement des actions réalisées successivement sans déplacer le curseur, dans ce chapitre, le curseur peut être déplacé lors d'une modification, mais sous certaines conditions. Ainsi, nous donnerons plusieurs définitions d'une modification suivant les suites d'actions considérées en fonction du mouvement du curseur ou de la chronologie, ou encore une combinaison du mouvement du curseur et de la chronologie. Puis, dans la section 5.3, nous présenterons deux modèles issus du *modèle sans perte* qui permettront de calculer l'indice de modification. Enfin, dans la section 5.4, nous donnons les étapes de calcul de l'indice et un exemple illustratif. Pour débuter, définissons le graphe de processus d'écriture que nous utiliserons durant ce chapitre.

Pour rappel, un *graphe de processus d'écriture* G est défini par un ensemble $W = \{\text{début,fin}\} \cup V$ de sommets avec $V = \{1, 2, 3, ..., n\}$, par un ensemble A d'arcs reliant ces sommets et par une fonction $\mathcal{C} : V \longrightarrow \{Bleu, Jaune, Rouge\}$ qui attribue l'une des trois couleurs à chaque sommet de V. Le graphe contient l'arc (début,fin) ainsi que deux arcs (x, v) et (v, y) avec $x \in \{\text{début}\} \cup V$ et $y \in \{\text{fin}\} \cup V$ pour tout $v \in V$. Le graphe a donc exactement $2|V| + 1$ arcs. Chaque caractère écrit dans le texte est représenté par un sommet.

Par exemple, pour le processus d'écriture de la figure **5.1b**, nous obtenons le graphe G de la figure **5.1a**.



(a) Graphe *modèle sans perte*

T_1 : '**xyz**'

T_2 : '*x***ab***yz*'

T_3 : '*xa***b̶c***yz*'

T_4 : '*xacy***d***z*'

T_5 : '*xacy***ef***dz*'

Texte : '*xacyefdz*'

(b) Processus d'écriture

FIGURE 5.1 Graphe de processus d'écriture.

Après avoir écrit « *xyz* », les lettres « *a* » et « *b* » sont rajoutées avant « *y* » pour obtenir « *xabyz* », puis la lettre « *b* » est supprimée et remplacée par la lettre « *c* »pour obtenir « *xacyz* ». Ensuite, la lettre « *d* » est rajoutée après « *y* pour obtenir « *xacydz* et, finalement les lettres « *e* » et « *f* » sont rajoutées avant « *d* » pour obtenir « *xacyefdz* ».

Les voisinages des sommets seront déterminés essentiellement dans le graphe de processus d'écriture.

5.1 Voisinages d'un sommet

Cette section a pour objectif principal de définir les voisinages qui seront considérés dans l'évaluation de l'indice de modification. Nous présenterons trois voisinages qui regrouperont les insertions ou suppressions effectuées respectivement à la position tout juste avant, après ou autour du sommet considéré.

Dans la sous-section 5.1.1, il sera question de présenter le voisinage entrant d'un sommet qui regroupe les sommets d'insertion ou de suppression effectuées à la position tout juste avant le caractère du sommet considéré. Ensuite, la sous-section 5.1.2 portera sur le voisinage sortant d'un sommet, qui est un ensemble de sommets correspondant à des insertions ou des suppressions réalisées à la position tout juste après le caractère du sommet considéré. Et enfin, la sous-section 5.1.3 présente le voisinage complet d'un sommet qui est une réunion du voisinage entrant et sortant. Pour conclure la section, nous présenterons, à la sous-section 5.1.4, les avantages et les inconvénients de chacun des voisinages définis. Avant toute chose, voici quelques notions, principes et notations qui seront utilisés dans cette section.

Pour rappel, les sommets `Début` et `Fin` ont les numéros -1 et 0 respectivement.

La *distance* $d(u, v)$ entre deux sommets d'un graphe orienté est le plus petit nombre d'arcs sur un chemin reliant u à v dans ce graphe.

Pour un sommet $v \in V$, notons G_v le sous-graphe de G induit par les sommets $x \geqslant v$.

Notons $d^{in}(v)$ le nombre d'arcs entrant en v et $d^{out}(v)$ le nombre d'arcs sortant de v dans le graphe G. Dans ce qui suit, pour chaque sommet $v \in V$, nous nous intéresserons à l'ensemble $N^{in}(v)$ contenant les prédécesseurs immédiats x de v avec $x > v$ ainsi qu'à l'ensemble $N^{out}(v)$ contenant les successeurs immédiats y de v avec $y > v$. Comme chaque sommet $v \in V$ a exactement un prédécesseur immédiat x tel que $x < v$ et un successeur immédiat y tel que $y < v$, on déduit que $|N^{in}(v)| = d^{in}(v) - 1$ et $|N^{out}(v)| = d^{out}(v) - 1$. En d'autres termes, $|N^{in}(v)|$ correspond au nombre de fois qu'un caractère a été supprimé ou ajouté juste avant le caractère correspondant au sommet v alors que $|N^{out}(v)|$ correspond au nombre de fois qu'un caractère a été supprimé ou ajouté juste après le caractère correspondant au sommet v.

Exemple : Si on s'intéresse au sommet $v = 2$ du graphe de la Figure **5.1a**, on a :

— $d^{in}(2) = 5$, $d^{out}(2) = 4$

— $N^{in}(2) = \{4, 5, 6, 7\}$, $N^{out}(2) = \{3, 8, 9\}$. Le sommet 1 n'est pas dans $N^{in}(2)$ par définition de N^{in} car $1 < 2$.

Ainsi, on a 4 caractères (insérés et supprimés) avant et 3 caractères après le caractère « y » du sommet $v = 2$.

Pour rappel, notre objectif est de définir un indice qui renseigne sur le nombre de modifications (insertion ou suppression) effectuées aux alentours d'un caractère du texte. Mais jusqu'à quel point pouvons-nous dire qu'une insertion ou suppression est proche d'une autre ? En effet, si dans l'exemple précédent on s'intéresse aux modifications proches du caractère *'y'*, on peut n'y voir que trois modifications :

— l'une consistant à écrire *'z'* après *'y'*,

— l'une consistant à écrire *'ac'* avant *'y'* (en incluant l'insertion, puis la suppression du caractère *'b'*),

— l'autre consistant à écrire *'efd'* après *'y'*.

Une autre vision pourrait y voir 7 modifications, à savoir

— l'écriture de *'z'* après *'y'*,

— l'écriture de *'a'* avant *'y'*,

— l'écriture de *'b'* avant *'y'*,

— la suppression de *'b'* avant *'y'*,

— le rajout de *'c'* avant *'y'*,

— le rajout de *'d'* après *'y'*, et

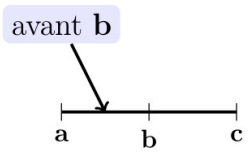
— le rajout de *'e'* après *'y'*.

Remarquons que, dans ce cas, le rajout de *'f'* n'est pas considéré comme une modification proche de *'y'*, car cette insertion est faite entre *'e'* et *'d'*. Il nous faut donc donner une définition de ce que nous entendons par proche, cela revient à préciser le *voisinage* du sommet correspondant au caractère considéré.

Dans un premier temps, intéressons-nous à ce qui se passe en avant d'un caractère.

5.1.1 Voisinage entrant d'un sommet dans un graphe de processus d'écriture

Ici, on se situe dans l'espace situé avant un caractère dans un texte, comme illustré ci-dessous.



Ainsi, dans cette sous-section, on définit une manière de regrouper toutes les actions réalisées avant un caractère précis dans le texte. Pour ce faire, on utilise la structure du graphe de processus d'écriture pour définir un ensemble de sommets appelé voisinage entrant qui correspond aux différentes actions réalisées dans cet espace.

Pour $i \geq 1$ et $v \in V$, nous définissons $N_i^{in}(v)$ suivant le type du sommet v.

— Si v n'est pas un sommet supprimé, alors on définit $N_i^{in}(v)$ comme l'ensemble des sommets $x > v$ de V tel que $d(x, v) = i$ dans G_v.

— Si v est supprimé, soit w le sommet de suppression de v, alors on définit $N_i^{in}(v)$ comme l'ensemble des sommets $x > v$ et $x < w$ de V tel que $d(x, v) = i$ dans G_v.

Intuitivement, on peut définir le voisinage entrant comme $\bigcup_{i=1}^{k} N_i^{in}(v)$, qui est l'ensemble des sommets de numéro plus grand que v, pour lesquels il existe un chemin d'au plus k arcs les menant à v. En considérant cette définition, pour les sommets 2 et 3 de l'exemple de la figure **5.1a** on a :

Pour $i = 1$ **:** $N_1^{in}(2) = \{4, 5, 6, 7\}$, $N_1^{in}(3) = \{8\}$. Le sommet 1 n'est pas dans $N^{in}(2)$ par définition de N^{in} car $1 < 2$.

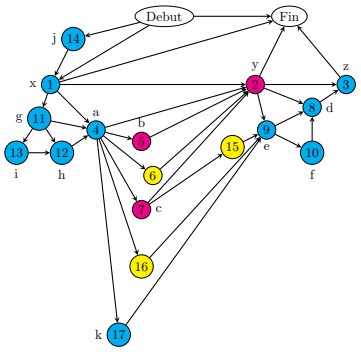
Pour $i = 2$ **:** $N_2^{in}(3) = \{9, 10\}$. Le sommet 1 n'est pas dans $N_2^{in}(3)$ car $1 < 3$.

D'où

$$\bigcup_{i=1}^{1} N_i^{in}(2) = \{4, 5, 6, 7\}$$

$$\bigcup_{i=1}^{2} N_i^{in}(3) = \{8, 9, 10\}$$

Augmentons maintenant la taille du graphe précédent en effectuant quelques insertions et suppressions supplémentaires. Les insertions de *'gh'* entre *'x'* et *'a'*, *'i'* entre *'g'* et *'h'*, *'j'* au début avant *'x'*, puis les suppressions de *'y'* et *'c'*, et en remplaçant *'c'* par *'k'*.



(a) Graphe *modèle sans perte*

T_1 : 'xyz'

T_2 : '*x***ab***yz*'

T_3 : '*xa*~~**b**~~**c***yz*'

T_4 : '*xacy***d***z*'

T_5 : '*xacy***ef***dz*'

T_6 : '*x***gh***acyefdz*'

T_7 : '*xg***i***hacyefdz*'

T_8 : '**j***xgihacyefdz*'

T_9 : '*jxgiha*~~*c*~~*yefdz*'

T_{10} : '*jxgiha***k***efdz*'

(b) Processus d'écriture

FIGURE 5.2 Graphe de processus d'écriture.

Si on s'intéresse au sommet $v = 3$ du graphe **(5.2a)**, on a $N_1^{in}(3) = \{8\}$, $N_2^{in}(3) = \{9, 10\}$, $N_3^{in}(3) = \{15, 16, 17\}$, $N_4^{in}(3) = \{7, 4\}$, $N_5^{in}(3) = \{11, 12\}$, $N_6^{in}(3) = \{13\}$. Ainsi

$$\bigcup_{i=1}^{6} N_i^{in}(3) = \{4, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17\}.$$

Pour le sommet $v = 2$, $N_1^{in}(2) = \{4, 5, 6, 7\}$, $N_2^{in}(2) = \{11, 12\}$, $N_3^{in}(2) = \{13\}$ et

$$\bigcup_{i=1}^{3} N_i^{in}(2) = \{4, 5, 6, 7, 11, 12, 13\}.$$

On remarque alors que certains sommets de $\bigcup_{i=1}^{3} N_i^{in}(2)$ sont dans $\bigcup_{i=1}^{6} N_i^{in}(3)$. En effet, $\bigcup_{i=1}^{3} N_i^{in}(2) \bigcap \bigcup_{i=1}^{6} N_i^{in}(3) = \{4, 7, 11, 12, 13\}$, on retrouve des insertions réalisées au voisi-

nage du sommet 2 avant sa suppression. Car en supprimant le sommet 2, certains sommets de son voisinage se sont reliés à ceux du sommet 3. En effet, lorsque le caractère *'y'* du sommet 2 n'est pas encore supprimé, on a quelque chose qui ressemble dans la figure **5.3a**. On distingue alors ce qui est dans l'espace avant *'y'* et avant *'z'*. Mais lorsque le caractère *'y'* du sommet 2 est supprimé, les deux espaces se relient, comme illustré sur la figure **5.3b**. Puisque techniquement, la partie en rouge a été écrite bien avant que *'y'* ne soit supprimé et qu'avant cette suppression, c'était le voisinage de *'y'* sommet 2 non celui de *'z'* sommet 3, alors il serait judicieux de faire cette distinction lors de la définition des voisinages des sommets.

(a) y pas encore supprimé (b) y supprimé

FIGURE 5.3 Voisinage entrant.

Pour distinguer les deux voisinages, on peut prendre comme voisinage entrant de 3 les sommets de $\bigcup_{i=1}^{i=6} N_i^{in}(3)$ qui n'appartiennent pas à $\bigcup_{i=1}^{i=3} N_i^{in}(2)$, alors on se retrouve avec les sommets $\{8, 9, 10, 15, 16, 17\}$ qui sont les voisins de 3.

Plus formellement, on considère la définition qui suit.

Définition 5.1.1 (**Voisinage entrant d'un sommet dans un graphe de processus d'écriture**)**.** Soit v un sommet, posons $\boldsymbol{\Psi_v^{in,k}}$ le voisinage entrant de v d'ordre k.

- Si v n'a pas de prédécesseur immédiat w tel que $w > v$, alors $\Psi_v^{in,k} = \emptyset$ et $|\Psi_v^{in,k}| = 0$.
- Si v a un prédécesseur immédiat, posons u le prédécesseur tel que $u < v$:
  - ∗ Si u est supprimé, alors :

$$\Psi_v^{in,k} = \bigcup_{i=1}^{k} N_i^{in}(v) / \bigcup_{i=1}^{\infty} N_i^{in}(u)$$

    l'ensemble des sommets $x > v$ tels que $d(x, v) \leqslant k$ dans G_v et $d(x, u) = \infty$ dans G_u.
  - ∗ Si u n'est pas supprimé, alors :

$$\Psi_v^{in,k} = \bigcup_{i=1}^{k} N_i^{in}(v)$$

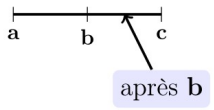
l'ensemble des sommets $x > v$ tels que $d(x, v) \leqslant k$ dans G_v.

Pour $k = \infty$, $\boldsymbol{\Psi_v^{in} = \Psi_v^{in,\infty}}$ est le voisinage entrant complet du sommet v.

En appliquant la définition à l'exemple précédent, on a : $\Psi_2^{in} = \{4, 5, 6, 7, 11, 12, 13\}$ et $\Psi_3^{in} = \{8, 9, 10, 15, 16, 17\}$

5.1.2 Voisinage sortant d'un sommet dans un graphe de processus d'écriture

Dans cette sous-section, comme représenté ci-dessous, on se situe dans l'espace situé après un caractère dans un texte.



On définit alors un voisinage sortant qui regroupe les sommets qui correspondent aux différentes actions réalisées dans cet espace. Pour $i \geq 1$ et $v \in V$, définissons $N_i^{out}(v)$ comme l'ensemble des sommets $x > v$ de V tels que $d(v, x) = i$ dans G_v, on a $N_1^{out}(v) = N^{out}(v)$.

On considère alors $\bigcup_{i=1}^{k} N_i^{out}(v)$ qui est l'ensemble des sommets x de numéro plus grand que v, tels qu'il existe un chemin d'au plus k arcs allant de v vers x.

Pour le sommet $v = 2$ de l'exemple de la figure **5.2a** on a :

$i = 1$, $N_1^{out}(2) = \{3, 8, 9\}$

$i = 2$, $N_2^{out}(2) = \{10\}$, en effet, il existe un chemin $(2, 9, 8)$ reliant 2 à 8 mais on prend le plus court chemin qui est $(2, 8)$ de ce fait $8 \notin N_2^{out}(2)$

D'où

$$\bigcup_{i=1}^{2} N_i^{out}(2) = \{3, 8, 9, 10\}$$

Pour le sommet $v = 1$ de l'exemple de la Figure **5.2a** on a :

$i = 1$, $N_1^{out}(1) = \{2, 4, 11\}$

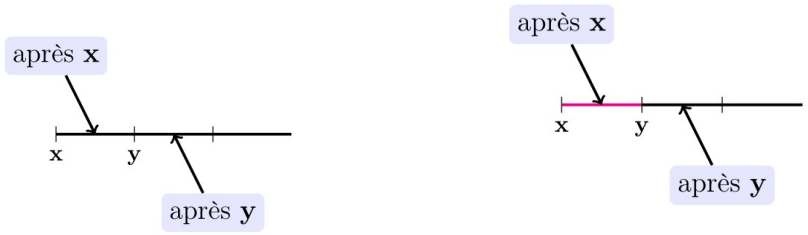
$i = 2$, $N_2^{out}(1) = \{3, 5, 6, 7, 8, 9, 12, 13, 16, 17\}$

$i = 3$, $N_3^{out}(1) = \{10, 15\}$

Ainsi

$$\bigcup_{i=1}^{3} N_i^{out}(1) = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17\}$$

On remarque que, $\bigcup_{i=1}^{3} N_i^{out}(1)$ regroupe tous les sommets d'insertion et suppressions effectuées après le sommet 1. Au fond, c'est tout ce qui est écrit entre le caractère *'x'* du sommet 1 et la fin du texte, comme illustré sur la figure **5.4a**, on a ce qui est écrit aussi après le caractère *'y'* du sommet 2.



(a) (b)

FIGURE 5.4 Voisinage sortant.

Or ce que l'on veut, c'est se restreindre à l'espace tout juste après le caractère *'x'* du sommet 1, la partie rouge de la figure **5.4b**. Pour distinguer cette partie du voisinage, on peut prendre comme voisinage sortant du sommet 1, les sommets de $\bigcup_{i=1}^{3} N_i^{out}(1)$ qui se trouvent dans le voisinage entrant de 2. Pour ce faire, nous définissons $N_i^{out,u}(v)$ un ensemble des sommets qui dépend du type du sommet v et du type du sommet u successeur de v.

* Si v n'est pas un sommet supprimé. Pour tout $u \in N^{out}(v)$:
  - Si u est un sommet d'insertion non supprimé : $N_i^{out,u}(v)$ est défini comme l'ensemble des sommets $x > v$ de Ψ_u^{in} tels que $d(v, x) = i$ dans G_v.
  - Si u est un sommet d'insertion supprimé. Soit y le sommet qui supprime u. Soit z le successeur du sommet y : $N_i^{out,u}(v)$ est défini comme l'ensemble des sommets $x > v$ de Ψ_u^{in} ou $x > y$ de Ψ_z^{in} tels que $d(v, x) = i$ dans G_v.
  - Si u est une suppression, soit y le successeur de u : $N_i^{out,u}(v)$ est défini comme l'ensemble des sommets $x > u$ de Ψ_y^{in} tels que $d(v, x) = i$ dans G_v.
* Si v est un sommet supprimé, soit w le sommet de suppression de v. Pour tout $u \in N^{out}(v)$:
  - Si u est un sommet d'insertion non supprimé : $N_i^{out,u}(v)$ est défini comme l'ensemble des sommets $x > v$ et $x < w$ de Ψ_u^{in} tels que $d(v, x) = i$ dans G_v.

- Si u est un sommet d'insertion supprimé. Soit y le sommet qui supprime u. Soit z le successeur du sommet y : $N_i^{out,u}(v)$ est défini comme l'ensemble des sommets x, $w > x > v$ de Ψ_u^{in} ou $w > x > y$ de Ψ_z^{in} tels que $d(v,x) = i$ dans G_v.
- Si u est une suppression, soit y le successeur de u : $N_i^{out,u}(v)$ est défini comme l'ensemble des sommets x, $w > x > u$ de Ψ_y^{in} tels que $d(v,x) = i$ dans G_v.

Définition 5.1.2. On dit que le sommet x vérifie la propriété Out_i de v, si on peut trouver $u \in N^{out}(v)$ tel que $x \in N_i^{out,u}(v)$.

On définit alors le voisinage sortant d'un sommet comme suit.

Définition 5.1.3 (**Voisinage sortant d'un sommet dans un graphe de processus d'écriture**)**.** Soit v un sommet, posons $\boldsymbol{\Psi_v^{out,k}}$ le voisinage sortant de v d'ordre k.

- Si v n'a pas de successeur immédiat w tel que $w > v$ alors $\Psi_v^{out,k} = \emptyset$ et $|\Psi_v^{out,k}| = 0$
- Sinon, posons u le plus petit successeur de v tel que $u > v$

$$\Psi_v^{out,k} = \bigcup_{i=1}^{k} \left(\bigcup_{u \in N^{out}(v)} N_i^{out,u}(v) \right)$$

l'ensemble des sommets vérifiant au moins une propriété Out_i, $i \leqslant k$

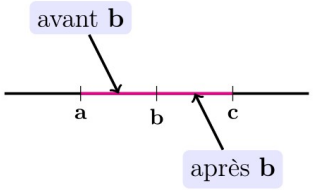
Pour $k = \infty$, $\boldsymbol{\Psi_v^{out}} = \boldsymbol{\Psi_v^{out,\infty}}$ est le voisinage sortant complet du sommet v.

En appliquant à l'exemple précédent, on a $\Psi_1^{out} = \{4, 5, 6, 7, 11, 12, 13\}$ et $\Psi_2^{out} = \{8, 9, 10\}$

Remarque 5.1.1. *Dans le cas où l'on s'intéresse à toutes les insertions et suppressions réalisées entre un caractère et la fin du texte, on peut alors utiliser, comme voisinage sortant d'ordre k du sommet v l'ensemble, $\bigcup_{i=1}^{k} N_i^{out}(v)$, des sommets $x > v$ tels que $d(x,v) \leqslant k$ dans G_v.*

5.1.3 Voisinage d'un sommet dans un graphe de processus d'écriture

Ici, on s'intéresse à ce qui se passe dans l'espace situé à la fois avant et après un caractère dans un texte.



En effet, le voisinage du sommet regroupe les sommets des actions réalisées aux alentours du caractère du sommet considéré. Plus formellement, on a la définition qui suit.

Définition 5.1.4. (Voisinage d'un sommet dans un graphe de processus d'écriture). Soit v un sommet, posons $\boldsymbol{\Psi}_v^k$ le voisinage de v d'ordre k.

$$\boldsymbol{\Psi}_v^k = \boldsymbol{\Psi}_v^{in,k} \cup \boldsymbol{\Psi}_v^{out,k}$$

Pour $k = \infty$, $\boldsymbol{\Psi}_v = \boldsymbol{\Psi}_v^{\infty}$ est le voisinage complet du sommet v.

5.1.4 Avantages et inconvénients des différents voisinages

Cette sous-section donne les avantages et les inconvénients de chacun des voisinages. De plus, elle donne une brève comparaison des différentes définitions de voisinage.

Nous avons présenté trois types de voisinage : un voisinage entrant, un voisinage sortant et un voisinage qui est une réunion des deux autres voisinages (sortant et entrant).

Pour le voisinage entrant, nous avons retenu la définition qui prend en compte deux façons de voir le voisinage entrant d'un sommet selon que le prédécesseur du sommet soit supprimé ou non. Cette manière de faire permet de distinguer les modifications relatives à chaque sommet en tenant compte de ce qui se passe autour de lui réellement. L'avantage de ce voisinage, c'est qu'il traduit à la fois la proximité des modifications, mais aussi la chronologie dans le voisinage. En effet, en séparant le voisinage d'un sommet de celui de son prédécesseur lorsque ce dernier est supprimé, on est certain d'avoir des modifications qui sont chronologiques et spatialement liées au caractère du sommet considéré. De plus, avec cette définition, lorsqu'un sommet est supprimé, son voisinage n'augmente plus. Ainsi, il n'y a aucun risque que l'on compte parmi le voisinage entrant d'un sommet les sommets insérés après sa suppression. Ce qui traduit bien le fait que, lorsqu'un caractère est supprimé, il n'y a plus de trace de lui dans le texte.

Pour le voisinage sortant, nous avons retenu la définition qui prend en compte le voisinage entrant du successeur du sommet considéré, en fonction du type du sommet et de son successeur. En effet, en prenant le voisinage sortant ainsi, on se retrouve avec les sommets qui traduisent les insertions effectuées tout juste à côté du caractère du sommet considéré. Ainsi, on est certain que, lorsqu'un sommet appartient au voisinage, alors il a été relié à un moment donné au successeur du sommet considéré. De plus, en considérant le voisinage sortant ainsi, on a dans certains cas l'égalité entre le voisinage sortant du sommet considéré et le voisinage

entrant de son successeur. En d'autres termes, pour certains sommets v, $\Psi_v^{out} = \Psi_u^{in}$ où u est le plus petit successeur du sommet v tel que $u > v$. Ce qui permet de diminuer le calcul des voisinages.

En utilisant la réunion de ces deux voisinages, on obtient un voisinage qui regroupe les sommets des insertions effectuées dans l'espace autour du caractère considéré. De plus, il hérite des avantages et des inconvénients des autres voisinages. Ce qui permet, par exemple, dans certains cas, de diminuer les temps de calculs des créations des voisinages. En effet, le fait de déterminer le voisinage entrant de chaque sommet permet d'obtenir les deux autres voisinages. Ainsi, le voisinage primordial à déterminer est le voisinage entrant, car c'est lui qui intervient dans les deux autres voisinages. Ce qui traduit bien le fait que, lorsqu'on modifie un texte, l'on revient sur ce que l'on a déjà écrit et, en général, on se place devant un caractère déjà écrit.

En résumé, les trois voisinages sont étroitement liés, bien qu'ils traduisent des situations différentes.

Maintenant que nous avons nos différents voisinages, il nous faut déterminer comment les diviser pour qu'ils puissent refléter réellement les séquences de modifications effectuées. C'est l'objectif de la section suivante.

5.2 Modifications

Dans cette section, il est question de donner une définition des modifications. En général, une modification est considérée comme une suite d'insertions et/ou de suppressions réalisées à une position donnée dans le texte. Puisque dans le graphe de processus d'écriture, les arcs traduisent le lien spatial (position) et les numéros de sommets traduisent le lien chronologique (succession), on peut alors considérer qu'une modification du texte est un chemin ou une chaîne dans le graphe de processus d'écriture. Ainsi, nous pourrons définir de plusieurs façons différentes les chemins et les chaînes suivant la vision que l'on a d'une modification. Nous présentons dans cette section quatre manières différentes de voir une modification.

La section 5.2.1 présente une modification comme une succession chronologique des insertions et/ou des suppressions. La section 5.2.2 présente deux visions où une modification est vue comme une succession d'actions à la fois spatiales et chronologiques. À la section 5.2.3, une modification un peu plus générale est définie : elle prend en compte, en plus de la spatialité et de la chronologie, la distance entre les insertions et/ou les suppressions effectuées. Enfin,

la section 5.2.4 fait ressortir les inconvénients et avantages des différentes définitions de modification.

5.2.1 Modification chronologique

Ici, une modification est essentiellement considérée comme une succession chronologique des insertions et/ou des suppressions.

Formellement, dans un graphe de processus d'écriture, une modification chronologique peut être définie comme suit :

Définition 5.2.1. (Modification chronologique). Une *modification chronologique* d'origine x et de longueur ℓ est le sous-ensemble $\{x, x+1, \ldots, x+\ell-1\}$. Alors que les arcs (et donc la spatialité) ne sont pas pris en compte, la chronologie est assurée par la succession des numéros des sommets.

Par exemple, dans le cas du graphe de la figure **5.2a**, on a pour rappel :

$\Psi_2^{in} = \{4, 5, 6, 7, 11, 12, 13\}$ et $\Psi_2^{out} = \{8, 9, 10\}$ pour le sommet 2

$\Psi_3^{in} = \{8, 9, 10, 15, 16, 17\}$ et $\Psi_3^{out} = \emptyset$ pour le sommet 3

On obtient alors les modifications chronologiques suivantes au voisinage du sommet 2 et 3 :

- $\{4, 5, 6, 7\}$ et $\{11, 12, 13\}$ avant 2,
- $\{8, 9, 10\}$ après 2.
- $\{8, 9, 10\}$ et $\{15, 16, 17\}$ avant 3 et aucune modification après 3

Avec cette manière de voir les modifications, la spatialité n'est pas prise en compte, tout ce qui compte est le fait que les événements d'insertions et de suppressions se suivent dans le temps. Si on considère par exemple le voisinage au complet du sommet 2,

$$\Psi_2 = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}.$$

On a alors une seule modification, même si certains sommets sont éloignés les uns des autres dans le graphe. Autrement dit, dans un texte, deux insertions peuvent être chronologiquement successives, mais se retrouver dans des positions éloignées l'une de l'autre. C'est le cas des insertions de *'z'* et *'g'* : elles sont respectivement à la fin et au début du texte, mais elles se retrouvent dans la même modification à cause de la succession chronologique. Ce qui peut être un inconvénient pour des études qui considèrent les modifications à une position précise du

texte. Dans les sections suivantes, nous présentons d'autres manières possibles de considérer une modification pour remédier à ce manque.

5.2.2 Modification spatiale avec seuil chronologique

Dans cette sous-section, en plus de considérer la succession chronologique des actions réalisées, on prend en compte la position des modifications dans la définition. Ainsi, il sera question d'introduire la présence des arcs entre les sommets, ce qui permet de définir des chemins qui représentent les modifications.

Définition 5.2.2. (Chemin de modification spatiale avec seuil chronologique). Un *chemin de modification spatiale de longueur ℓ, avec seuil chronologique $s \geq 1$* est un chemin $(x_1, x_2, x_3, ..., x_\ell)$ tel que, $x_i \in V$ $(i = 1, \ldots, \ell)$ et $|x_i - x_{i+1}| \leq s$ $(i = 1, \ldots, \ell - 1)$. La spatialité est prise en compte par les arcs, mais pour la chronologie, un seuil est fixé pour l'écart entre les numéros de deux sommets successifs du chemin.

Exemple : Dans le cas du graphe de la figure **5.2a**, si on s'intéresse maintenant aux chemins de modification spatiale avec seuil chronologique s on obtient :

Pour $s = 1$:

- les modifications $\{4, 5\}, \{6\}, \{7\}, \{11, 12\}, \{13\}$ avant le sommet 2
- les modifications $\{8\}, \{9, 10\}$ après le sommet 2
- les modifications $\{8\}, \{9, 10\}, \{15\}, \{16\}, \{17\}$ avant le sommet 3

Pour $s = 2$:

- les modifications $\{4, 5\}, \{6\}, \{7\}, \{11, 13, 12\}$ avant le sommet 2
- les modifications $\{9, 10, 8\}$ après le sommet 2
- les modifications $\{9, 10, 8\}, \{15\}, \{16\}, \{17\}$ avant le sommet 3

Dans le cas présent, les arcs indiquent la direction dans laquelle les sommets doivent être considérés. Ainsi, le fait de revenir en arrière d'un caractère peut être considéré comme une autre modification. C'est par exemple le cas des modifications $\{8\}$ et $\{9, 10\}$ après le sommet 2. Ces modifications sont successives chronologiquement, mais le fait que 9 soit inséré tout juste avant 8 coupe ce lien spatial. Certes, le fait d'augmenter le seuil chronologique à $s = 2$ a permis de réunir ces deux modifications en une seule dans ce cas-ci. Mais une autre manière de voir les choses serait d'oublier la direction des arcs. Il sera alors question de considérer les chaînes à la place des chemins. On obtient alors une autre définition de la modification qui prend aussi en compte la spatialité et la chronologie.

Définition 5.2.3. (Chaîne de modifications avec seuil chronologique). Une *chaîne de modifications de longueur ℓ, avec seuil chronologique* $s \geq 1$ est une chaîne $[x_1, x_2, x_3, ..., x_\ell]$ telle que, $x_i \in V$ $(i = 1, \ldots, \ell)$ et $|x_i - x_{i+1}| \leq s$ $(i = 1, \ldots, \ell - 1)$.

Par exemple, pour les sommets $v = 2$ et $v = 3$, en considérant la chaîne de modifications avec seuil chronologique s on a :

Pour $s = 1$:

- les modifications $\{4, 5\}, \{6\}, \{7\}, \{11, 12, 13\}$ avant le sommet 2
- les modifications $\{8, 9, 10\}$ après le sommet 2
- les modifications $\{8, 9, 10\}, \{15\}, \{16\}, \{17\}$ avant le sommet 3

Pour $s = 8$:

- les modifications $\{11, 13, 12, 4, 5\}, \{6\}, \{7\}$ avant le sommet 2
- les modifications $\{8, 9, 10\}$ après le sommet 2
- les modifications $\{15, 9, 10, 8\}, \{16\}, \{17\}$ avant le sommet 3

Jusqu'à présent, les définitions des modifications présentées permettent de prendre en compte la chronologie ou la spatialité et la chronologie à la fois, mais certaines spatialités ne sont pas forcément prises en compte : c'est le cas par exemple des sommets $5, 6$ et 7 qui ne se retrouvent pas dans la même modification, bien que ces sommets représentent des actions effectuées successivement à la même position dans le texte. En effet, *'b'* est supprimé et remplacé par *'c'*. Pour prendre en compte ce genre de modification, on peut considérer également la distance en termes d'arêtes entre les sommets d'un même voisinage. La sous-section 5.2.3 donne plus de détail à ce sujet.

5.2.3 Modification spatio-chronologique avec seuil chronologique

Cette sous-section présente une définition de modification qui prend en compte la distance entre les sommets d'un même voisinage, en plus de la spatialité et de la chronologie déjà considérées dans les autres méthodes. En définissant ainsi les modifications, on prend en considération certaines spatialités qui n'étaient pas forcément prises en compte dans les définitions précédentes (sous-sections 5.2.1 et 5.2.2). Pour introduire cette définition de modification, nous avons besoin de quelques notions supplémentaires.

Pour tout sommet $v \in V$, posons :

G_v^{Ψ} **:** le sous-graphe induit par le voisinage du sommet v.

$G_v^{\Psi^{in}}$ **:** le sous-graphe induit par le voisinage entrant du sommet v.

$G_v^{\Psi^{out}}$ **:** le sous-graphe induit par le voisinage sortant du sommet v.

Pour chaque sommet $x \in \Psi_v$ nous nous intéressons à l'ensemble $\Gamma^v(x)$ contenant les prédécesseurs immédiats et successeurs immédiats de x dans le voisinage de v.

$$\Gamma^v(x) = \{u \in \Psi_v / (u,x) \in A \ \texttt{ou} \ (x,u) \in A\}$$

Pour $j \geqslant 1$ et $x \in \Psi_v$, définissons $\Gamma_j^v(x)$ comme l'ensemble de sommets u de Ψ_v tel qu'il existe une chaîne sans corde $[u = y_0, ..., y_j = x]$ dans G_v^Ψ (l'orientation des arcs de G_v^Ψ n'est pas prise en compte). On a donc $\Gamma_1^v(x) = \Gamma^v(x)$.

$\bigcup_{j=1}^k \Gamma_j^v(x)$ est l'ensemble des sommets $u \in \Psi_v$ tel qu'il existe une chaîne dans G_v^Ψ de longueur au plus k entre u et v, constituée uniquement des sommets de Ψ_v.

Similairement, on peut définir les voisinages suivants.

Pour le voisinage entrant de v, on définit

$$\Gamma^{v,in}(x) = \{u \in \Psi_v^{in} / (u,x) \in A \ \texttt{ou} \ (x,u) \in A\}$$

Pour $j \geqslant 1$ et $x \in \Psi_v^{in}$, définissons $\Gamma_j^{v,in}(x)$ comme l'ensemble de sommets u de Ψ_v^{in} tel qu'il existe une chaîne sans corde $[u = y_0, ..., y_j = x]$ dans $G_v^{\Psi^{in}}$. On a donc $\Gamma_1^{v,in}(x) = \Gamma^{v,in}(x)$.

$\bigcup_{j=1}^k \Gamma_j^{v,in}(x)$ est l'ensemble des sommets $u \in \Psi_v^{in}$ tel qu'il existe une chaîne d'au plus k arêtes entre u et v dans $G_v^{\Psi^{in}}$.

Pour le voisinage sortant de v, on définit

$$\Gamma^{v,out}(x) = \{u \in \Psi_v^{out} / (u,x) \in A \ \texttt{ou} \ (x,u) \in A\}$$

Pour $j \geqslant 1$ et $x \in \Psi_v^{out}$, définissons $\Gamma_j^{v,out}(x)$ comme l'ensemble de sommets u de Ψ_v^{out} tel qu'il existe une chaîne sans corde $[u = y_0, ..., y_j = x]$ dans $G_v^{\Psi^{out}}$. On a donc $\Gamma_1^{v,out}(x) = \Gamma^{v,out}(x)$.

$\bigcup_{j=1}^k \Gamma_j^{v,out}(x)$ est l'ensemble des sommets $u \in \Psi_v^{out}$ tel qu'il existe une chaîne d'au plus k arêtes entre u et v dans $G_v^{\Psi^{out}}$.

Définition 5.2.4. (modification spatio-chronologique avec seuil chronologique). Soit v un sommet de G, une modification spatio-chronologique de seuil chronologique $s > 0$, d'origine x, de longueur l et d'ordre k est un sous-ensemble $\{x_0, x_1, x_2, ..., x_l\}$ de sommets de G_v^Ψ tel que : $x = x_0$, $0 < x_{i+1} - x_i \leqslant s$ et $x_{i+1} \in \bigcup_{j=1}^{j=k} \Gamma_j^v(x_i)$ $(i = 0, 1, 2, ..., l-1)$.

Pour le graphe de la figure **5.2a**, on a pour les sommets $v = 2$ et $v = 3$ les résultats suivants

pour le voisinage entrant :

Pour $v = 2$:

$\Gamma_1^{2,in}(4) = \{5, 6, 7, 11, 12\}$, $\Gamma_1^{2,in}(5) = \{4\}$, $\Gamma_1^{2,in}(6) = \{4\}$, $\Gamma_1^{2,in}(7) = \{4\}$

$\Gamma_2^{2,in}(4) = \{13\}$, $\Gamma_2^{2,in}(5) = \{6, 7, 11, 12\}$, $\Gamma_2^{2,in}(6) = \{5, 7, 11, 12\}$, $\Gamma_2^{2,in}(7) = \{5, 6, 11, 12\}$.

On a alors :

$\bigcup_{j=1}^{2} \Gamma_j^{2,in}(4) = \{5, 6, 7, 11, 12, 13\}$,

$\bigcup_{j=1}^{2} \Gamma_j^{2,in}(5) = \{4, 6, 7, 11, 12\}$,

$\bigcup_{j=1}^{2} \Gamma_j^{2,in}(6) = \{4, 5, 7, 11, 12\}$,

$\bigcup_{j=1}^{2} \Gamma_j^{2,in}(7) = \{4, 5, 6, 11, 12\}$.

Pour $v = 3$:

$\Gamma_1^{3,in}(15) = \{9\}$, $\Gamma_2^{3,in}(15) = \{8, 10, 16, 17\}$,

$\Gamma_1^{3,in}(16) = \{9\}$, $\Gamma_2^{3,in}(16) = \{8, 10, 15, 17\}$,

$\Gamma_1^{3,in}(17) = \{9\}$, $\Gamma_2^{3,in}(17) = \{8, 10, 15, 16\}$.

On a alors :

$\bigcup_{j=1}^{2} \Gamma_j^{3,in}(15) = \{8, 9, 10, 16, 17\}$,

$\bigcup_{j=1}^{2} \Gamma_j^{3,in}(16) = \{8, 9, 10, 15, 17\}$,

$\bigcup_{j=1}^{2} \Gamma_j^{3,in}(17) = \{8, 9, 10, 15, 16\}$.

En s'intéressant aux modifications spatio-chronologiques pour le seuil chronologique $s = 1$, on a :

- pour $k = 1$ les modifications sont :
  - $\{4, 5\}, \{6\}, \{7\}, \{11, 12, 13\}$ avant le sommet 2
  - $\{8, 9, 10\}, \{15\}, \{16\}, \{17\}$ avant le sommet 3
- pour $k = 2$ les modifications sont :
  - $\{4, 5, 6, 7\}, \{11, 12, 13\}$ avant le sommet 2. En effet, on a $6 \in \bigcup_{j=1}^{2} \Gamma_j^{2,in}(5)$ et $7 \in \bigcup_{j=1}^{2} \Gamma_j^{2,in}(6)$ ce qui conduit à la réunion de $\{4, 5\}, \{6\}, \{7\}$
  - $\{8, 9, 10\}, \{15, 16, 17\}$ avant le sommet 3. En effet, on a $16 \in \bigcup_{j=1}^{2} \Gamma_j^{3,in}(15)$ et $17 \in \bigcup_{j=1}^{2} \Gamma_j^{3,in}(16)$ ce qui conduit à la réunion de $\{15\}, \{16\}, \{17\}$.

5.2.4 Comparaison des différentes définitions de modification

Dans cette sous-section, nous présentons les avantages et les inconvénients des différents types de modifications.

Nous avons présenté dans les sous-sections précédentes trois types de modification : la modification chronologique, la modification spatiale avec seuil chronologique et la modification spatio-chronologique avec seuil chronologique un peu plus général.

Modification chronologique : le principal atout de la modification chronologique est qu'elle permet de regrouper les modifications suivant la succession chronologique de leur insertion. Elle permet ainsi d'avoir dans un même groupe toutes les actions successives réalisées dans le voisinage d'un sommet donné. De ce fait, elle est un atout dans la détection des épisodes de révisions réalisées autour d'un caractère donné dans un texte. Cet avantage devient un inconvénient lorsqu'on s'intéresse à la position des révisions dans le texte. En effet, on n'est pas en mesure de savoir si les modifications d'un même groupe sont effectuées à une même position dans le texte.

Modification spatiale avec seuil chronologique : elle a pour principal avantage le fait qu'elle prend en compte la spatialité en plus de la chronologie. Ainsi, elle permet de regrouper dans un même ensemble des actions effectuées de manière successive à un même endroit dans le texte. Ce qui permet non seulement d'avoir des épisodes de révisions dans le texte, mais de savoir en plus qu'elles sont réalisées à une même position dans le texte. Elle est donc un apport important dans la localisation des révisions. Bien qu'elle soit un peu plus avantageuse par rapport à la modification chronologique, elle ne permet toutefois pas de prendre en compte certaines actions, c'est le cas par exemple des suppressions successives à une même position dans le texte.

Modification spatio-chronologique avec seuil chronologique : cette modification apporte une couche supplémentaire à la modification précédente. En effet, en ajoutant la distance entre les actions réalisées, cela permet de prendre en compte un grand nombre d'éléments que les autres modifications ne prennent pas en compte. Ainsi, elle permet d'avoir des ensembles qui traduisent au mieux les insertions et les suppressions réalisées de manière chronologique à un endroit bien précis du texte. Son principal inconvénient réside dans sa détermination, qui dépend de plusieurs voisinages mais avec de bonnes techniques de programmation cela peut être surmonté.

En résumé, les trois modifications sont utilisées suivant le genre d'information que l'on veut obtenir sur les modifications. Si l'on veut simplement connaître la succession chronologique des modifications, alors la modification chronologique sera la plus indiquée. Si on veut de plus la position des modifications, alors il sera judicieux de se tourner vers les modifications spatiales avec seuil chronologique. Et enfin, si l'on veut en plus une précision sur les distances entre les modifications, alors la modification spatio-chronologique avec seuil chronologique sera une bonne alliée.

En définissant les modifications, nous avons remarqué que certains éléments ne se retrouvent pas ensemble tout simplement à cause de l'absence d'un arc entre eux. L'objectif de la section qui suit est d'essayer d'améliorer le modèle pour pallier ce manque.

5.3 Nouveaux Modèles

Cette section présente deux nouveaux modèles dans lesquels l'indice de modification sera calculé. En effet, le fait que les sommets $5, 6$ et 7 de même que $15, 16$ et 17 (graphe Figure **5.2a**) se retrouvent dans les modifications différentes dans certains cas est dû au fait qu'il n'existe pas d'arc direct entre eux, bien qu'ils soient chronologiquement et spatialement successifs. Ainsi, pour résoudre ce problème dans le cas général, et non simplement associé à un voisinage particulier d'un sommet ou à une modification particulière, nous introduisons deux nouveaux modèles. Ces modèles sont introduits afin de faciliter l'identification des blocs de modifications.

Le premier modèle, qui est l'objet de la sous-section **5.3.1**, ajoute les arcs entre certains sommets du graphe du *modèle sans perte*. Tandis que le second modèle, présenté à la sous-section **5.3.2**, consiste à changer la façon de relier quelques sommets entre eux en plus de rajouter des sommets. Ces deux modèles, bien qu'ils se ressemblent sur plusieurs points, présentent aussi des différences qui vont être le sujet de la sous-section **5.3.3**.

5.3.1 *Modèle 1*

Cette sous-section présente le *modèle* 1, qui est un graphe différent de celui obtenu par le *modèle sans perte*. Nous commençons par donner les principes du *modèle* 1. Ensuite, nous définissons les transformations qui permettront de passer du *modèle sans perte* au *modèle* 1. Et enfin, nous terminons la sous-section en donnant quelques remarques et propriétés du *modèle* 1.

Comme mentionné en début de section, nous avons remarqué que, pour certains chemins avec ou sans seuil chronologique, des modifications sont considérées comme différentes, car il n'y a pas d'arc entre les sommets. C'est le cas principalement lorsque des suppressions et des insertions s'effectuent sur une même position dans le texte. Cela est dû au fait que, selon la construction du *modèle sans perte*, il n'y a pas d'arc entre un sommet d'insertion supprimé et le sommet de suppression qui le supprime. De plus, il n'y a pas d'arc entre les sommets de suppression qui se suivent, bien qu'ils soient effectués à la même position dans le texte. Le modèle 1 a pour but de relier par un chemin tous les sommets qui ont un même prédécesseur. En effet, lorsque des sommets ont un même prédécesseur (dans le *modèle sans perte*), alors cela signifie qu'ils sont insérés à la même position dans le texte.

Principe du *Modèle* 1 (P1)

Énoncé du principe : Soit G un graphe de processus d'écriture, soit x un sommet de G. Soit $L_x = \{v \in V/(x, v) \in A\}$ l'ensemble des successeurs immédiats du sommet x.

Pour toute paire de sommets u, v dans L_x tel que $u < v$, rajouter un arc (u, v) si $(u, v) \notin A$, $(v, u) \notin A$ et il n'existe pas de sommets $y \in L_x$ tel que $u < y < v$.

Définition 5.3.1. Un sommet x vérifie **P1** si $(u, v) \in A$ ou $(v, u) \in A$ pour toute paire $u, v \in L_x$ telle que $u < v$ et telle qu'il n'existe pas de sommet $y \in L_x$ avec $u < y < v$.

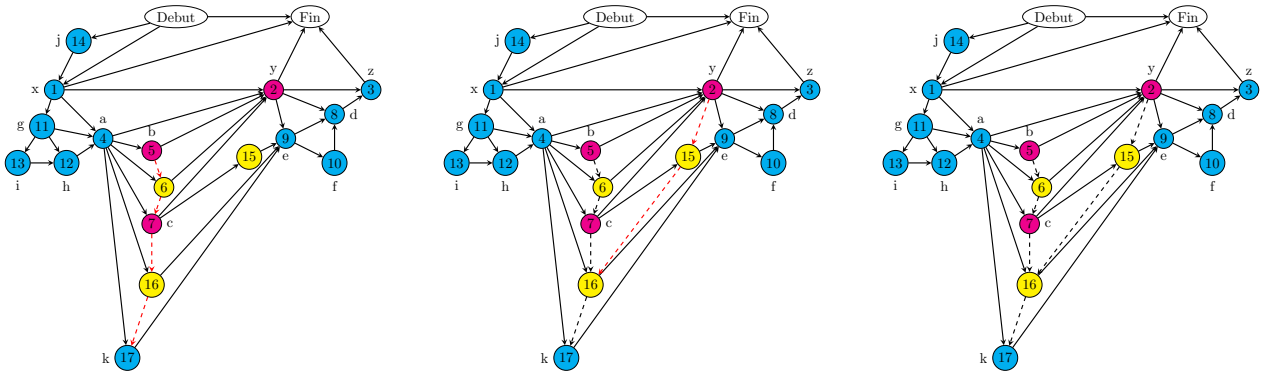
L'application du principe à un sommet x donné d'un graphe de processus d'écriture G produit un graphe GL_x où x vérifie **P1**. Pour obtenir le graphe M_1 du *modèle* 1, on doit répéter le processus plusieurs fois en commençant par le sommet *début*, jusqu'à ce que tous les sommets soient visités.

Ainsi, on peut écrire :

$$M_1 = GL_{-1}L_{u_1} \dots L_{u_n}$$

où l'ordre des sommets $\{u_1, u_2, \dots, u_n\}$ est obtenu en suivant l'ordre des sommets visités.

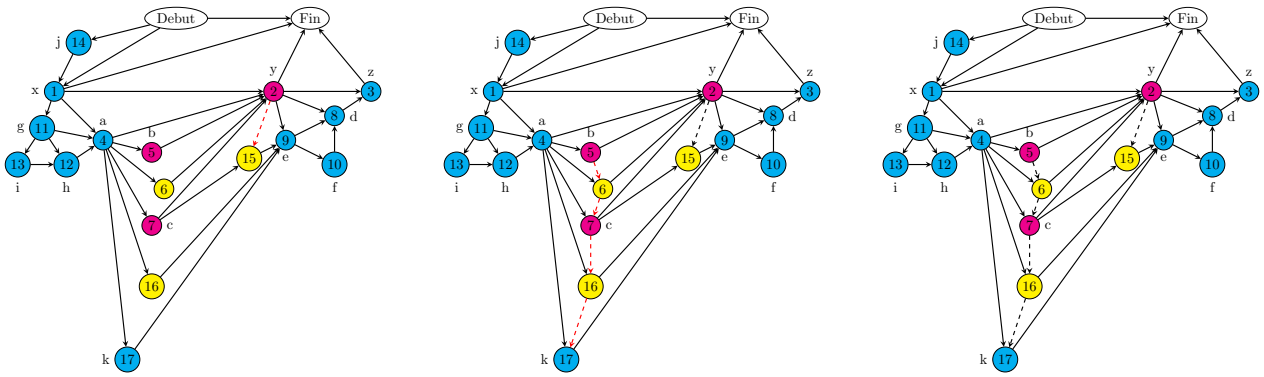
Par exemple, pour le sommet 4 du graphe de la figure **5.2a**, $L_4 = \{2, 5, 6, 7, 16, 17\}$ en appliquant le principe **P1**, on obtient $\{(5, 6), (6, 7), (7, 16), (16, 17)\}$ comme liste des arcs à ajouter dans le graphe G. Il en résulte, alors le graphe GL_4 de la figure **5.5a** où les arcs ajoutés sont représentés en pointillé rouge. La figure **5.5b** illustre le graphe obtenu en appliquant le principe **P1** au sommet 7 du graphe GL_4. On a $L_7 = \{2, 15, 16\}$ dans GL_4. En appliquant **P1**, on obtient la liste $\{(2, 15), (15, 16)\}$ des arcs à ajouter au graphe GL_4. Ce qui donne le graphe de la figure **5.5c**.



(a) GL_4 (b) GL_4L_7 (c) Graphe *modèle* 1

FIGURE 5.5 Exemple du *modèle* 1.

Si, par contre, on considère d'abord le sommet 7, puis le sommet 4, on obtient un graphe différent (Figure **5.6c**). En effet, $L_7 = \{2, 15\}$ dans G et alors seulement l'arc $(2, 15)$ est ajouté. Si l'on applique le principe **P1** au sommet 4 de GL_7, alors $L_4 = \{2, 5, 6, 7, 16, 17\}$ dans GL_7. On ajoute alors les arcs $\{(5, 6), (6, 7), (7, 16), (16, 17)\}$ à GL_7 pour obtenir GL_7L_4. Dans GL_4L_7 tous les sommets vérifient **P1**. Par contre, dans GL_7L_4, il n'y a pas d'arc entre 15 et 16 ; pourtant, 15 et 16 sont des successeurs de 7. D'où il est important de choisir un ordre qui permet d'avoir le « bon graphe » du *modèle* 1, où un graphe de processus d'écriture est dit « bon », si tous ces sommets vérifient **P1**.



(a) GL_7 (b) GL_7L_4 (c) Graphe résultant

FIGURE 5.6 Exemple en utilisant un ordre différent de celui du *modèle* 1.

Transformations et algorithmes du *modèle* 1

La transformation $\mathbf{T_1}$ définie par l'**Algorithme 6** permet d'avoir le graphe du *modèle* 1 en suivant l'ordre établi par un parcours en largeur du graphe. En effet, on commence avec le sommet -1 (Début) et on explore ses successeurs dans l'ordre croissant des numéros de sommets, avant d'explorer leurs successeurs toujours dans l'ordre croissant des numéros de sommets. Son implémentation repose sur une file (queue) dont le principe est premier entré, premier sorti, ce qui permet de s'assurer que les sommets découverts d'abord seront explorés en premier. Pour l'algorithme, nous considérons les notations suivantes :

`Notations` :

nod_{nv} :	`ensemble ordonné des sommets à visiter`
$v[u]$:	`spécifie si le sommet` u `est déjà visité (vrai) ou non (faux)`
L_x :	`liste des successeurs du sommet` x
`P1` :	`principe 1`
NL_x :	`ensemble des sommets de` L_x `non visités et qui ne sont pas encore dans` nod_{nv}

Pour illustrer le fonctionnement de l'algorithme, nous considérons le graphe de la Figure **5.7** et les étapes de l'algorithme sont regroupées dans le Tableau **5.1**.

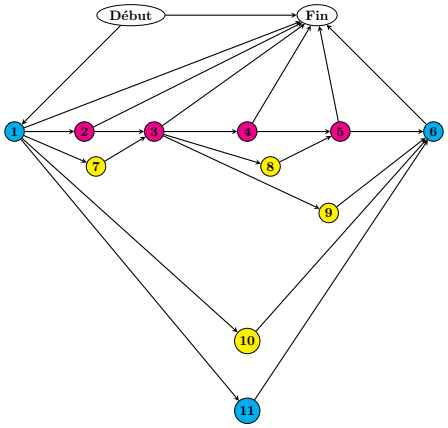


FIGURE 5.7 Graphe initial.

Il en résulte alors que

$$M = GL_1L_2L_7L_3L_4L_8L_9L_5$$

Puisque l'application du principe avec les ensembles L_2, L_4, L_9, L_5 n'ajoute aucun arc au

Algorithme 6 : Transformation $\mathbf{T_1}$

Entrées : G Graphe de processus d'écriture (*modèle sans perte*)
Sorties : M Graphe du modèle 1

$M \leftarrow G$; Faire une copy de G
pour $u \in M$ **faire**
  $v[u] \leftarrow Faux$
fin
$nod_{nv} \leftarrow [-1]$;
tant que $nod_{nv} \neq \emptyset$ **faire**
  $x \leftarrow$ premier élement de la file nod_{nv}; Ôter x de nod_{nv}
  $v[x] \leftarrow Vrai$;
  Déterminer L_x de x dans M;
  si $|L_x| > 1$ **alors**
    $M \leftarrow \mathbf{P1}(M)$; Appliquer le principe P1 à M
    $NL_x \leftarrow \{u \in L_x / v[u] = Faux \text{ et } u \notin nod_{nv}\}$;
    $nod_{nv} \leftarrow nod_{nv} \cup NL_x$; ajouter NL_x à la fin de la file nod_{nv}
  sinon
    si $|L_x| = 1$ **alors**
      $a \leftarrow L_x[0]$; l'unique élément de L_x
      si $v[a] = Faux$ *et* $a \notin nod_{nv}$ **alors**
        $nod_{nv} \leftarrow nod_{nv} \cup \{a\}$; ajouter a à la fin de la file nod_{nv}
      fin
    fin
  fin
fin
retourner M

TABLEAU 5.1 Illustration du fonctionnement de la transformation $\mathbf{T_1}$.

	nod_{nv}	x	$v[x]$	L_x	$\|L_x\|$	M	NL_x
1	[-1]	-1	Vrai	[1]	1		[1]
2	[1]	1	Vrai	[2,7,10,11]	4	ML_1	[2,7,10,11]
3	[2,7,10,11]	2	Vrai	[3,7]	2	ML_2	[3]
4	[7,10,11,3]	7	Vrai	[3,10]	2	ML_7	[]
5	[10,11,3]	10	Vrai	[6,11]	2		[6]
6	[11,3,6]	11	Vrai	[6]	1		[]
7	[3,6]	3	Vrai	[4,8,9,10]	4	ML_3	[4,8,9]
8	[6,4,8,9]	6	Vrai	[]	0		
9	[4,8,9]	4	Vrai	[5,8]	2	ML_4	[5]
10	[8,9,5]	8	Vrai	[5,9]	2	ML_8	[]
11	[9,5]	9	Vrai	[6,10]	2	ML_9	[]
12	[5]	5	Vrai	[6,9]	2	ML_5	[]
13	[]						

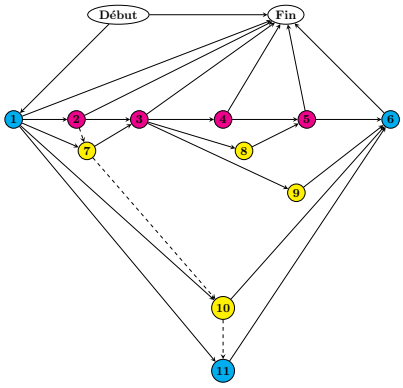
graphe, alors l'on peut écrire :

$$M = GL_1L_7L_3L_8$$

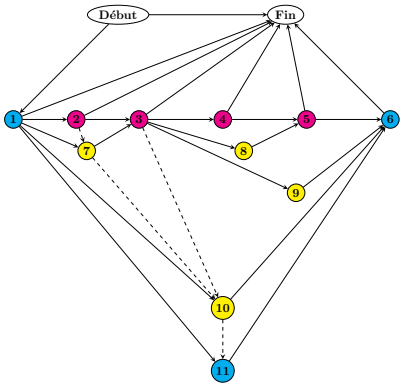
La figure **5.8** donne les graphes obtenus aux quatre étapes de la transformation.

Le but du *modèle* 1 est d'établir un lien entre les sommets d'insertions et/ou de suppressions effectuées à la même position dans le texte. En général, lorsque des insertions ou suppressions sont effectuées à la même position dans le texte, alors ils ont soit un même prédécesseur, soit un même successeur, ou même parfois les deux (successeur et prédécesseur). Dans la transformation $\mathbf{T_1}$, on a simplement considéré les sommets qui ont le même prédécesseur. Il se pose alors un problème : comment choisir l'orientation de l'arc à ajouter pour prendre en compte à la fois les sommets ayant un même prédécesseur et les sommets ayant un même successeur ? En effet, si le sommet u supprime le sommet v, doit-on ajouter l'arc (u, v) ou l'arc (v, u) ? Par exemple, dans le graphe de la figure **5.9**, si l'on ajoute les arcs $(2, 3)$ et $(3, 4)$, alors on n'a pas de chemin pour les successeurs du sommet **Debut**. Par contre, on a un chemin pour les sommets prédécesseurs du sommet **1**. Si l'on ajoute plutôt les arcs $(3, 2)$ et $(4, 3)$, on a par contre un chemin pour les successeurs de **Début** et pas de chemin pour les prédécesseurs du sommet **1**.

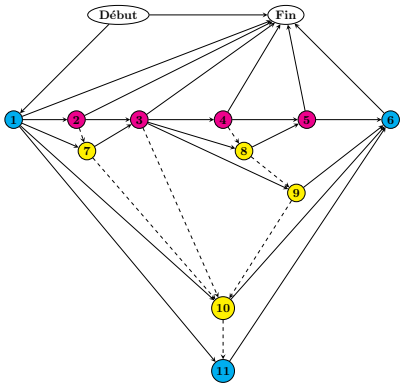
Pour tenir compte de ces différents cas, on améliore le *modèle* 1 en ajoutant à la fois les arcs (u, v) et (v, u) au lieu d'ajouter un seul des deux, ce qui permet d'avoir un chemin entre les sommets insérés à la même position dans le texte suivant qu'on choisisse les successeurs ou les prédécesseurs d'un sommet. On établit ainsi un lien spatial entre les sommets d'insertions supprimés et les sommets qui les suppriment. L'**Algorithme 7** décrit la transformation $\mathbf{T_2}$ qui permet d'obtenir le modèle amélioré.



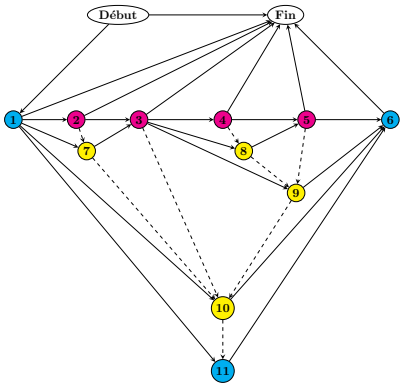
(a) GL_1



(b) GL_1L_7

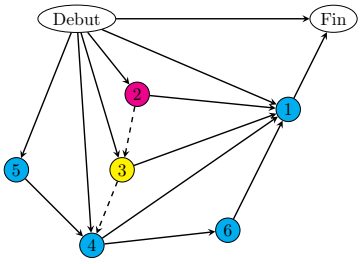


(c) $GL_1L_7L_3$

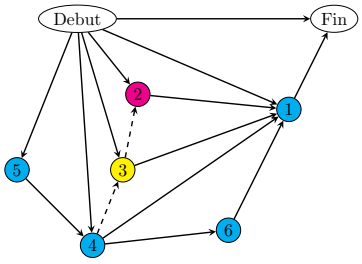


(d) $GL_1L_7L_3L_8$

FIGURE 5.8 Évolution du graphe lors de la transformation $\mathbf{T_1}$.



(a) Ajout de $(2,3)$ et $(3,4)$



(b) Ajout de $(3,2)$ et $(4,3)$

FIGURE 5.9 Ajout d'arcs.

En appliquant la transformation $\mathbf{T_2}$ au graphe $M = GL_1L_7L_3L_8$, on obtient le graphe de la Figure **5.10**

Remarque 5.3.1. *Les transformations* $\mathbf{T_1}$ *et* $\mathbf{T_2}$ *peuvent être combinées pour donner une unique transformation* $\mathbf{T}$.

Algorithme 7 : Transformation $\mathbf{T_2}$

Entrées : G Graphe *modèle sans perte*;
M Graphe du modèle 1
Sorties : H Graphe du modèle 1 amélioré
$H \leftarrow M$; `Faire une copy de` M
$edg \leftarrow \{(u,v) \in A_M/(u,v) \notin A_G\}$; `l'ensemble des arcs de` M `pas de` G
modèle
pour $(u,v) \in edg$ **faire**
| Ajouter l'arc (v,u) à H
fin
retourner H

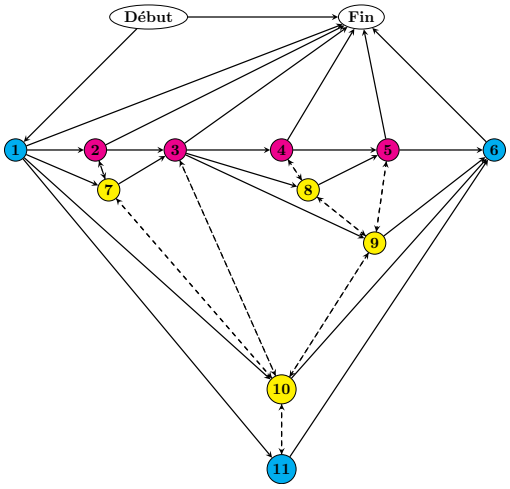


FIGURE 5.10 Graphe *modèle* 1 amélioré.

Dans la suite, sans mention contraire, le *modèle* 1 fera référence au graphe obtenu par la transformation **T**.

Remarque 5.3.2. *Si M est obtenu du graphe initial G par l'application de la transformation* **T**, *alors :*

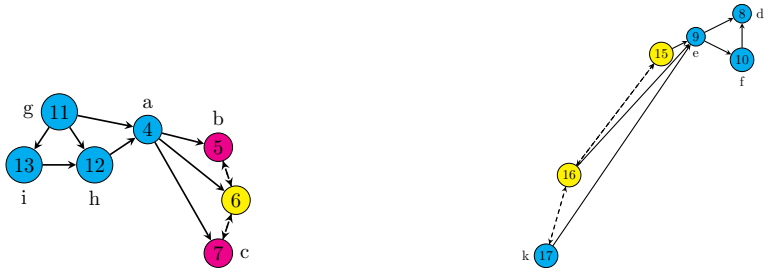
1⋆ $\mathbf{W_M} = \mathbf{W_G}$, *les graphes M et G ont les mêmes sommets.*

2⋆ $\mathbf{A_G} \subseteq \mathbf{A_M}$, *l'ensemble des arcs G est contenu dans l'ensemble des arcs de M. De plus, si $(u,v) \in A_M$ et $(u,v) \notin A_G$, alors $(v,u) \in A_M$ et $(v,u) \notin A_G$. Autrement dit, si un arc est dans M et non dans G alors l'arc inverse (v,u) est aussi dans M et non dans G.*

La présence de certains arcs avec leurs inverses dans le *modèle* 1 complique la détermination du voisinage (entrant et sortant). En effet, un même sommet peut être à la fois dans le voisinage entrant et sortant d'un autre sommet. Pour éviter ce cas de figure, on considère comme voisinage d'un sommet le voisinage du sommet obtenu dans le graphe initial (*modèle*

sans perte). Mais par contre, pour déterminer les modifications, l'on utilise le sous-graphe du *modèle* 1 induit par le voisinage du sommet.

Exemple : Considérons le graphe G de la figure **5.2a**. Les sommets $v = 2$ et $v = 3$ ont pour voisinage entrant respectif $\Psi_2^{in} = \{4, 5, 6, 7, 11, 12, 13\}$ et $\Psi_3^{in} = \{8, 9, 10, 15, 16, 17\}$. Ainsi, pour déterminer les modifications dans le *modèle* 1 au voisinage des sommets 2 et 3, on considère le sous-graphe induit par Ψ_2^{in} (figure **5.11a**) et Ψ_3^{in} (figure **5.11b**) respectivement.



(a) `sommet 2` (b) `sommet 3`

FIGURE 5.11 Sous-graphe induit par le voisinage entrant des sommets 2 et 3.

Si on considère les chemins de modification spatiale avec seuil chronologique, on obtient :

Pour $s = 1$:

- les modifications $\{4, 5, 6, 7\}, \{11, 12\}, \{13\}$ avant le sommet 2
- les modifications $\{8\}, \{9, 10\}, \{15, 16, 17\}$ avant le sommet 3

Pour $s = 2$:

- les modifications $\{4, 5, 6, 7\}, \{11, 13, 12\}$ avant le sommet 2
- les modifications $\{9, 10, 8\}, \{15, 16, 17\}$ avant le sommet 3

Si on s'intéresse aux chaînes de modifications avec seuil chronologique, on a :

Pour $s = 1$:

- les modifications $\{4, 5, 6, 7\}, \{11, 12, 13\}$ avant le sommet 2
- les modifications $\{8, 9, 10\}, \{15, 16, 17\}$ avant le sommet 3

Pour $s = 8$:

- les modifications $\{11, 13, 12, 4, 5, 6, 7\}$ avant le sommet 2
- les modifications $\{17, 16, 15, 9, 10, 8\}$ avant le sommet 3

Propriété 5.3.1. *Le modèle* 1 *est un modèle sans perte. Autrement dit, le fichier log peut être reconstruit à partir du graphe du modèle* 1.

Preuve : Soit $G = (W_G, A_G)$ le graphe du *modèle sans perte*. Soit $M = (W_M, A_M)$ le graphe du *modèle* 1 obtenu de G en utilisant la transformation **T**. L'**Algorithme 8** présente une transformation $\mathbf{T_3}$ qui permet d'ôter les liens doubles du graphe M.

Algorithme 8 : Transformation $\mathbf{T_3}$

Entrées : M Graphe du *modèle 1*
Sorties : H Graphe du *modèle sans perte*
$H \leftarrow M$; `Faire une copie de` M
$edg \leftarrow \{(u,v) \in A_M / (v,u) \in A_M\}$; `l'ensemble des arêtes de` M
pour $(u,v) \in edg$ **faire**
  Ôter les arcs (v,u) et (u,v) de H
fin
retourner H

Montrons que le graphe obtenu par cette transformation $\mathbf{T_3}$ est bel et bien le graphe initial G. Puisque aucune des transformations n'ajoute ni ne supprime des sommets, alors cela revient à montrer que les arcs ôtés par la transformation $\mathbf{T_3}$ sont les mêmes que les arcs ajoutés par la transformation **T**. Ce qui est évident, car les seuls liens ajoutés par la transformation **T** sont des arêtes et les seuls liens supprimés par $\mathbf{T_3}$ sont des arêtes aussi. En effet,

- Si (u,v) est un arc ôté de M par la transformation $\mathbf{T_3}$ alors $(v,u) \in M$, ainsi $(v,u) \notin G$, alors (u,v) est un arc ajouté.
- Si (u,v) est un arc ajouté par la transformation **T** alors $(u,v) \in M$ et $(v,u) \in M$ alors (u,v) est un arc ôté.

Il en résulte alors que les graphes H et G sont identiques, ce qui signifie que le graphe G peut être reconstruit du graphe M en utilisant la transformation $\mathbf{T_3}$. Or, le fichier log peut être reconstruit de G, car G est sans perte. D'où le fichier log peut être reconstruit du graphe M. Donc le *modèle* 1 est également sans perte. □

N.B. : Pour faire référence au fait qu'il est sans perte, le *modèle* 1 pourra souvent être appelé *modèle sans perte* 1.

Le *modèle* 1 ainsi obtenu permet d'avoir un lien spatial entre les sommets des insertions et/ ou des suppressions effectuées à une même place dans le texte. Avec ce modèle, on n'a pas besoin d'augmenter un seuil dans certains cas pour trouver des modifications dans une même position. Cela permet d'avoir des chemins ou des chaînes qui traduisent effectivement le lien spatial de la modification. La difficulté pour déterminer les modifications dans le graphe du *modèle sans perte* réside dans le fait que certains sommets n'ont pas de lien spatial direct (arc) entre eux, bien qu'ils soient effectués à la même position dans le texte. Pour

résoudre le problème, nous avons proposé un *modèle* 1 qui permet d'ajouter des arcs au *modèle sans perte*. Ces arcs sont ajoutés de manière à avoir un chemin entre les sommets insérés ou supprimés dans une même position dans le texte. Bien que le *modèle* 1 permette de pallier à ce problème, il existe aussi d'autres manières de le résoudre. Par exemple, avec une restructuration du *modèle sans perte* qui est à la base du deuxième modèle décrit dans la sous-section 5.3.2 suivante.

5.3.2 *Modèle 2*

Cette sous-section introduit le *modèle* 2 qui consiste à restructurer le graphe de processus d'écriture. Pour ce fait, nous allons tout d'abord aborder les principes qui régissent le *modèle* 2, ensuite l'algorithme de la transformation qui permet d'obtenir le graphe du *modèle* 2 à partir du *modèle sans perte*. Et nous finirons la sous-section en donnant quelques remarques et propriétés du *modèle* 2.

En analysant le graphe du *modèle sans perte*, on constate que l'absence d'arcs entre les sommets de suppression est due au fait que, lorsque l'on supprime une insertion, le sommet de suppression est relié au sommet d'insertion le plus proche de l'insertion supprimée. En effet, comme on peut l'observer dans la Figure **5.7**, le sommet de suppression 7 est relié aux sommets 1 et 3 qui sont des sommets d'insertion et voisins directs du sommet 2. Par contre, lors de la suppression du sommet 3 par le sommet 10, puisque ses voisins sont déjà supprimés, il est relié au sommet 1 et 6 les seuls sommets d'insertion qui sont les plus proche du sommet 3. Ce qui est logique, car les autres voisins sont supprimés et donc n'apparaissent plus dans le texte. Mais, dans le cas pratique pour la détermination des modifications, cela représente un frein pour la constitution des chemins. Pour remédier à cela, le modèle 2 propose de relier un sommet de suppression au dernier prédécesseur et au dernier successeur du sommet qu'il supprime. La nouveauté vient du fait que le dernier prédécesseur et le dernier successeur peuvent être de n'importe quel type. C'est-à-dire qu'ils peuvent être des insertions, des insertions supprimées ou même des suppressions.

Principe du *modèle* 2 (P2)

Énoncé du principe : Soit G un graphe de processus d'écriture (*modèle sans perte*). La construction du graphe du *modèle* 2 peut se diviser en trois grandes phases :

Phase 1 (P2a) : La première phase consiste à relier un sommet de suppression au dernier prédécesseur et au dernier successeur du sommet qu'il supprime. Soit $(u_1, \ldots, u_k)$, avec $k \geqslant 1$,

une séquence maximale de suppressions consécutives au même endroit du texte. Soit x_i le sommet supprimé par u_i. Pour $i = 1, \ldots, k$, soit $r(i)$ l'indice du $i^{ème}$ élément de $\{x_1, \ldots, x_k\}$ lorsqu'on trie cet ensemble par ordre croissant. De plus, soient $r(0) = 0, r(k+1) = k+1, a = x_0$ et $b = x_{k+1}$. Soit H le sous-graphe de G induit par $x_0, \ldots, x_{k+1}, u_1, \ldots, u_k$. Pour construire le *modèle* 2, on supprime les arcs :

- $(x_{r(i)}, u_{r(j)})$ pour tout $j > i + 1$, $i = 0, \ldots, k - 2$
- $(u_{r(i)}, x_{r(j)})$ pour tout $j > i + 1$, $i = 1, \ldots, k - 1$

et on rajoute les arcs $(u_{r(i)}, u_{r(i+1)})$ pour $i = 1, \ldots, k - 1$. On obtient alors un graphe G'

Phase 2 (P2b) : La deuxième phase reprend la transformation T_1 du *modèle* 1 qui consiste à relier les sommets de numéro successif d'un même prédécesseur. Cette fois-ci, cette transformation est appliquée au graphe G' résultant de la phase 1. On obtient alors un graphe G''.

Phase 3 (P2c) : La troisième phase reprend la transformation T_2 du *modèle* 1 qui permet d'améliorer le *modèle* 1 en ajoutant des arcs de telle sorte qu'on ait des arêtes entre certains sommets. Cette fois-ci, cette transformation est appliquée au graphe G'' résultant de la phase 2.

L'algorithme de la transformation T_4 regroupe les trois phases et permet d'obtenir le graphe du *modèle* 2 à partir du graphe initial.

`Notations` :

$Pred(x)$:	`l'ensemble des prédécesseurs du sommet` x
$Predg(x)$:	`le plus grand prédécesseur du sommet` x
$Suc(x)$:	`l'ensemble des successeurs du sommet` x
$Sucg(x)$:	`le plus grand successeur du sommet` x
$Type(x)$:	`nature du sommet` x `(insertion(I), suppression(S) ou insertion supprimée(IS))`
$Supp(x)$:	`sommet qui supprime` x
$edg_p(x)$:	`ensemble des arcs entrant en` x
$edg_s(x)$:	`ensemble des arcs sortant de` x
T_1 :	`Transformation` T_1
T_2 :	`Transformation` T_2

Exemple : Appliquons cette transformation au graphe de la figure **5.7**. Les étapes de l'algorithme sont regroupées dans le tableau **5.2**.

Algorithme 9 : Transformation $\mathbf{T_4}$

Entrées : G Graphe du *modèle sans perte*
Sorties : M Graphe du modèle 2

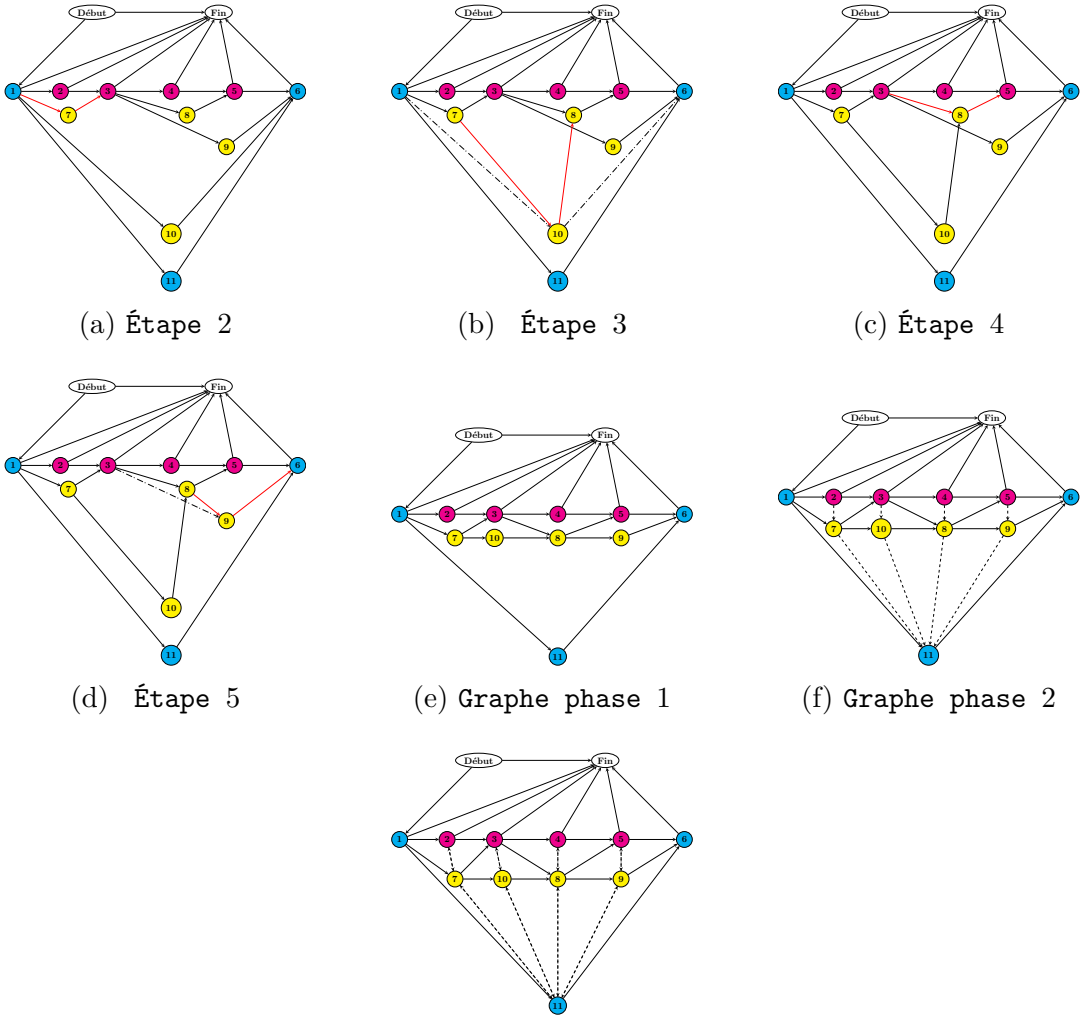
```
M ← G ;                                          Faire une copie de G
pour x ∈ M faire
    si Type(x) = IS alors
        u ← Supp(x);
        a ← Predg(x);
        b ← Sucg(x);
        si Type(a) = S alors
            pred_s ← {v ∈ Pred(x) | Type(v) = S};
            a ← min(pred_s)
        fin
        si Type(b) = S alors
            suc_s ← {v ∈ Suc(x) | Type(v) = S};
            b ← min(suc_s)
        fin
        Supprimer l'arc entrant en u;
        Supprimer l'arc sortant de u;
        Ajouter l'arc (a, u) et l'arc (u, b) au graphe M
    fin
fin
M ← T1(M);                          Appliquer la transformation T1 au graphe M
M ← T2(M);                          Appliquer la transformation T2 au graphe M
retourner M
```

TABLEAU 5.2 Illustration du fonctionnement de la transformation $\mathbf{T_4}$.

Étapes	$Type(x)$	u	a	b	$Type(a)$	a	$Type(b)$	b	edg_p	edg_s	Arcs supprimés	Arcs ajoutés
1	I											
2	IS	7	1	3	I	1	IS	3	$\{(1,7)\}$	$\{(7,3)\}$	$\{(1,7),(7,3)\}$	$\{(1,7),(7,3)\}$
3	IS	10	7	9	S	7	S	8	$\{(1,10)\}$	$\{(10,6)\}$	$\{(1,10),(10,6)\}$	$\{(7,10),(10,8)\}$
4	IS	8	3	5	IS	3	IS	5	$\{(3,8)\}$	$\{(8,5)\}$	$\{(3,8),(8,5)\}$	$\{(3,8),(8,5)\}$
5	IS	9	8	6	S	8	I	6	$\{(3,9)\}$	$\{(9,6)\}$	$\{(3,9),(9,6)\}$	$\{(8,9),(9,6)\}$
Fin phase 1	graphe M											
Fin phase 2	graphe T_1(M)											
Fin phase 3	graphe T_2(M)											

Les figures **5.12** donnent l'évolution de la transformation de ce graphe. Sur ces figures, les arcs noirs en pointillé représentent les arcs supprimés et les arcs en rouge les arcs ajoutés.

Sur les figures des étapes 2 et 4 il n'y a pas d'arcs en pointillé, car les arcs supprimés sont



(a) `Étape 2` (b) `Étape 3` (c) `Étape 4`

(d) `Étape 5` (e) `Graphe phase 1` (f) `Graphe phase 2`

(g) `Graphe du` *`modèle`* 2

FIGURE 5.12 Étapes de la transformation T_4.

les mêmes que les arcs qui ont été ajoutés. Cela se produit en général lorsque le sommet de suppression est relié au prédécesseur et au successeur direct du sommet qu'il supprime. En appliquant la transformation au graphe de la phase 1, on obtient le graphe du *modèle* 2. On remarque que ce graphe est différent du graphe du *modèle* 1. Les sommets de suppression successifs sont reliés par des arcs.

Remarques et Propriétés du *modèle* 2

Remarque 5.3.3. *Si M est obtenu du graphe initial G par l'application de la transformation* $\mathbf{T_4}$ *alors :*

1. $\mathbf{W_M = W_G}$ *les graphes* M *et* G *ont les mêmes sommets.*
2. *Si* G *est constitué uniquement de sommets d'insertion, alors les graphes* M *et* G *ont les mêmes arcs* ($A_M = A_G$). *De ce fait, les graphes* G *et* M *sont égaux car* $W_M = W_G$ (*d'après* 1).
3. *Si* G *a des sommets de suppression, alors on a* $A_M \neq A_G$.
   a) *Lorsque* G *contient un seul sommet de suppression ou que les sommets de suppression sont éloignés l'un des autres (c'est-à-dire qu'il n'y a pas de suppression d'insertion successive), alors l'ensemble des arcs de* G *est inclus dans l'ensemble des arcs de* M ($A_G \subset A_M$).
   b) *Par contre, il n'y a pas d'inclusion* ($A_G \nsubseteq A_M$), *lorsque* G *contient des sommets de suppressions d'insertion consécutifs. En effet, dans ce cas, il y a des arcs du graphe* G *qui sont supprimés lors de la transformation.*
   c) *Dans ces deux cas,* $|A_G| \leqslant |A_M|$, *c'est-à-dire le nombre d'arcs du graphe* M *est supérieure ou égale à ceux de* G. *Plus précisément, en posant* $E_M = \{(u,v) \in A_M | (v,u) \in A_M\}$ *l'ensemble des arêtes présents dans* M, *on a :*

$$|A_G| = |A_M| - |E_M|$$

   En effet, à la phase 1 *de la construction de* M, *chaque fois qu'on supprime un arc dans* G, *on ajoute un arc, ainsi le graphe obtenu à la fin de la phase* 1 *a le même nombre d'arcs que le graphe* G. *Et pour obtenir le graphe* M *les phase* 2 *et* 3 *ajoutent exactement* $|E_M|$ *arêtes au graphe de la phase* 1, *soit* $|A_M| = |A_G| + |E_M|$

   De façon plus générale on a :

   d) *Pour tout arc* (u,v) *de* A_M :
      i) *Si* $(u,v) \in A_G$ *alors* $(v,u) \notin A_M$, *autrement dit lorsqu'un arc* (u,v) *est dans* G *alors son inverse* (v,u) *n'est pas dans* M.
      ii) *Lorsque l'arc* (u,v) *n'est pas dans* G ($(u,v) \notin A_G$), *alors* u *et* v *sont des sommets de suppressions si et seulement si* $(v,u) \notin A_M$.

Comme dans le *modèle* 1, la présence des arêtes dans le *modèle* 2 complique la détermination du voisinage (entrant et sortant). Nous appliquons alors la même méthode que pour le *modèle* 1, c'est-à-dire qu'on considère comme voisinage d'un sommet le voisinage du sommet obtenu dans le graphe initial (*modèle sans perte*). Mais par contre, pour déterminer les modifications, on utilise le sous-graphe du *modèle* 2 induit par le voisinage du sommet.

Exemple : Considérons le graphe G de la figure **5.2a**. En appliquant la transformation T_4 au graphe G on obtient le graphe suivant :

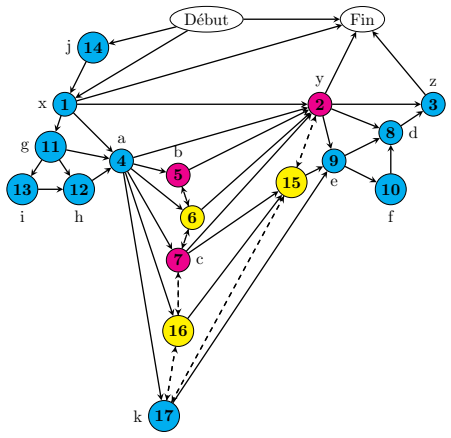
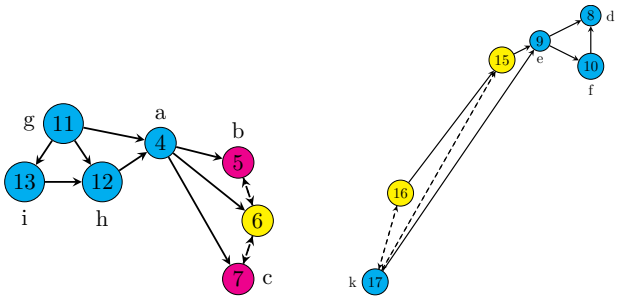


FIGURE 5.13 Graphe *modèle* 2.

Pour rappel, les sommets $v = 2$ et $v = 3$ ont pour voisinage entrant respectif $\Psi_2^{in} = \{4, 5, 6, 7, 11, 12, 13\}$ et $\Psi_3^{in} = \{8, 9, 10, 15, 16, 17\}$. Ainsi, pour déterminer les modifications dans le *modèle* 2 au voisinage des sommets 2 et 3 on considère le sous-graphe induit par Ψ_2^{in} (Figure **5.14(a)**) et Ψ_3^{in} (Figure **5.14(b)**) respectivement.



(a) sommet 2 (b) sommet 3

FIGURE 5.14 Sous-graphe induit par le voisinage entrant des sommets 2 et 3.

Si on considère les chemins de modification spatiale avec un seuil chronologique s, on obtient :

Pour $s = 1$:

— les modifications $\{4, 5, 6, 7\}, \{11, 12\}, \{13\}$ avant le sommet 2

— les modifications $\{8\}, \{9, 10\}, \{15\}, \{16, 17\}$ avant le sommet 3

Pour $s = 2$:

— les modifications $\{4, 5, 6, 7\}, \{11, 13, 12\}$ avant le sommet 2

— les modifications $\{9, 10, 8\}, \{15, 17, 16\}$ avant le sommet 3

Si on s'intéresse aux chaînes de modifications avec seuil chronologique on a :

Pour $s = 1$:

— les modifications $\{4, 5, 6, 7\}, \{11, 12, 13\}$ avant le sommet 2

— les modifications $\{8, 9, 10\}, \{15, 16, 17\}$ avant le sommet 3

Pour $s = 8$:

— les modifications $\{11, 13, 12, 4, 5, 6, 7\}$ avant le sommet 2

— les modifications $\{8, 10, 9, 15, 16, 17\}$ avant le sommet 3

Remarque 5.3.4. *Soit $G = (W_G, A_G)$ le graphe du modèle sans perte. Soit $M = (W_M, A_M)$ le graphe du modèle 2 obtenu de G en utilisant la transformation $\mathbf{T_4}$. L'**Algorithme 10** présente une transformation $\mathbf{T_5}$ qui permet d'obtenir un graphe à partir du graphe M. En effet, comme nous pourrons le constater avec la propriété **5.3.2**, le graphe obtenu est le graphe initial.*

Notations :

$Pred(x)$:	*l'ensemble des prédécesseurs du sommet* x
$Predg_{<y}(x)$:	*le plus grand prédécesseur du sommet* x *inférieur à* y
$Suc(x)$:	*l'ensemble des successeurs du sommet* x
$Sucg_{<y}(x)$:	*le plus grand successeur du sommet* x *inférieur à* y
$Type(x)$:	*nature du sommet* x *(insertion(I), suppression(S) ou insertion supprimée(IS))*

Remarque 5.3.5. *Si H est un graphe obtenu de la transformation T_5 de M graphe du modèle 2, alors :*

1) $\mathbf{W_H = W_M}$ *les graphes H et M ont les mêmes sommets.*
2) *Si M n'a que des sommets d'insertion, alors les graphes H et M ont les mêmes arcs ($A_H = A_M$). Ainsi, les graphes M et H sont identiques.*
3) *Si M a au moins deux sommets de suppression consécutif, alors on a $A_H \neq A_M$.*

   a) $|A_H| = |A_M| - |E_M|$ *où* $E_M = \{(u, v) \in A_M | (v, u) \in A_M\}$ *l'ensemble des arêtes présents dans M. En effet, pour construire H, on supprime les arêtes de M et puis dans le graphe obtenu après la suppression, à chaque fois qu'on ajoute un arc on supprime aussi un arc. Ce qui veut dire que le nombre d'arcs ne change pas dans la phase d'ajout et suppression. D'où on obtient le résultat.*

Algorithme 10 : Transformation $\mathbf{T_5}$

Entrées : M Graphe du modèle 2
Sorties : H Graphe du *modèle sans perte*

$H \leftarrow M$; `Faire une copie de` M
$edg \leftarrow \{(u,v) \in A_M \mid (v,u) \in A_M\}$; `Ensemble d'arêtes de` M
pour $(u,v) \in edg$ **faire**
  Ôter les arcs (v,u) et (u,v) de H
fin
$edg_{sp} \leftarrow \{(u,v) \in A_H \mid Type(u) = \texttt{S}$ `et` $Type(v) = \texttt{S}\}$; `Ensemble d'arcs à ôter`
$nod \leftarrow \{u \in W_H \mid Type(u) = \texttt{S}\}$; `Sommets classés par ordre croissant`
$edg_{aj} \leftarrow \{\}$; `Ensemble d'arcs à ajouter`
pour $u \in nod$ **faire**
  $a_p \leftarrow Predg_{<u}(u)$;
  $a_s \leftarrow Sucg_{<u}(u)$;
  tant que $Type(a_p) =$ `S` **faire**
    $a_p \leftarrow Predg_{<u}(a_p)$
  fin
  Ajouter l'arc (a_p, u) à la liste edg_{aj};
  tant que $Type(a_s) =$ `S` **faire**
    $a_s \leftarrow Sucg_{<u}(a_s)$
  fin
  Ajouter l'arc (u, a_s) à la liste edg_{aj};
fin
pour $(u,v) \in edg_{sp}$ **faire**
  Supprimer l' arc (u,v) de H
fin
pour $(u,v) \in edg_{aj}$ **faire**
  Ajouter l' arc (u,v) à H
fin
retourner H

b) Pour tout arc (u,v) de A_M si :

i) $(u,v) \in A_H$ alors $(v,u) \notin A_M$.

ii) $(u,v) \notin A_H$ alors u et v sont des sommets de suppressions si et seulement si $(v,u) \notin A_M$.

c) Pour tout arc (u,v) de A_H si :

$\star$) $(u,v) \in A_M$ alors $(v,u) \notin A_M$

$\star\star$) $(u,v) \notin A_M$, alors l'arc (u,v) a exactement un sommet de suppression. Et il existe des sommets de suppression $x_1, x_2, ..., x_k$ avec $k \geqslant 1$ tel que (u, x_1, x_2, ..., x_k, v) soit un chemin dans M.

- *Si u est une suppression :*

$$x_1 = Sucg_{<u}(u),$$
$$x_{i+1} = Sucg_{<u}(x_i),$$
$$v = Sucg_{<u}(x_k), \quad pour\ 1 \leqslant i \leqslant k-1$$

où $Sucg_{<y}(x)$ est le plus grand successeur du sommet x inférieur à y.

- *Si v est une suppression :*

$$x_k = Predg_{<v}(v),$$
$$x_{i-1} = Predg_{<v}(x_i),$$
$$u = Predg_{<v}(x_1), \quad pour\ 2 \leqslant i \leqslant k$$

où $Predg_{<y}(x)$ est le plus grand prédécesseur du sommet x inférieur à y.

Ces résultats sont obtenus de l'algorithme 10 de la transformation T_5.

Propriété 5.3.2. *Le modèle 2 est un modèle sans perte. Autrement dit, le fichier log peut être reconstruit à partir du graphe du modèle 2.*

Preuve : Soit $G = (W_G, A_G)$ le graphe du *modèle sans perte*. Soit $M = (W_M, A_M)$ le graphe du *modèle* 2 obtenu de G en utilisant la transformation $\mathbf{T_4}$. Montrons que le graphe obtenu par cette transformation est bel et bien le graphe initial G. Soit H le graphe obtenu en appliquant la transformation $\mathbf{T_5}$. Montrons que $H = G$.

Si M est un graphe sans sommet de suppression, alors $H = M$ et $G = M$ d'où $H = G$.

Supposons maintenant que M contienne des sommets de suppressions. On a $W_H = W_G$, car les transformations T_4 et T_5 n'ajoutent ni ne suppriment les sommets des graphes.

Montrons que $A_H = A_G$

D'après $3.c)$ de la remarque **5.3.3** et $3*)a*)$ de la remarque **5.3.5**

$$\text{on a } \begin{cases} |A_G| = |A_M| - |E_M| \\ \\ |A_H| = |A_M| - |E_M| \end{cases} \text{ alors } |A_G| = |A_H|$$

Supposons que $A_H \nsubseteq A_G$. Il existe alors un arc (u, v) de H qui n'appartient pas à G.

- **`Si`** $(u, v) \in A_M$ **`:`** Comme $(u, v) \in A_H$, alors $(v, u) \notin A_M$ car le graphe H n'a pas d'arêtes. Puisque $(u, v) \notin A_G$ par hypothèse et que $(v, u) \notin A_M$ alors u et v sont des sommets de suppressions (d'après $3.d)ii)\star$) remarque **5.3.3**). Ce qui contredit le fait que $(u, v) \in A_H$, car le graphe H n' admet pas d'arc entre deux sommets de suppression.
- **`Si`** $(u, v) \notin A_M$ **`:`** Comme $(u, v) \in A_H$ on a d'après $3*)c*)\star\star$) de la remarque **5.3.5**, l'arc (u, v) a un unique sommet de suppression. Et il existe un chemin $(u,\ x_1, x_2, ...,\ x_k, v)$

dans M tel que $x_1, x_2, ..., x_k$ soient des sommets de suppressions avec $k \geqslant 1$. Posons $y_1, y_2, ..., y_k$ les sommets supprimés respectivement par les sommets $x_1, x_2, ..., x_k$.

∘) Si u est un sommet de suppression alors

$x_1 = Sucg_{<u}(u)$, $x_{i+1} = Sucg_{<u}(x_i)$, $v = Sucg_{<u}(x_k)$, $1 \leqslant i \leqslant k-1$

Posons x le sommet supprimé par le sommet u. Puisqu'il y a un chemin (u, x_1, x_2, ..., x_k, v) dans M alors on a aussi un chemin $(x, y_1, y_2, ..., y_k, v)$ dans le graphe G, par construction de M. De plus, $x, y_1, y_2, ..., y_k, v$ sont tous inférieur à u (car $\forall i\ y_i < x_i < u$). Ainsi, v est le sommet le plus proche (parmi les successeurs) de x non supprimé au moment de la suppression de x par u. Autrement dit, le caractère du sommet v est le successeur du caractère du sommet x dans le texte au moment de la suppression du caractère du sommet x par u. D'où, d'après la construction du graphe G, $(u, v) \in A_G$ ce qui contredit l'hypothèse $(u, v) \notin A_G$.

∘∘) Si v est un sommet de suppression, alors $x_k = Predg_{<v}(v)$, $x_{i-1} = Predg_{<v}(x_i)$, $u = Predg_{<v}(x_1)$, $2 \leqslant i \leqslant k$

Posons x le sommet supprimé par le sommet v. Comme il y a un chemin (u, x_1, x_2, ..., x_{k-1}, x_k, v) dans M, alors on a aussi un chemin (u, y_1, y_2, ..., y_{k-1}, y_1, x) dans le graphe G. De plus, $u, y_1, y_2, ..., y_k, x$ sont tous inférieur à v (car $\forall i\ y_i < x_i < v$). Ainsi, u est le sommet le plus proche (parmi les prédécesseurs) de x non supprimé au moment de la suppression de x par v. Autrement dit, le caractère du sommet u est le prédécesseur du caractère du sommet x dans le texte au moment de la suppression du caractère du sommet x par v. D'où, d'après la construction du graphe G, $(u, v) \in A_G$ ce qui contredit l'hypothèse $(u, v) \notin A_G$.

D'où il n'existe pas d'arc (u, v) de H qui ne soit pas dans G, de ce fait $A_H \subseteq A_G$.

Puisque les ensembles A_G et A_H ont le même nombre d'éléments ($|A_G| = |A_H|$) et l'un est contenu dans l'autre ($A_H \subseteq A_G$) alors on a $A_H = A_G$. Par conséquent, les graphes G et H sont identiques, $H = G$ lorsque M contient des sommets de suppression.

Que M soit sans sommets de suppression ou avec sommets de suppression, les graphes G et H sont identiques. Ainsi, le graphe G peut être reconstruit du graphe M grâce à la transformation T_5. De plus, le fichier log peut être reconstruit de G, car G est sans perte. D'où, on peut conclure que le *modèle* 2 est également sans perte car le fichier log peut être reconstruit du graphe issu du *modèle* 2. □

N.B. : Pour faire référence au fait qu'il est sans perte, le modèle 2 pourra souvent être appelé *modèle sans perte* 2.

5.3.3 Comparaisons sommaire des modèles

Dans cette sous-section, nous donnerons des points communs et des différences entre les deux modèles et aussi avec le modèle initial.

1. Lorsque le graphe de processus d'écriture initial G est constitué uniquement de sommets d'insertion, le graphe initial, le graphe du *modèle* 1 et du *modèle* 2 sont égaux. En effet, les transformations T et T_4 ne rajoutent et ne suppriment aucun sommet. De plus, aucun nouvel arc et aucune nouvelle arête n'est ajouté (car il n'y a pas de sommet de suppression), alors les trois graphes ont les mêmes sommets et arcs. D'où l'égalité des trois graphes.
2. Lorsque le graphe de processus d'écriture initial G contient des sommets de suppressions consécutives au même endroit du texte, alors le graphe initial est différent de celui des *modèles* 1 et 2. S'il n'y a qu'un sommet de suppression ou que des sommets de suppression éloignés les uns des autres, alors il peut y avoir égalité entre les graphes du *modèle* 1 et 2.

5.4 Détermination de l'indice de modification

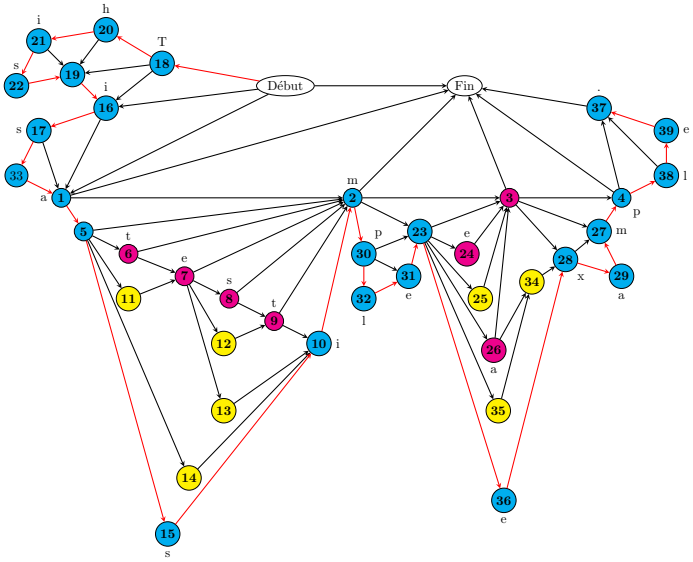
Cette section présente comment calculer l'indice de modification dans un graphe de processus d'écriture.

Pour rappel, l'indice de modification se veut être un indice qui renseigne sur le nombre de modifications (insertion et/ou suppression) effectuées aux alentours d'un caractère du texte. Pour préciser ce que nous entendons par « aux alentours d'un caractère », nous avons défini trois notions de voisinage, entre autres les voisinages entrant, sortant et une réunion des deux. Ces voisinages permettent d'appréhender la notion de proximité dans le graphe de processus par le même fait, la proximité des caractères du texte. Ensuite, les voisinages sont partitionnés en blocs de modifications. Ces blocs de modification sont définis de trois différentes façons : suivant la chronologie ou la spatialité ou encore en considérant à la fois la chronologie et la spatialité des actions réalisées. En observant que certains sommets n'étaient pas pris en compte de façon adéquate dans les blocs de modification, nous avons introduit deux nouveaux modèles. Un modèle qui ajoute de simples arêtes entre ces sommets et un autre modèle qui fait une restructuration du graphe initial. Au vu de toutes ces notions que nous avons définies, cette section présente comment combiner ces définitions pour calculer l'indice de modification. En effet, suivant le voisinage, la modification et le modèle choisi, l'on peut se retrouver avec un indice de modification différent. D'où la nécessité de structurer la

manière de calculer cet indice.

La sous-section 5.4.1 parle de la détermination du voisinage dans un graphe. Puis, la sous-section 5.4.2 présente une méthode pour trouver des chemins et des chaînes qui sont les éléments importants de la définition des modifications. Par la suite, la sous-section 5.4.3 donne les particularités de ces chemins suivant le modèle choisi.

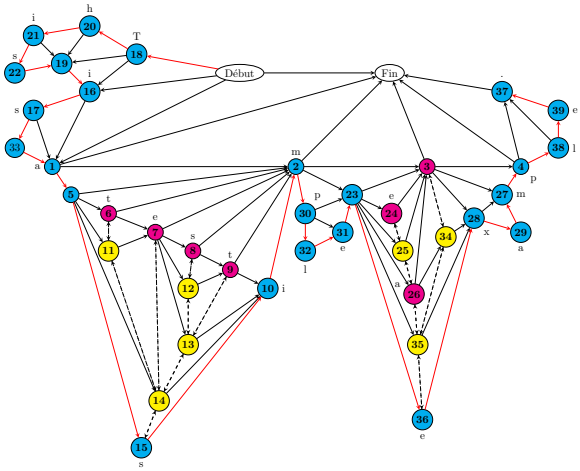
Pour illustrer le calcul d'indice de modification, considérons le graphe de processus d'écriture de la figure 5.15a.



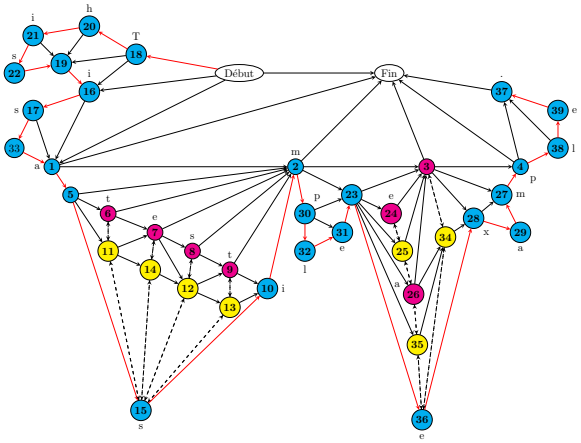
(a) Graphe *modèle sans perte*

T$_1$: '**am p**'
T$_2$: '*a* **testi***m p*'
T$_3$: '*a* ~~**t**~~*estim p*'
T$_4$: '*a* ~~**e**~~*stim p*'
T$_5$: '*a* ~~**t**~~*im p*'
T$_6$: '*a* ~~**e**~~*im p*'
T$_7$: '*a* **s***im p*'
T$_8$: '**is***a sim p*'
T$_9$: '**T** *isa sim p*'
T$_{10}$: '***T*his** *isa sim p*'
T$_{11}$: '*This isa sim* ~~**e**~~**a** *p*'
T$_{12}$: '*This isa sim a* **m***p*'
T$_{13}$: '*This isa sim a* **xa***mp*'
T$_{14}$: '*This is a sim a xamp*'
T$_{15}$: '*This is a sim***pe** *a xamp*'
T$_{16}$: '*This is a simp***l***e a xamp*'
T$_{17}$: '*This is a simple* ~~**a**~~ **e***xamp*'
T$_{18}$: '*This is a simple examp***.**'
T$_{19}$: '*This is a simple examp***le**.'
Text : '*This is a simple example.*'

(b) Processus d'écriture



(c) Modèle 1



(d) Modèle 2

FIGURE 5.15 Graphe de processus d'écriture.

La transformation du graphe par le *modèle* 1 et 2 est représentée par les figures 5.15c et 5.15d respectivement. Pour cet exemple, nous considérons le calcul d'indice pour les sommets 1, 2, 3 et 4. Nous utiliserons les notations suivantes.

`Notations` :

$idM_{\mathtt{c}}(v)$: indice de modification chronologique du voisinage du sommet v. De même, on a respectivement $idM_c^{in}(v)$ et $idM_c^{out}(v)$ pour le voisinage entrant et sortant du sommet v.

$idM^{s}_{\mathtt{cs}_1}(v)$: indice de modification spatiale de seuil chronologique s du voisinage du sommet v (chemin de modification spatiale de seuil chronologique). De même, on a respectivement $idM^{in,s}_{\mathrm{cs}_1}(v)$ et $idM^{out,s}_{\mathrm{cs}_1}(v)$ pour le voisinage entrant et sortant du sommet v.

$idM^{s}_{\mathtt{cs}_2}(v)$: indice de modification spatiale de seuil chronologique s du voisinage du sommet v (chaîne de modification spatiale de seuil chronologique). De même, on a respectivement $idM^{in,s}_{\mathrm{cs}_2}(v)$ et $idM^{out,s}_{\mathrm{cs}_2}(v)$ pour le voisinage entrant et sortant du sommet v.

$idM^{s,k}_{\mathtt{sc}}(v)$: indice de modification spatio-chronologique de seuil chronologique s et d'ordre k du voisinage du sommet v. De même, on a respectivement $idM^{in,s,k}_{\mathrm{sc}}(v)$ et $idM^{out,s,k}_{\mathrm{sc}}(v)$ pour le voisinage entrant et sortant du sommet v.

5.4.1 Détermination des voisinages

Les voisinages sont déterminés dans le graphe du *modèle sans perte*. Cela est dû à la présence des arêtes dans les autres modèles. De ce fait, pour un sommet donné, le voisinage dans le *modèle* 1 ou 2 est obtenu en considérant le graphe induit par le voisinage du sommet obtenu dans le *modèle sans perte*.

Voisinage entrant

– Sommet **1** : on a $N_1^{in}(1) = \{16, 17, 33\}$, $N_2^{in}(1) = \{18, 19\}$, $N_3^{in}(1) = \{20, 21, 22\}$

Puisque le sommet 16, premier prédécesseur du sommet 1, n'est pas supprimé alors

$$\begin{aligned}\Psi_1^{in} &= \bigcup_{i=1}^{3} N_i^{in}(1)\\ &= \{16, 17, 18, 19, 20, 21, 22, 33\}\end{aligned}$$

Ce qui donne le sous-graphe induit de la Figure **5.16** qui est le même pour les trois modèles.

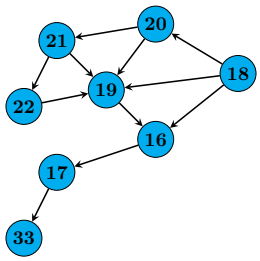


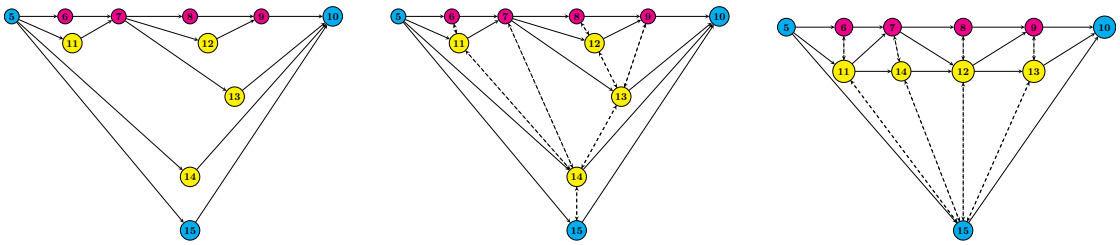
FIGURE 5.16 Voisinage entrant du sommet 1.

– Sommet **2** : on a $N_1^{in}(2) = \{5, 6, 7, 8, 9, 10\}$ et $N_2^{in}(2) = \{11, 12, 13, 14, 15\}$

Comme le sommet 1, premier prédécesseur du sommet 2, n'est pas supprimé alors

$$\begin{aligned}\Psi_2^{in} &= \bigcup_{i=1}^{2} N_i^{in}(2)\\ &= \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}\end{aligned}$$

On obtient les sous-graphes induits de la Figure **5.17** pour les différents modèles :



(a) *Modèle sans perte* (b) Modèle 1 (c) Modèle 2

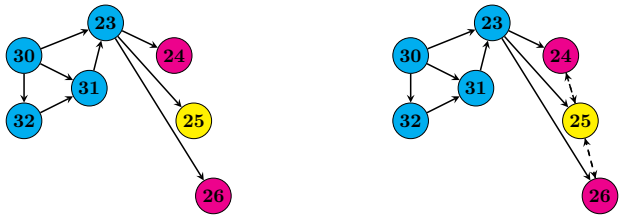
FIGURE 5.17 Voisinage entrant du sommet 2.

– Sommet **3** : on a $N_1^{in}(3) = \{23, 24, 25, 26\}$, $N_2^{in}(3) = \{30, 31\}$ et $N_3^{in}(3) = \{32\}$

Comme le sommet 2, premier prédécesseur du sommet 3, n'est pas supprimé alors

$$\begin{aligned}\Psi_3^{in} &= \bigcup_{i=1}^{3} N_i^{in}(3)\\ &= \{23, 24, 25, 26, 30, 31, 32\}\end{aligned}$$

On obtient les sous-graphes induits de la Figure **5.18** pour les différents modèles :



(a) *Modèle sans perte* (b) Modèle 1 et 2

FIGURE 5.18 Voisinage entrant du sommet 3.

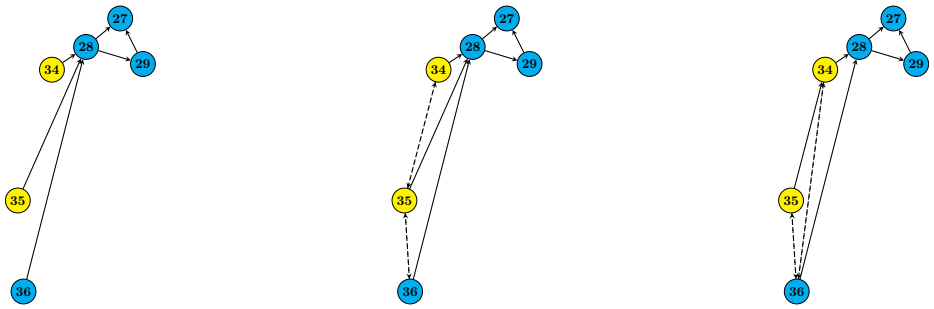
– Sommet **4** : on a $N_1^{in}(4) = \{27\}$, $N_3^{in}(4) = \{34, 35, 36\}$, $N_5^{in}(4) = \{30, 31\}$

$N_2^{in}(4) = \{28, 29\}$, $N_4^{in}(4) = \{23, 26\}$, $N_6^{in}(4) = \{32\}$

Comme le sommet 3, premier prédécesseur du sommet 4, est supprimé alors

$$\begin{aligned} \Psi_4^{in} &= \bigcup_{i=1}^{6} N_i^{in}(4) / \bigcup_{i=1}^{\infty} N_i^{in}(3) \\ &= \{27, 28, 29, 34, 35, 36\} \end{aligned}$$

On obtient les sous-graphes induits de la figure **5.19** pour les différents modèles.



(a) *Modèle sans perte* (b) *Modèle 1* (c) *Modèle 2*

FIGURE 5.19 Voisinage entrant du sommet 4.

Voisinage sortant

– Sommet **1** : on a $N^{out}(1) = \{2, 5\}$, où les sommet 2 et 5 sont des sommets d'insertion non supprimés. Pour rappel $\Psi_2^{in} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ et $\Psi_5^{in} = \emptyset$ on a alors :

$N_1^{out,2}(1) = \{5\}$ $N_3^{out,2}(1) = \{7, 10\}$ $N_5^{out,2}(1) = \{9\}$

$N_2^{out,2}(1) = \{6, 11, 14, 15\}$ $N_4^{out,2}(1) = \{8, 12, 13\}$

Ce qui donne $\Psi_1^{out} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

Ainsi $\Psi_1^{out} = \Psi_2^{in}$. On obtient, alors les mêmes sous-graphes induits que le voisinage entrant du sommet 2 représenté à la Figure **5.17**.

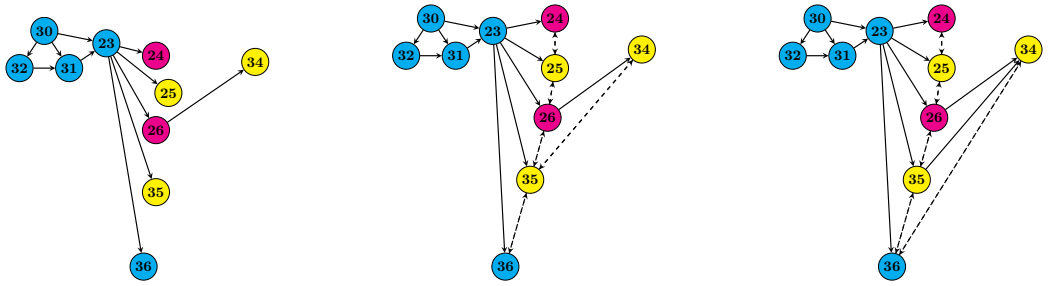
– Sommet **2** : on a $N^{out}(2) = \{3, 23, 30\}$, où le sommet 3 est un sommet d'insertion supprimé par le sommet de suppression 34 et le successeur de 34 est 28, $\Psi_{28}^{in} = \{34, 35, 36\}$. Alors on a : $N_1^{out,3}(2) = \{23, 30\}$, $N_2^{out,3}(2) = \{24, 25, 26, 31, 32, 35, 36\}$, $N_3^{out,3}(2) = \{34\}$

Les sommets 23 et 30 sont des sommets d'insertion non supprimés. On a

$\Psi_{23}^{in} = \{30, 31, 32\}$ et $\Psi_{30}^{in} = \emptyset$ alors : $N_1^{out,23}(2) = \{30\}$ et $N_2^{out,23}(2) = \{31, 32\}$

Ce qui donne $\Psi_2^{out} = \{23, 24, 25, 26, 30, 31, 32, 34, 35, 36\}$

On obtient, alors les sous-graphes induits de la Figure **5.20**.



(a) *Modèle sans perte* (b) *Modèle 1* (c) *Modèle 2*

FIGURE 5.20 Voisinage sortant du sommet 2.

– Sommet **3** : Le sommet 3 est supprimé par le sommet 34 et $N^{out}(3) = \{4, 27, 28\}$. Les sommet 4, 27 et 28 sont des insertions non supprimés. Pour rappel

$\Psi_4^{in} = \{27, 28, 29, 34, 35, 36\}$, $\Psi_{27}^{in} = \{28, 29\}$ et $\Psi_{27}^{in} = \emptyset$ alors on a :

$N_1^{out,4}(3) = \{27, 28\}$ $N_1^{out,27}(3) = \{28\}$

$N_2^{out,4}(3) = \{29\}$ $N_2^{out,27}(3) = \{29\}$

Ce qui donne $\Psi_3^{out} = \{27, 28, 29\}$

On obtient un sous-graphe induit identique pour les trois modèles (Figure **5.21**).

– Sommet **4** : On a $N^{out}(4) = \{37, 38\}$ avec $\Psi_{37}^{in} = \{38, 39\}$ et $\Psi_{38}^{in} = \emptyset$. Ainsi $\Psi_4^{out} = \{38, 39\}$

Voisinage

Avec les voisinages entrant et sortant, on obtient alors les voisinages suivants :

$$\Psi_1 = \Psi_1^{in} \cup \Psi_1^{out} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 33\}$$

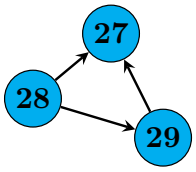


FIGURE 5.21 Voisinage sortant du sommet 3.

$$\Psi_2 = \Psi_2^{in} \cup \Psi_2^{out} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 23, 24, 25, 26, 30, 31, 32, 34, 35, 36\}$$

$$\Psi_3 = \Psi_3^{in} \cup \Psi_3^{out} = \{23, 24, 25, 26, 27, 28, 29, 30, 31, 32\}$$

$$\Psi_4 = \Psi_4^{in} \cup \Psi_4^{out} = \{27, 28, 29, 34, 35, 36, 38, 39\}$$

Après avoir déterminé les voisinages, la prochaine étape est de déterminer les blocs de modification dans le voisinage obtenu. Ce qui est l'objectif de la sous-section suivante.

5.4.2 Détermination des chemins et des chaînes

Dans cette sous-section, nous présentons une méthode pour la détermination des chemins et des chaînes qui sont la base de la définition des modifications.

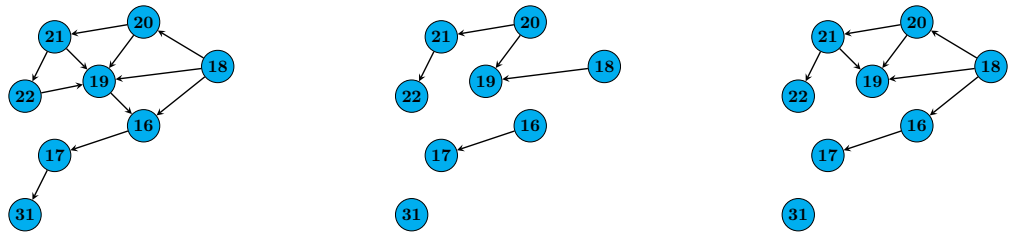
L'étape de la division des voisinages en blocs de modification débute par l'élimination des arcs ou arêtes qui ne respectent pas le seuil chronologique considéré. Ainsi, pour tout sommet $v \in V$, et pour un seuil chronologique $s \geqslant 1$ donné, posons :

$G_v^{\Psi,s}$ le graphe obtenu du graphe G_v^{Ψ} en ôtant les arcs ou les arêtes (x, y) du graphe G_v^{Ψ} tel que $|x - y| > s$, autrement dit, les arcs ou les arêtes qui ne respectent pas le seuil chronologique.

De même, on définit $G_v^{\Psi^{in},s}$ et $G_v^{\Psi^{out},s}$ respectivement pour les voisinages entrant et sortant du sommet v.

Par exemple, en supposant que le graphe de la figure **5.22a** est un graphe induit par un voisinage d'un sommet v, alors pour les seuils chronologiques $s = 1$ et $s = 2$ on obtient respectivement les graphes $G_v^{\Psi,1}$ et $G_v^{\Psi,2}$ des figures **5.22b** et 5.22c

Comme nous pouvons le constater, suivant le modèle, la façon de choisir les chemins ou les chaînes peut impacter le calcul de l'indice de modification. En effet, considérons par exemple le graphe de la figure **5.23** et supposons qu'il s'agisse d'un graphe induit par un voisinage d'un sommet. Pour le chemin de modification spatiale avec seuil chronologique $s = 2$, on a plusieurs ensembles de chemins possibles, par exemple $\{\{16, 17\}, \{18, 19\}, \{20, 21, 22\}\}$, $\{\{16, 17\}, \{18, 20, 19\}, \{21, 22\}\}$, $\{\{18, 16, 17\}, \{19\}, \{20, 21, 22\}\}$. Dans ce cas-ci, les chemins ont sensiblement la même longueur et le même nombre. Toutefois, il y a des cas où les



(a) G_v^{Ψ} (b) $G_v^{\Psi,1}$ (c) $G_v^{\Psi,2}$

FIGURE 5.22 Exemple de graphe induit par un voisinage pour les seuil chronologique $s = 1$ et $s = 2$.

chemins trouvés sont très différents. Il est alors primordial de trouver la manière adéquate pour déterminer les chemins et les chaînes. Dans cette sous-section, nous présenterons une méthode basée sur la circulation de flots dans un réseau.

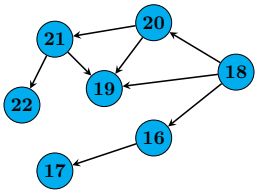


FIGURE 5.23 Exemple de graphe induit par un voisinage.

Indépendamment de la méthode utilisée pour calculer l'indice de modification, l'on peut se retrouver avec plusieurs ensembles de chemins ou de chaînes. Toutefois, on souhaite avoir les chemins ou les chaînes qui prennent en compte toutes les modifications effectuées et, de plus, le nombre de chemins ou de chaînes doit traduire le nombre de modifications effectuées. Alors comment choisir ces chemins ou chaînes ? Ce problème peut être vu comme un problème de circulation dans un réseau. En effet, on construit un graphe H à partir du graphe d'origine G_v^{Ψ} (sous-graphe induit par le voisinage de v) en remplaçant chaque sommet u de G_v^{Ψ} par deux sommets a_u et b_u ; les arcs qui entrent en u, sont remplacés par les arcs entrant en a_u ; les arcs qui sortent de u par les arcs sortant par b_u ; on ajoute les arcs (a_u, b_u) ; on ajoute enfin deux sommets s et t appelés respectivement source et puits. On relie s à chacun des sommets a_u du graphe H par un arc (s, a_u) et chaque sommet b_u est relié au sommet t par un arc (b_u, t). On ajoute aussi un arc du puits à la source $(t,\ s)$. En considérant le graphe de la Figure **5.23** on obtient le graphe H de la Figure **5.24** :

On cherche à minimiser la quantité de flots sur l'arc (t,s). Les arcs de ce flot compatible vont

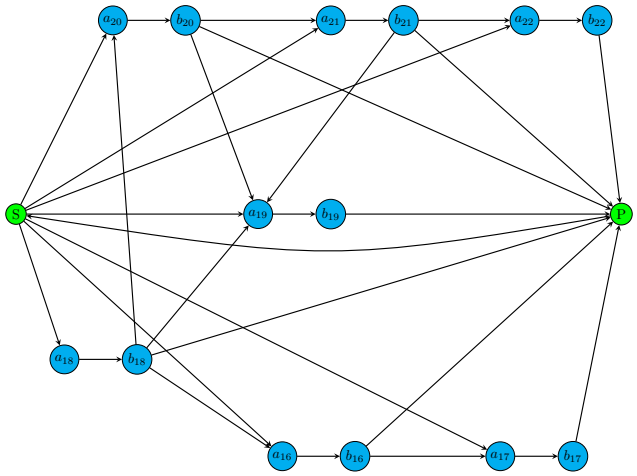


FIGURE 5.24 Graphe H correspondant au graphe de la figure **5.23**.

alors correspondre à un ensemble de chemins dans le graphe original.

Description du modèle :

Notations : Posons

$K^+(u)$ les arcs entrant du sommet u dans H

$K^-(u)$ les arcs sortant du sommet u dans H

$A(H)$ l'ensemble des arcs de H

Données : Considérons

- μ_a la capacité de chaque arc a
- C_a coût de l'arc a
- ν_a la borne inférieure sur le flot devant passer sur l'arc a

Variables de décisions : f_a valeur du flot sur l'arc a

Objectif : minimiser le flot sur l'arc (t,s)

Contraintes : Le flot doit respecter les contraintes suivantes :

- Conservation de flot :

  À chaque sommet du graphe, le nombre d'arcs entrant doit être égal au nombre d'arcs sortants. La contrainte est alors pour tout sommet $u \in H$:

$$\sum_{a \in K^+(u)} f_a = \sum_{a \in K^-(u)} f_a$$

– Capacité des arcs :

Les bornes, inférieure et supérieure, sur le flot devant passer sur chaque arc doivent être respectées. Dans le graphe H, il y a 4 types d'arcs. Les arcs (s, a_x), (a_x, b_x), (b_x, a_y), (b_x, t), (t, s) avec $x, y \in G_v^{\Psi}$.

Pour les arcs de la forme (a_x, b_x), on considère que la borne inférieure et supérieure sont égales à 1. En effet, les arcs (a_x, b_x) représentent le sommet x du graphe G_v^{Ψ} et les sommets x de G_v^{Ψ} doivent appartenir à un seul chemin. Ce qui donne pour tout x sommet de G_v^{Ψ} :

$$\nu_{(a_x,b_x)} = \mu_{(a_x,b_x)} = 1$$

Pour l'arc (t, s), on considère que la capacité est infinie.

Pour les autres arcs du graphe H, on considère que le flot minimal est 0 et que le flot maximal est 1. Car ils doivent appartenir à au plus un chemin.

– Le flot doit avoir une valeur entière.

Le problème se formule alors comme :

$$\min \ \sum_{a \in A(H)} f_a C_a$$

$$\text{s.c.} \quad \sum_{a \in K^+(u)} f_a = \sum_{a \in K^-(u)} f_a \quad \text{pour tout } u \neq s, t$$

$$\nu_a \leqslant f_a \leqslant \mu_a \quad \text{pour tout } a \in A(H)$$

$$f_a \text{ entier} \quad \text{pour tout } a \in A(H)$$

En considérant :

- $\nu_a = 1$, $\mu_a = 1$ pour tout $a = (a_v, b_v) \in A(H)$
- $\nu_a = 0$, $\mu_a = 1$ pour tout $a \in A(H)$, tel que $a \neq (a_v, b_v)$ et $a \neq (s, t)$
- $\nu_{(t,s)} = 0$, $\mu_{(t,s)} = \infty$
- $C_{(t,s)} = 1$ et $C_a = 0$ pour tout $a \in A(H)$, tel que $a \neq (t, s)$

En utilisant la fonction *milp (Mixed-integer linear programming)* du package *Scipy* de python on obtient les chemins en rouge de la figure **5.25**.

Ce qui correspond alors aux chemins : $\{\{18, 16, 17\}, \{19\}, \{20, 21, 22\}\}$. Ces trois chemins représentent les trois modifications effectuées au voisinage du sommet considéré.

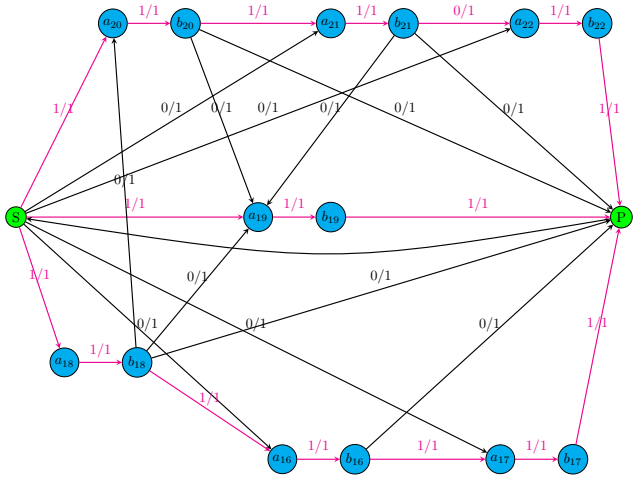


FIGURE 5.25 Graphe H avec les chemins.

Indice de modification

Pour rappel, on a les voisinages suivants :

$\Psi_1^{in} = \{16, 17, 18, 19, 20, 21, 22, 33\}$ $\quad$ $\Psi_1^{out} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

$\Psi_2^{in} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ $\quad$ $\Psi_2^{out} = \{23, 24, 25, 26, 30, 31, 32, 34, 35, 36\}$

$\Psi_3^{in} = \{23, 24, 25, 26, 30, 31, 32\}$ $\quad$ $\Psi_3^{out} = \{27, 28, 29\}$

$\Psi_4^{in} = \{27, 28, 29, 34, 35, 36\}$ $\quad$ $\Psi_4^{out} = \{38, 39\}$

$\Psi_1 = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 33\}$

$\Psi_2 = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 23, 24, 25, 26, 30, 31, 32, 34, 35, 36\}$

$\Psi_3 = \{23, 24, 25, 26, 27, 28, 29, 30, 31, 32\}$

$\Psi_4 = \{27, 28, 29, 34, 35, 36, 38, 39\}$

Indice de modification chronologique

Le tableau **5.4** regroupe les résultats obtenus. On peut remarquer que l'indice de modification chronologique d'un sommet est identique pour les trois modèles pour un voisinage donné. Ce qui est prévisible, car ce n'est que les numéros des sommets qui sont pris en compte.

Indice de modification spatial avec seuil chronologique.

— Méthode des chemins :

Pour le voisinage entrant du sommet 1 et les voisinages sortant des sommets 3 et 4, l'indice de modification est le même pour les trois modèles pour un seuil donné. Ce qui

TABLEAU 5.4 Indice de modification chronologique.

sommets	Entrant			Sortant			Voisinage		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	2	2	2	1	1	1	2	2	2
2	1	1	1	3	3	3	4	4	4
3	2	2	2	1	1	1	1	1	1
4	2	2	2	1	1	1	3	3	3

TABLEAU 5.5 Indice de modification spatial avec seuil chronologique : méthode des chemins.

Voisinage entrant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	4	4	4	2	2	2	1	1	1
2	6	3	4	6	3	2	6	1	1
3	5	3	3	4	2	2	3	1	1
4	5	3	3	4	2	2	3	1	1
Voisinage sortant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	6	3	4	6	3	2	6	1	1
2	8	4	4	7	3	3	5	1	1
3	2	2	2	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1
Voisinage entrant et sortant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	10	7	8	8	5	4	7	2	2
2	14	7	9	13	6	1	11	2	2
3	7	5	5	5	3	3	4	2	2
4	6	4	4	5	3	3	4	2	2

est prévisible, car le graphe induit est le même pour les trois modèles. Pour le voisinage entrant du sommet 3 et 4, l'indice est le même pour les modèles 1 et 2. Parfois, on peut avoir l'égalité d'indice entre les modèles 1 et 2 pour un seuil donné et une différence pour un autre seuil. C'est par exemple le cas du voisinage sortant du sommet 1.

Lorsque le seuil est grand, l'indice de modification est égal à 1 pour les voisinages entrant et les voisinages sortant des sommets avec les modèles 1 et 2. Par contre, avec le modèle initial, quelquefois, c'est différent de 1. Pour le voisinage au complet, l'indice est égal à 2 pour les modèles 1 et 2, mais supérieur à 2 pour le modèle initial.

— Méthode des chaînes :

En considérant les chaînes à la place des chemins, l'indice de modifications a nettement diminué. Ce qui indique que la longueur des modifications considérées a augmenté. De plus, lorsque le seuil est grand, l'indice de modification est égal à 1 pour les modèles 1 et 2, quelque fois aussi pour le *modèle sans perte*.

Indice de modification spatio-chronologique avec seuil chronologique

— **Cas** $k = 1$

TABLEAU 5.6 Indice de modification spatial avec seuil chronologique : méthode des chaînes.

Voisinage entrant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	3	3	3	2	2	2	1	1	1
2	6	3	4	5	3	2	5	1	1
3	4	2	2	3	2	2	3	1	1
4	4	2	2	4	2	2	3	1	1
Voisinage sortant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	6	3	4	5	3	2	5	1	1
2	7	3	3	7	3	3	5	1	1
3	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1
Voisinage entrant et sortant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	9	6	7	7	5	4	6	2	2
2	13	6	7	12	6	5	10	2	2
3	5	3	3	4	3	3	4	2	2
4	5	3	3	5	3	3	4	2	2

Le tableau **5.7** regroupe les indices de modification spatio-chronologique pour $k = 1$. On remarque que les indices de modification obtenus par les modèles 1 et 2 sont plus petits que ceux obtenus par le *modèle sans perte* initial, ce qui signifie qu'avec la méthode spatio-chronologique, la longueur des modifications considérées a augmenté. L'indice de modification obtenu par le *modèle* 2 est quelquefois légèrement supérieur à celui obtenu par le *modèle* 1. C'est le cas, par exemple, pour le voisinage entrant du sommet 2 qui est aussi le voisinage sortant du sommet 1, mais dans l'ensemble, l'indice de modification spatio-chronologique est le même pour les *modèles* 1 et 2.

— **Cas** $k = 99$

Lorsque $k = 99$, d'après le tableau **5.8**, l'indice de modification diminue, ce qui veut dire que les modifications deviennent de plus en plus longues. De plus, les indices obtenus par les trois modèles ont tendance à être identiques. Ce qui pourrait signifier que pour un certain ordre de voisinage k, les trois modèles deviennent semblables pour le calcul des indices de modifications spatio-chronologiques.

5.4.3 Quelques remarques et théorèmes

Lorsqu'on choisit un seuil tel que l'on a le sous-graphe induit au complet, alors dans ce cas, on s'attend à avoir un chemin qui passe par tous les points du sous-graphe induit par le voisinage entrant ou sortant (c'est-à-dire un chemin hamiltonien). Ce qui permet d'avoir l'indice de modification égal à 1 et ce qui signifie aussi qu'on a une seule modification. Suivant la nature des sommets du graphe de processus d'écriture, on a deux cas de figure possibles.

TABLEAU 5.7 Indice de modification spatio-chronologique avec seuil chronologique : cas $k = 1$.

Voisinage entrant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	3	3	3	3	3	3	3	3	3
2	6	3	4	6	3	4	6	3	4
3	4	2	2	4	2	2	4	2	2
4	4	2	2	4	2	2	4	2	2
Voisinage sortant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	6	3	4	6	3	4	6	3	4
2	7	3	3	7	3	3	7	3	3
3	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1
Voisinage entrant et sortant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	9	6	7	9	6	7	9	6	7
2	13	6	7	13	6	7	13	6	7
3	5	3	3	5	3	3	5	3	3
4	5	3	3	5	3	3	5	3	3

TABLEAU 5.8 Indice de modification spatio-chronologique avec seuil chronologique : cas $k = 99$.

Voisinage entrant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	2	2	2	2	2	2	1	1	1
2	1	1	1	1	1	1	1	1	1
3	2	2	2	2	2	2	1	1	1
4	2	2	2	2	2	2	1	1	1
Voisinage sortant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	1	1	1	1	1	1	1	1	1
2	3	3	3	2	2	2	1	1	1
3	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1
Voisinage entrant et sortant									
sommets	**s=1**			**s=3**			**s=99**		
	MSP	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**	**MSP**	**Modèle 1**	**Modèle 2**
1	3	3	3	3	3	3	2	2	2
2	4	4	4	3	3	3	2	2	2
3	3	3	3	3	3	3	3	3	2
4	3	3	3	3	3	3	2	2	2

1^{er}**cas :** Le sous-graphe induit au complet ne contient que des sommets d'insertions.

Dans ce cas, on trouve à chaque fois un chemin qui passe par tous les sommets. On peut alors énoncer le théorème suivant :

Théorème 5.1. *Tout graphe de processus d'écriture constitué uniquement de sommets d'insertions possède un chemin hamiltonien.*

Preuve du théorème 5.1 : La preuve est constructive, il suffit de prendre le chemin produit par la succession des sommets correspondant aux caractères du texte final. Comme le graphe n'a pas de sommet de suppression, alors chaque sommet du graphe représente un caractère du texte final. De plus, par définition du graphe de processus d'écriture, chaque caractère écrit dans le texte est représenté par un sommet. Ainsi, le chemin construit passe par tous les sommets et ne passe pas deux fois par le même sommet. D'où on a un chemin hamiltonien.

Corolaire 5.1.1. *Tout sous-graphe induit par un voisinage entrant d'un sommet constitué uniquement de sommets d'insertion admet un chemin hamiltonien.*

Preuve du corolaire 5.1.1 : Soit $G_x^{\Psi^{in}}$ le sous-graphe induit par le voisinage entrant d'un sommet x constitué uniquement de sommets d'insertion. Posons $V_x = \{x_1, x_2, ..., x_k\}$ avec $x_1 < x_2 < ... < x_k$ de $G_x^{\Psi^{in}}$. Posons G le graphe de processus d'écriture contenant $G_x^{\Psi^{in}}$, posons y le prédécesseur immédiat de x dans G. Supposons que $y = Debut$ et $x = Fin$, alors en posant $L = G_x^{\Psi^{in}} + \{y, x\}$, on obtient un graphe de processus d'écriture. Puisque $G_x^{\Psi^{in}}$ est constitué uniquement d'insertions, alors L est un graphe de processus d'écriture uniquement constitué d'insertions. D'où, d'après le théorème précédent, L possède un chemin hamiltonien qui passe par tous ses sommets, de sommet initial y et de sommet final x. En enlevant y et x du chemin, on se retrouve avec un chemin hamiltonien dans $G_x^{\Psi^{in}}$.

Il en résulte alors que tout sous-graphe induit par un voisinage entrant d'un sommet constitué uniquement de sommets d'insertion admet un chemin hamiltonien.

Une preuve similaire donne le résultat suivant :

Corolaire 5.1.2. *Tout sous-graphe induit par un voisinage sortant d'un sommet constitué uniquement de sommets d'insertion admet un chemin hamiltonien.*

Remarque 5.4.1. *Le sous-graphe induit par le voisinage d'un sommet constitué uniquement de sommets d'insertion admet deux chemins disjoints issus respectivement des chemins hamiltoniens des voisinages entrant et sortant. Ainsi, l'indice de modification peut être égal à 2 lorsqu'on choisit un seuil tel que l'on a le sous-graphe induit au complet.*

Corolaire 5.1.3. *Pour tout sommet x d'un graphe de processus d'écriture G. Si G_x^{Ψ} est constitué uniquement de sommets d'insertions, alors $G_x^{\Psi} + \{x\}$ admet un chemin hamiltonien.*

Preuve du corolaire 5.1.3 : Soit G_x^{Ψ} le sous-graphe induit par le voisinage d'un sommet x constitué uniquement de sommets d'insertions.

Posons $V_x = V_x^{in} \cup V_x^{out}$ avec $V^{in} = \{x_1, x_2, ..., x_k\}$ et $V^{out} = \{y_1, y_2, ..., y_k\}$ respectivement les sommets de $G_x^{\Psi^{in}}$ et $G_x^{\Psi^{out}}$. Posons G le graphe de processus d'écriture contenant $G_x^{\Psi} + \{x\}$. Posons y le prédécesseur immédiat de x dans G et z le successeur immédiat de x dans G. D'après le corolaire **5.1.1**, on a un chemin hamiltonien de y à x passant par tous les sommets de V_x^{in}, de même, d'après le corolaire **5.1.2**, on a un chemin hamiltonien de x à z passant par tous les sommets de V_x^{out}. Il en résulte alors un chemin hamiltonien de y à z passant par les sommets de $G_x^{\Psi} + \{x\}$.

D'où $G_x^{\Psi} + \{x\}$ admet un chemin hamiltonien.

Remarque 5.4.2. *D'après le corolaire **5.1.3**, pour que l'indice de modification soit égal à 1, il faut que tout sommet soit considéré dans le sous-graphe induit par son voisinage.*

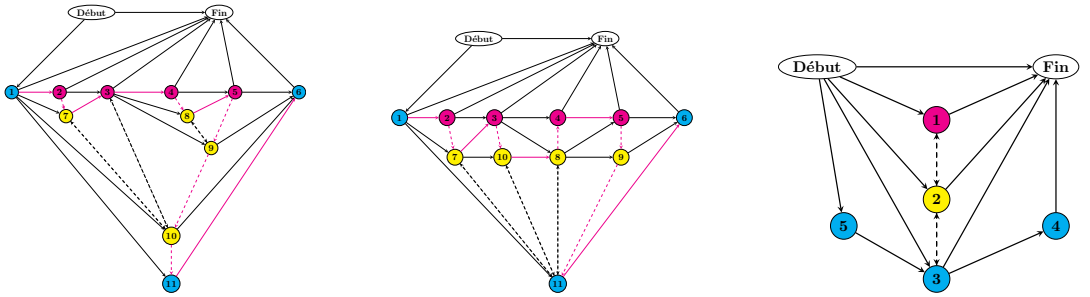
2^{ieme}**cas :** le sous-graphe induit par le voisinage au complet contient des sommets de suppressions.

Dans ce cas, on n'arrive pas toujours à trouver des chemins hamiltoniens. En effet, cela dépend du modèle choisi.

Pour le modèle initial, il n'y a pas de chemin hamiltonien possible. En effet, dans le graphe du modèle initial, certains sommets ne sont pas reliés par des arcs ni des chemins. C'est le cas des sommets d'insertion supprimés et des sommets de suppression qui les suppriment.

Pour le *modèle* 1 et 2, dans certains graphes c'est possible, mais dans d'autres non. En effet, comme le montrent les figures **5.26a** et **5.26b** du *modèle* 1 et 2 respectivement, on a un chemin hamiltonien représenté par la ligne rouge. Par contre, pour le graphe de la figure **5.26c** qui représente à la fois un graphe du *modèle* 1 et du *modèle* 2, on n'a pas de chemin hamiltonien possible.

Ainsi, lorsqu'on a des suppressions dans le graphe induit par le voisinage, il sera parfois impossible d'obtenir un indice de modification égal à 1 quel que soit le seuil chronologique que l'on considère. C'est en général le cas lorsque l'indice de modification est calculé avec la notion de modification chronologique ou la modification spatiale avec seuil chronologique. Mais par contre, lorsque l'on utilise la modification spatio-chronologique, alors la présence de l'ordre du voisinage permet de remédier à cela. En effet, il suffit de prendre k aussi grand que possible pour avoir l'indice de modification



(a) *Modèle* 1 (b) Modèle 2 (c) *Modèle* 1 et *Modèle* 2

FIGURE 5.26 Exemples de modèles 1 et 2 avec et sans chemin hamiltonien.

égal à 1.

Les deux cas de figure précédents nous permettent alors d'établir le théorème suivant :

Théorème 5.2. *Pour tout graphe de processus d'écriture G, pour tout sommet $v \in G$.*

1. (i). $idM^{in,\infty}_{cs_1}(v) = 1$ *si et seulement si* $G^{\Psi^{in}}_v$ *possède un chemin hamiltonien.*
   (ii). $idM^{out,\infty}_{cs_1}(v) = 1$ *si et seulement si* $G^{\Psi^{out}}_v$ *possède un chemin hamiltonien.*

   En d'autres termes, pour un seuil s assez grand qui prend tous les arcs de $G^{\Psi^{in}}_v$ (respectivement $G^{\Psi^{out}}_v$), alors l'indice de modification spatiale de seuil chronologique s du voisinage du sommet v est égal à 1 si et seulement si le sous-graphe induit par le voisinage entrant (respectivement sortant) de v possède un chemin hamiltonien.
2. (i). $idM^{s}_{c}(v) \leqslant idM^{in}_{c}(v) + idM^{out}_{c}(v)$
   (ii). $idM^{s}_{cs_1}(v) = idM^{in,s}_{cs_1}(v) + idM^{out,s}_{cs_1}(v)$
   (iii). $idM^{s}_{cs_2}(v) = idM^{in,s}_{cs_2}(v) + idM^{out,s}_{cs_2}(v)$
   (iv). $idM^{s,k}_{sc}(v) = idM^{in,s,k}_{sc}(v) + idM^{out,s,k}_{sc}(v)$

   Pour tous $s \geqslant 1$ et $k \geqslant 1$

Preuve du théorème 5.2 : Soit G un graphe de processus d'écriture, soit v un sommet de G.

1. Lorsque l'indice de modification spatiale du seuil chronologique entrant (respectivement sortant) est égal à 1, cela traduit qu'on a un chemin qui passe une fois par tous les sommets induits par le voisinage entrant (respectivement sortant). Ce qui signifie que $G^{\Psi^{in}}_v$ (respectivement $G^{\Psi^{out}}_v$) possède un chemin hamiltonien.

   Réciproquement, lorsque $G^{\Psi^{in}}_v$ (respectivement $G^{\Psi^{out}}_v$) possède un chemin hamiltonien,

en considérant ce chemin comme chemin de modification, alors on a une seule modification spatiale avec seuil chronologique. D'où l'indice de modification est égal à 1.

2. Pour rappel, on a $\Psi_v = \Psi_v^{in} \cup \Psi_v^{out}$ et $\Psi_v^{in} \cap \Psi_v^{out} = \emptyset$.

   En terme de cardinalité, $card(\Psi_v) = card(\Psi_v^{in}) + card(\Psi_v^{out})$

   (i). Montrer que $idM_c^s(v) \leqslant idM_c^{in}(v) + idM_c^{out}(v)$, revient à montrer qu'on peut trouver dans certains cas une modification chronologique dans Ψ_v qui contient à la fois des sommets du voisinage entrant (Ψ_v^{in}) et sortant (Ψ_v^{out}) de v.

   Soit $\{x, x+1, \ldots, x+l-1\}$ une modification chronologique du voisinage entrant de v. Soit $\{y, y+1, \ldots, y+k-1\}$ une modification chronologique du voisinage sortant de v.

   — Si $x + l = y$ alors $\{x, x+1, \ldots, x+l-1, y, x+1, \ldots, y+k-1\}$ est une modification chronologique dans Ψ_v avec des sommets dans Ψ_v^{in} et Ψ_v^{out}

   — Si $y + k = x$, alors $\{y, y+1, \ldots, y+k-1, x, x+1, \ldots, x+l-1\}$ est une modification chronologique dans Ψ_v avec des sommets dans Ψ_v^{in} et Ψ_v^{out}

   (ii). Pour ii), iii) et iv), le principe est le même. Comme les ensembles Ψ_v^{in} et Ψ_v^{out} sont disjoints, les chemins que l'on obtient dans les sous-graphes induits $G_v^{\Psi^{in}}$ et $G_v^{\Psi^{out}}$ sont aussi disjoints. De plus, comme il n'y a pas d'arc entre des sommets du voisinage entrant et du voisinage sortant dans le sous-graphe induit G_v^{Ψ}, alors il ne peut pas y avoir de chemin entre les deux voisinages. Par conséquent, le nombre total de chemins est égal à la somme du nombre de chemins obtenus dans $G_v^{\Psi^{in}}$ et $G_v^{\Psi^{out}}$.

5.5 Liens entre les communautés de modifications et les indices de modifications

Dans cette section, nous présentons quelques liens qui existent entre la notion de communauté de modification et d'indice de modification.

En appliquant l'algorithme **3** de décomposition du graphe de processus d'écriture en communautés du chapitre **4** au graphe de processus d'écriture de la Figure **5.15a**, on obtient les communautés suivantes :

$C_1 = \{1, 2, 3, 4\}$,	$C_5 = \{16, 17\}$,	$C_9 = \{27\}$,	$C_{13} = \{33\}$
$C_2 = \{5, 6, 7, 8, 9, 10\}$,	$C_6 = \{18, 19\}$,	$C_{10} = \{28, 29\}$,	$C_{14} = \{34, 35, 36\}$
$C_3 = \{11\}$,	$C_7 = \{20, 21, 22\}$,	$C_{11} = \{30, 31\}$,	$C_{15} = \{37\}$
$C_4 = \{12, 13, 14, 15\}$,	$C_8 = \{23, 24, 25, 26\}$,	$C_{12} = \{32\}$,	$C_{16} = \{38, 39\}$

Pour rappel, on a obtenu les voisinages suivants pour les sommets $1, 2, 3$ et 4 respectivement :

$\Psi_1^{in} = \{16, 17, 18, 19, 20, 21, 22, 33\}$ $\Psi_1^{out} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$
$\Psi_2^{in} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ $\Psi_2^{out} = \{23, 24, 25, 26, 30, 31, 32, 34, 35, 36\}$
$\Psi_3^{in} = \{23, 24, 25, 26, 30, 31, 32\}$ $\Psi_3^{out} = \{27, 28, 29\}$
$\Psi_4^{in} = \{27, 28, 29, 34, 35, 36\}$ $\Psi_4^{out} = \{38, 39\}$

On peut alors écrire :

$\Psi_1^{in} = C_5 \cup C_6 \cup C_7 \cup C_{13}$ $\Psi_1^{out} = C_2 \cup C_3 \cup C_4$
$\Psi_2^{in} = C_2 \cup C_3 \cup C_4$ $\Psi_2^{out} = C_8 \cup C_{11} \cup C_{12} \cup C_{14}$
$\Psi_3^{in} = C_8 \cup C_{11} \cup C_{12}$ $\Psi_3^{out} = C_9 \cup C_{10}$
$\Psi_4^{in} = C_9 \cup C_{10} \cup C_{14}$ $\Psi_4^{out} = C_{16}$

Ce qui veut dire que les voisinages des sommets peuvent s'écrire comme une réunion des communautés de modifications obtenues au chapitre 4. De plus, d'après le **5.5** pour le *modèle* 1, on observe que les indices de modifications spatiaux avec seuil chronologique $s = 1$ (méthode des chemins) des sommets 1, 2, 3 et 4 ont les valeurs :

$idM_{cs_1}^{in,1}(1) = 4$ $idM_{cs_1}^{out,1}(1) = 3$
$idM_{cs_1}^{in,1}(2) = 3$ $idM_{cs_1}^{out,1}(2) = 4$
$idM_{cs_1}^{in,1}(3) = 3$ $idM_{cs_1}^{out,1}(3) = 2$
$idM_{cs_1}^{in,1}(4) = 3$ $idM_{cs_1}^{out,1}(1) = 1$

En d'autres termes, le nombre de communautés de modifications composant le voisinage d'un sommet correspond à l'indice de modification spatial avec seuil chronologique égal à 1 du sommet considéré.

Formellement, on peut écrire le Théorème suivante :

Théorème 5.3. *Soit G un graphe de processus d'écriture et soit $C = \{C_1, C_2, \ldots, C_l\}$ la partition en l communautés de modification du graphe G. Pour tout sommet v de G, il existe $K_0 \subseteq \{1, 2, \ldots, l\}$ et $J_0 \subseteq \{1, 2, \ldots, l\}$ tels que :*

$$\Psi_v^{in} = \bigcup_{k \in K_0} C_k \quad \texttt{et} \quad \Psi_v^{out} = \bigcup_{j \in J_0} C_j$$

De plus, pour le modèle 1*, on a :*

$$idM_{cs_1}^{in,1}(v) = card(K_0) \quad \texttt{et} \quad idM_{cs_1}^{out,1}(v) = card(J_0)$$

Preuve du théorème 5.3 : Soit G un graphe de processus d'écriture, soit $C = \{C_1, C_2, \ldots, C_l\}$ la partition en l communautés de modification du graphe G. Soit v un sommet de G.

D'une part, d'après l'algorithme de construction des communautés de modifications, chaque sommet appartient à une unique communauté. Ainsi, pour tout sommet d'un voisinage entrant (respectivement sortant) du sommet v, on peut trouver une unique communauté de modifications contenant le sommet. En d'autres termes :

— Pour tout $x \in \Psi_v^{in}$, il existe k_x, $1 \leqslant k_x \leqslant l$ tel que $x \in C_{k_x}$. En posant, $K_0 = \{k_x, x \in \Psi_v^{in}\}$ on a $\Psi_v^{in} \subseteq \bigcup_{k \in k_0} C_k$.

— Pour tout $x \in \Psi_v^{out}$ il existe j_x, $1 \leqslant j_x \leqslant l$ tel que $x \in C_{j_x}$. En posant, $J_0 = \{j_x, x \in \Psi_v^{out}\}$ on a $\Psi_v^{out} \subseteq \bigcup_{j \in J_0} C_j$.

D'autre part, l'algorithme de construction des communautés de modification permet aussi de constater que chaque communauté est composée des sommets d'insertions ou de suppressions effectuées sans déplacer le curseur d'un endroit à l'autre dans le texte. Ce qui veut dire que chaque communauté est l'ensemble des prédécesseurs (respectivement successeurs) immédiats d'un unique sommet. En d'autres termes, pour toute communauté C_k, il existe un unique sommet u tel que $C_k \subseteq N_1^{in}(u)$ (respectivement, $C_k \subseteq N_1^{out}(u)$). Ainsi, si deux sommets font partie d'une même communauté, alors ils sont dans le même voisinage.

Soit $y \in \bigcup_{k \in K_0} C_k$, alors il existe $k_y \in K_0$ tel que $y \in C_{k_y}$. Or, par définition de K_0, $k_y \in K_0$ implique qu'il existe $x \in \Psi_v^{in}$ tel que $x \in C_{k_y}$ ce qui veut dire que $x, y \in C_{k_y}$. Comme x et y sont dans la même communauté alors ils sont dans le même voisinage. Ainsi, $y \in \Psi_v^{in}$ car $x \in \Psi_v^{in}$. On a alors $\bigcup_{k \in K_0} C_k \subseteq \Psi_v^{in}$. De la même façon on obtient $\bigcup_{j \in J_0} C_j \subseteq \Psi_v^{out}$.

D'où $\Psi_v^{in} = \bigcup_{k \in K_0} C_k$ et $\Psi_v^{out} = \bigcup_{j \in J_0} C_j$ pour $K_0 = \{k_x, x \in \Psi_v^{in}\}$ et $J_0 = \{j_x, x \in \Psi_v^{out}\}$.

Étant donné que, dans le *modèle* 1, tous les sommets d'un même ensemble de successeurs immédiats sont reliés par un chemin, alors chaque communauté avec un indice dans K_0 (respectivement J_0) représente un chemin avec le *modèle* 1. Or l'indice de modification spatiale de seuil chronologique $s = 1$ du voisinage entrant $idM_{cs_1}^{in,1}(v)$ (respectivement sortant $idM_{cs_1}^{out,1}(v)$) compte le nombre de chemins présent dans le voisinage entrant (respectivement sortant). Par conséquent, $idM_{cs_1}^{in,1}(v) = card(K_0)$ (respectivement, $idM_{cs_1}^{out,1}(v) = card(J_0)$). □

Fusion par proximité : Pour rappel, la fusion des communautés de modifications par proximité consiste à fusionner deux communautés successives si le mouvement du curseur entre les deux communautés successives ne dépasse pas un certain seuil. En faisant varier les seuils de proximité, on constate que les communautés deviennent de plus en plus grandes et que leur nombre diminue. Ce qui a pour effet que les voisinages des sommets sont contenus dans des communautés de plus en plus grandes. On peut alors faire la remarque suivante :

Remarque 5.5.1. *Soit G un graphe de processus d'écriture, soit $\Omega_p = \{\omega_1^p, \omega_2^p, \ldots, \omega_l^p\}$ la*

partition du graphe G en l communautés de modification, avec fusion de communauté, pour le seuil de proximité p.

1. *Pour tout sommet v de G, il existe $K \subseteq \{1, 2, \ldots, l\}$ et $J \subseteq \{1, 2, \ldots, l\}$ tel que :*

$$\Psi_v^{in} \subseteq \bigcup_{k \in K} \omega_k^p \ \texttt{et} \ \Psi_v^{out} \subseteq \bigcup_{j \in J} \omega_j^p$$

   Autrement dit, pour un seuil de proximité donné, on peut toujours trouver une réunion de communautés de modifications contenant le voisinage entrant (sortant) d'un sommet donné.

2. *Pour tout sommet v de G, il existe $p_0 \geqslant 1$ et $K^{'}, K^{''} \subseteq \{1, 2, \ldots, l\}$ tels que :*

$$\Psi_v^{in} = \bigcup_{k \in K^{'}} \omega_k^{p_0 - 1} \ \texttt{et} \ \Psi_v^{in} \subset \bigcup_{k \in K^{''}} \omega_k^{p_0}$$

   En d'autres termes, on peut toujours trouver un seuil de proximité p_0 à partir duquel le voisinage entrant d'un sommet n'est plus égal à une réunion de communautés de modification. Autrement dit, $p_0 - 1$ représente le déplacement maximal du curseur dans le voisinage du sommet considéré.

3. *On a un résultat similaire pour le voisinage sortant. Pour tout sommet v de G, il existe $p_0^{'} \geqslant 1$, il existe $J^{'}, J^{''} \subseteq \{1, 2, \ldots, l\}$ tel que :*

$$\Psi_v^{out} = \bigcup_{j \in J^{'}} \omega_j^{p_0^{'} - 1} \ \texttt{et} \ \Psi_v^{out} \subseteq \bigcup_{j \in J^{''}} \omega_j^{p_0^{'}}$$

La valeur de p_0 (respectivement $p_0^{'}$) permet d'avoir une idée des rapprochements des modifications dans le voisinage entrant (respectivement sortant) d'un sommet. En effet, une petite valeur de p_0 (respectivement $p_0^{'}$) signifie que les modifications successives effectuées dans le voisinage entrant (sortant) sont très rapprochées les unes des autres. Par contre, une grande valeur de p_0 (respectivement $p_0^{'}$) indique des modifications plus éloignées les unes des autres. On obtient un résultat semblable avec la fusion par taille des communautés.

Fusion par taille des modifications : Comme mentionné dans le chapitre 4, la fusion par taille des modifications consiste à regrouper deux modifications successives lorsque leurs tailles sont inférieures à un certain seuil.

Remarque 5.5.2. *Soit G un graphe de processus d'écriture, soit $\delta_t = \{\delta_1^t, \delta_2^t, \ldots, \delta_l^t\}$ la partition du graphe G en l communautés de modifications, avec fusion de communautés, pour le seuil de taille t.*

1. *Pour tout sommet v de G, il existe $K \subseteq \{1, 2, \ldots, l\}$ et $J \subseteq \{1, 2, \ldots, l\}$ tel que :*

$$\Psi_v^{in} \subseteq \bigcup_{k \in K} \delta_k^t \ \texttt{et} \ \Psi_v^{out} \subseteq \bigcup_{j \in J} \delta_j^t$$

2. *Pour tout sommet v de G, il existe $t_0 \geqslant 1$ et $K^{'}, K^{''} \subseteq \{1, 2, \ldots, l\}$ tel que :*

$$\Psi_v^{in} = \bigcup_{k \in K^{'}} \delta_k^{t_0 - 1} \ \texttt{et} \ \Psi_v^{in} \subset \bigcup_{k \in K^{''}} \delta_k^{t_0}$$

   Ce qui veut dire, qu'on peut trouver un seuil de taille t_0 à partir duquel le voisinage entrant d'un sommet n'est plus égal à une réunion de communautés de modification. En d'autres termes, $t_0 - 1$ représente la taille maximale des modifications dans le voisinage du sommet considéré.

3. *On a un résultat similaire pour le voisinage sortant. Pour tout sommet v de G, il existe $t_0^{'} \geqslant 1$, il existe $J^{'}, J^{''} \subseteq \{1, 2, \ldots, l\}$ tel que :*

$$\Psi_v^{out} = \bigcup_{j \in J^{'}} \delta_j^{t_0^{'} - 1} \ \texttt{et} \ \Psi_v^{out} \subseteq \bigcup_{j \in J^{''}} \delta_j^{t_0^{'}}$$

La valeur t_0 ($t_0^{'}$) donne un aperçu en termes des modifications effectuées dans le voisinage du sommet. Par exemple, si la valeur de t_0 ($t_0^{'}$) est petite, alors cela signifie que les modifications sont successives et de petite taille. En revanche, une grande valeur de t_0 ($t_0^{'}$) laisse présager des modifications de grande taille.

En résumé, les communautés de modifications et les indices de modifications sont étroitement liés. L'utilisation de la notion de communauté de modifications dans le voisinage d'un sommet montre que la notion d'indice de modification peut, dans certains cas, être obtenue par la notion de communauté de modifications. Elle permet, grâce à la notion de fusion par proximité par taille, de mieux analyser les modifications effectuées au voisinage d'un sommet.

5.6 Conclusion

Ce chapitre a permis de poser les bases de l'indice de modification, qui renseigne sur le nombre de modifications effectuées à un endroit précis du texte. Pour définir cet indice, nous avons introduit la notion de voisinage d'un sommet et précisé la notion de modification. De plus, nous avons défini deux nouveaux modèles de processus d'écriture issus du *modèle sans perte* pour mieux calculer l'indice de modification. Nous avons également présenté un algorithme permettant de calculer cet indice. Enfin, nous avons présenté quelques liens entre l'indice de modification et les communautés de modifications définies au chapitre précédent.

CHAPITRE 6 APPLICATIONS

Dans ce chapitre, nous présentons quelques utilisations possibles des notions que nous avons définies dans les deux chapitres précédents.

Dans la section 6.1, il est question de présenter les applications possibles des communautés de modifications, notamment dans l'analyse des processus d'écriture. Dans la section 6.2, nous présentons les applications de l'indice de modifications pour la visualisation du processus d'écriture et pour la mesure de centralité d'un graphe de processus d'écriture.

6.1 Applications des communautés de modifications

Cette section présente quelques utilisations possibles des communautés de modifications pour l'analyse des processus d'écriture. Certaines de ces applications sont des preuves de concept. Dans la sous-section 6.1.1, nous expliquons comment la notion de communauté de modification peut être utilisée pour mettre en évidence le lien entre les modifications réalisées et le niveau d'étude. Dans la sous-section 6.1.2, il est question de montrer comment la combinaison des communautés de modification et des mesures de lisibilité peut aider à classer les processus d'écriture par niveau.

Dans cette section nous utilisons les données tiré du corpus *Pro-TEXT* réalisé par Miletić et al. [78]. Pour rappel, comme mentionné dans l'introduction, la version en libre accès que nous utilisons contient 45 textes (en français) d'écoliers, répartis en trois groupes `CE2`, `CM2` et `C6`. Le tableau suivant présente un résumé des données :

	`Primaire` 3	`Primaire` 5	`Secondaire` 1
`Argumentatif`	9	8	6
`Narratif`	10	5	7

6.1.1 Influence du niveau sur les quantités de modifications

Dans cette sous-section, nous illustrons comment les communautés de modifications peuvent permettre de mettre en exergue le lien qui peut exister entre les niveaux d'étude et la quantité de modifications effectuées. Comme défini précédemment, les communautés de modifications permettent de regrouper les actions réalisées successivement de manière chronologique et spatiale. De plus, le nombre de communautés de modifications traduit aussi le nombre de modifications effectuées. Ainsi, un graphe avec un grand nombre de communautés de modifications correspond à un processus d'écriture comportant plusieurs modifications. On peut

alors choisir de représenter le niveau d'étude, le nombre de communautés, le nombre de sommets et le nombre d'arcs sur un même graphique pour bien illustrer leurs interactions. Pour cela, on adopte le principe suivant :

- Le nombre de sommets du graphe du processus d'écriture est placé sur l'axe des abscisses.
- Le nombre d'arcs du graphe du processus d'écriture est placé sur l'axe des ordonnées.
- Chaque processus est représenté par un point.
- La couleur d'un point indique le niveau d'étude.
  - le bleu pour le secondaire 1
  - l'orange pour le primaire 3
  - le vert pour le primaire 5
- La taille d'un point correspond au nombre de modifications (communautés) du processus d'écriture.

Pour le corpus considéré, on obtient la Figure **6.1**. On a donc une vision globale des différents processus d'écriture. On remarque que le plus grand graphe est obtenu par un élève du secondaire, et le plus petit par un élève du primaire 3, les graphes des primaires 5 se retrouvent entre les deux. De plus, les graphes du secondaire 1 ont le plus grand nombre de communautés de modifications, suivis par ceux des primaire 5. On retrouve aussi quelques élèves de primaire 3 avec plus de dix modifications, ainsi que des élèves du secondaire 1 et des primaire 5 avec moins de cinq modifications. Au vu de cela, on peut dire que plus la taille du graphe et le nombre de communautés sont grands, plus le processus d'écriture correspond à celui d'un écolier de niveau supérieur. Autrement dit, dans ces données, les écoliers de niveau plus élevé écrivent et modifient davantage.

6.1.2 Communauté de modifications et mesure de lisibilité

Dans cette sous-section, nous présentons un exemple d'utilisation de la variation de la lisibilité durant le processus d'écriture pour classifier les processus d'écriture. Pour l'évaluation de la lisibilité, nous avons retenu six mesures de lisibilité. Notamment, le score de Dale-Chall (`Dcr`) [31], l'indice de Flesch (`Fre`) [41], l'indice de Gunning (`Gfg`) [55], l'indice de lisibilité automatisé (`Ari`) [104], l'indice de Coleman et Liau (`Cli`) [29], et les mots difficiles (`Dw`) (pour plus de précisions, voir le chapitre "Outils et méthodes").

Transformation des processus d'écriture en série chronologique des valeurs de la lisibilité :

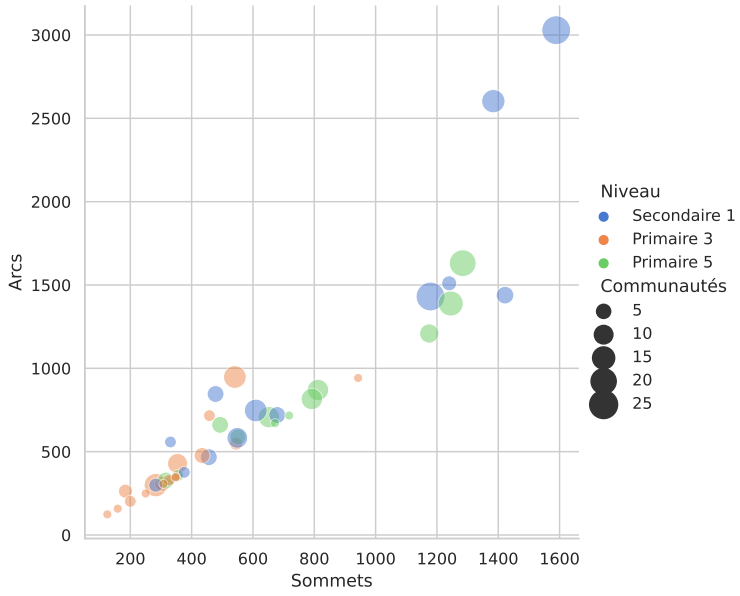


FIGURE 6.1 Influence du niveau sur les quantités de modifications.

Nous partons du principe que, lorsqu'on écrit un texte, on commence avec une page vide pour obtenir un texte final en réalisant des modifications successives dans différentes parties du texte. D'après le chapitre 3, on peut retrouver ces modifications en utilisant la notion de communauté de modification. De plus, les communautés de modifications permettent également d'obtenir la version du texte après chaque modification. Ainsi, en considérant les versions du texte après chaque modification, on se retrouve avec un ensemble de versions pour chaque processus d'écriture. Pour un processus d'écriture donné, cela peut s'illustrer comme sur la Figure **6.2**.

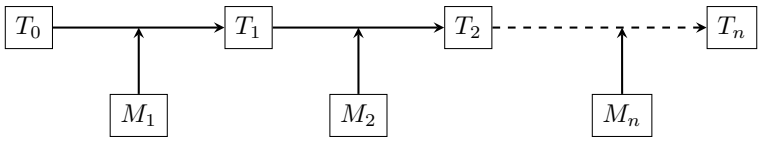


FIGURE 6.2 Évolution du texte.

Où :

— M_i représente la $i^{\text{ème}}$ modification apportée au texte.

— T_i représente la version du texte après la $i^{\text{ème}}$ modification.

Ainsi, $T_0, T_1, ..., T_n$ est l'ensemble des différentes versions successives du texte. En posant l_i comme la valeur de la lisibilité du texte T_i, avec $0 < i < n$, l'ensemble $l_0, l_1, ..., l_n$ représente l'ensemble des valeurs de la lisibilité obtenues au cours du processus d'écriture du texte.

Puisque ces modifications suivent une chronologie, l'ensemble $l_0, l_1, ..., l_n$ peut être considéré comme une série chronologique des valeurs de la lisibilité.

On a donc, pour chaque processus d'écriture donné, une série chronologique des valeurs de la lisibilité. Ainsi, pour un corpus donné (ensemble de plusieurs processus d'écriture), on obtient un ensemble de séries chronologiques. On peut alors utiliser les méthodes de classification ou de clustering des séries chronologiques, soit pour classer, soit pour créer des clusters dans cet ensemble de séries chronologiques. Cela peut, en quelque sorte, permettre de comparer les processus d'écriture selon la variation de leurs lisibilités. Nous essayerons d'illustrer ce concept sur le corpus utilisé dans la sous-section précédente.

De ce fait, nous réalisons une classification en utilisant la méthode des k plus proches voisins, avec, comme mesure de similarité, le Dynamic Time Warping [97], qui est une mesure de similarité entre séries chronologiques. En d'autres termes, une série chronologique est classée selon le résultat majoritaire des statistiques de classes d'appartenance de ses k plus proches voisins. Pour démontrer la faisabilité du concept, nous avons utilisé la fonction `KNeighborsTimeSeriesClassifier` du module `tslearn` de `Python`, en faisant varier le nombre de voisins considérés. Comme nous ne disposons que de 45 textes, nous avons choisi au hasard 36 textes pour entraîner le modèle et 9 pour le tester (ceci étant uniquement pour illustrer la faisabilité de ce concept). Le tableau 6.1 donne le taux de succès obtenu par chacune des méthodes de lisibilité et pour chaque nombre de voisins considérés.

Tableau 6.1 Taux de réussite de la classification

Nombre de voisins	Dcr	Fre	Gfog	Ari	Cli	Dw
2	0.3	**0.7**	0.4	0.3	0.3	**0.7**
3	0.2	0.4	0.4	0.3	**0.6**	**0.6**
4	0.3	0.4	0.3	0.3	**0.6**	**0.6**
5	0.2	0.4	0.3	0.4	0.4	**0.4**

On remarque que la méthode utilisant comme variable principale les mots difficiles présents dans le texte obtient le plus grand taux de réussite, dans le classement des processus d'écriture suivant le niveau d'étude, pour les différents nombres de voisins choisis. La méthode de `Fre` et celle de `Cli` obtiennent respectivement un et deux meilleur(s) score(s). Par contre, la méthode de Dale-Chall obtient les moins bons scores. Cela signifie que la méthode de comptage des mots difficiles est la plus appropriée pour classer les processus d'écriture de ce corpus. Cela peut s'expliquer par le fait que le corpus comprend essentiellement les processus d'écriture des écoliers qui sont dans les débuts de leur apprentissage. En effet, dans cette phase, les écoliers d'une même classe apprennent les mêmes mots difficiles, qu'ils sont susceptibles d'utiliser tous lors d'une rédaction.

Ce petit exemple illustre comment les mesures de lisibilité peuvent être utilisées pour la classification des processus d'écriture.

6.2 Applications des indices de modifications

Cette section présente deux applications possibles de l'indice de modifications. En effet, l'indice de modifications permet de trouver les sommets les plus influents suivant le nombre de modifications effectuées près du sommet. Ainsi, en plus de donner le nombre de modifications, cet indice peut permettre de classer les sommets suivant l'ordre de grandeur des modifications.

Dans la sous-section 6.2.1, nous présentons une représentation synthétique du processus d'écriture sous forme de graphe, qui prend en compte le nombre de modifications, le type de modification et la localisation de la modification. Dans la sous-section 6.2, nous comparons le classement de sommets obtenu suivant l'indice de modifications avec d'autres classements de sommets issus des mesures de centralité dans un graphe. Le but est de voir dans quelle mesure les indices de modifications peuvent être utilisés comme mesure de centralité dans un graphe de processus d'écriture.

6.2.1 Représentation synthétique du processus d'écriture

Dans cette sous-section, il est question de présenter une application de l'indice de modifications pour la représentation d'un processus d'écriture.

On part du principe selon lequel, lorsqu'on écrit quelque chose pour la première fois, c'est la base du texte. En effet, en considérant la construction du graphe (*modèle sans perte*), lorsqu'on écrit à la fin du texte, le sommet est relié au sommet `Fin`. Ainsi, toute insertion qui n'est pas reliée au sommet `Fin` est nécessairement une insertion qui modifie le texte déjà écrit et donc une modification.

Au vu de ce principe, nous considérons comme base de notre représentation les arcs du graphe de processus d'écriture dont les sommets sont reliés au sommet `Fin` (c'est-à-dire les sommets d'insertions réalisées à la fin du texte courant). Parmi ces arcs, nous considérons les arcs dont les sommets sont des insertions. Posons F l'ensemble de ces arcs.

$$F = \{(u, v) \in A | (u, \texttt{Fin}), (v, \texttt{Fin}) \in A, u, v \in I\}$$

Nous construisons un nouveau graphe K composé des sommets d'insertions reliés au sommet `Fin` et qui a `F` comme ensemble d'arcs. Pour chaque arc $(u, v) \in F$, nous créons un sommet $a_{u,v}$ et nous ajoutons les arcs $(u, a_{u,v})$ et $(a_{u,v}, v)$ au graphe K. Les sommets $a_{u,v}$ représentent

les modifications effectuées entre les sommets u et v. Suivant le cas où l'on se trouve $a_{u,v}$ peut représenter les modifications entrantes du sommet v ou les modifications sortant du sommet u.

Pour visualiser le graphe K, on adopte le principe suivant :

- Les sommets $a_{u,v}$ qui représentent les modifications entre les sommets u et v ont :
  - Une taille proportionnelle aux nombres de modifications.
  - Une couleur correspondant à la longueur moyenne des modifications effectuées (en termes du nombre de caractères). La couleur varie du blanc (plus petite longueur) au noir (plus grande longueur).
- Les autres sommets (sommets d'insertion) ont :
  - Tous une même taille standard.
  - Une couleur en fonction du type de sommet. Soit bleu pour les sommets d'insertions présents à la fin du texte et rouge pour les sommets d'insertions supprimées.
- Le sommet de `Début` est représenté par le sommet de numéro -1 et a une couleur différente et une taille standard.

La Figure **6.3** donne une illustration du graphe K pour les indices de modifications chronologiques (voisinage entrant) du processus d'écriture de l'exemple de la figure **5.15a** :

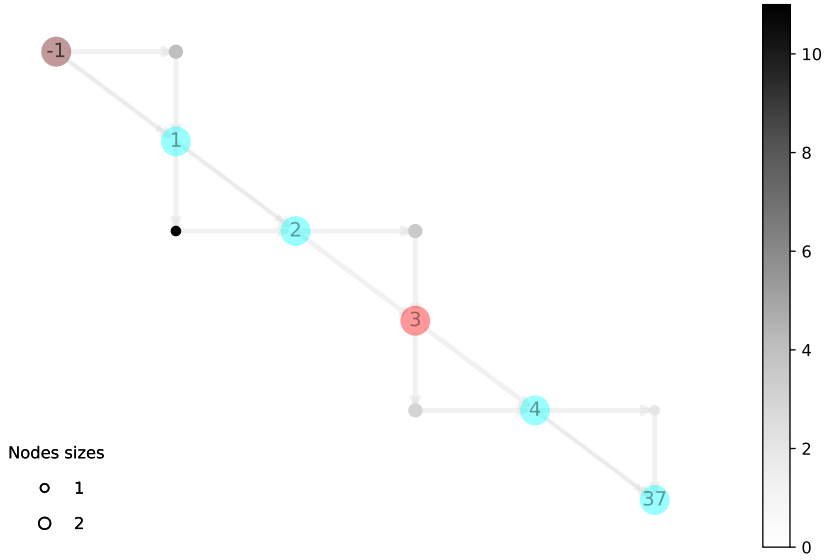


FIGURE 6.3 Graphe K (Indice de Modifications chronologique).

On observe qu'on a 2 modifications entrant aux sommets 1, 3 et 4 avec une longueur moyenne respective de 4, 3 et 3. Par contre, les sommets 2 et 37 ont chacun une seule modification chronologique de longueur respective 11 et 2.

On peut aussi visualiser le graphe K pour les indices de modifications avec seuil chronologique ou les indices de modifications spatio-chronologique. Par exemple, la Figure **6.4** est une représentation des indices de modifications avec seuil chronologique (voisinage entrant) de l'exemple de la figure **5.15a** :

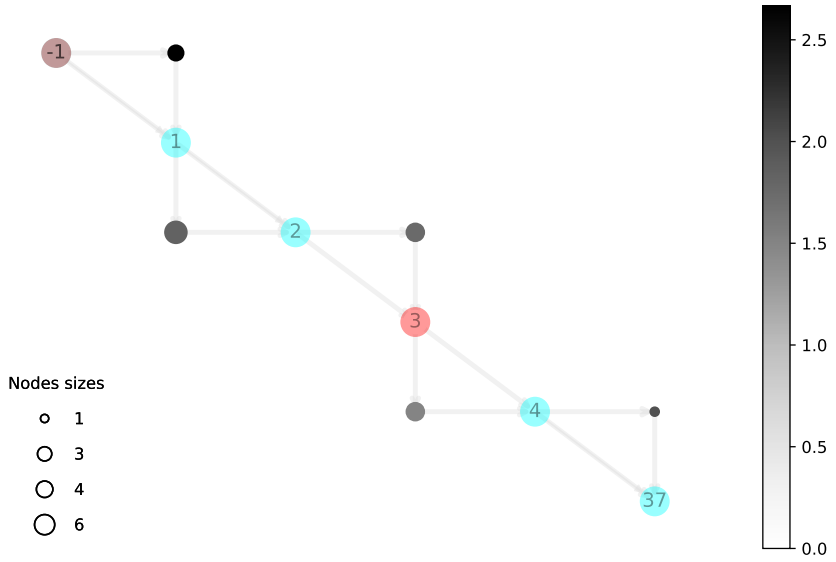


FIGURE 6.4 Graphe K (Indice de Modifications spatiale avec seuil chronologique).

On remarque que les sommets 4 et 3 ont 4 modifications spatiales avec seuil chronologique ($s = 1$). Ces modifications ont une longueur moyenne autour de $1,5$ caractère. Le sommet 37 a quant à lui un indice de 1 avec une longueur de 2 caractères. Le sommet 1 a pour sa part un indice de 3 avec une longueur moyenne de $2,7$ caractères. Enfin, le sommet 2 a un indice de 6 avec une longueur moyenne de $1,8$ caractère.

Ainsi, cette visualisation permet de connaitre les parties de texte les plus modifiées et la longueur moyenne des modifications réalisées dans différentes parties du texte. De plus, en associant deux visualisations suivant des indices de modifications différents, on obtient encore une idée plus concise de la modification effectuée. Par exemple, en considérant à la fois l'indice de modifications chronologique et l'indice de modifications spatial avec seuil chronologique. On peut dire que le voisinage entrant du sommet 2 est modifié une fois de manière chronologique, mais 6 fois de manière spatiale. Autrement dit, le rédacteur a fait d'un coup des modifications dans cet espace, mais en se déplaçant 6 fois ça et là, en insérant ou supprimant un à deux caractères en moyenne sans sortir de l'espace du texte délimité par les caractères du sommet 1 et 2. Ce qui reflète bien ce qui s'est passé lors du processus d'écriture du texte considéré.

En résumé, cette représentation à base des indices de modifications permet d'avoir une vision synthétique d'un processus d'écriture.

6.2.2 Indice de modification comme mesure de centralité d'un graphe de processus d'écriture

L'indice de modifications permet d'évaluer, pour chaque sommet d'un graphe de processus d'écriture, le nombre de modifications réalisées aux alentours du sommet. Dans cette sous-section, nous essayons de voir dans quelle mesure l'indice de modifications peut être utilisé comme mesure de centralité du graphe de processus d'écriture. Afin de vérifier l'efficacité de l'indice de modifications pour mesurer l'importance des sommets, nous comparons la méthode proposée avec d'autres mesures de centralité, notamment la centralité de degré (`CD`), la centralité de Katz (`CK`), la centralité d'intermédiarité (`CI`), la centralité de proximité (`CP`) et PageRank (`PgR`) (tous définis au chapitre **3**). Ces centralités ont chacune en commun quelques traits caractéristiques avec notre méthode, par exemple la considération des voisins (`CD`) et des chemins (`CK`, `PgR`, `CI`, `CP`).

Pour comparer les performances de notre indice, nous étudions l'importance des sommets pour la connectivité du graphe de processus en supprimant sélectivement les sommets dans l'ordre décroissant de la valeur de l'indice des sommets. Nous évaluons alors trois différentes métriques souvent utilisées dans la littérature pour comparer les performances des algorithmes de classement, notamment la sensibilité, la robustesse et l'efficacité (cf. chapitre **3**). Comme nous avons défini plusieurs indices de modifications, nous essayons tout d'abord de comparer le classement des sommets obtenu suivant chacun des indices, dans le but de pouvoir évaluer la concordance des différents classements.

Concordance des classements des sommets suivant les indices de modifications

Pour évaluer la concordance, nous avons simulé 10 graphes de processus d'écriture, de taille différente, comme présenté dans le tableau suivant :

`Tailles`	5	10	25	50	100	250	500	1000	1500	2000
`Nombre de Graphes`	1	1	1	1	1	1	1	1	1	1

Nous utilisons ces graphes tout simplement pour illustrer qu'il y a des différences entre notre méthode et les autres mesures de centralité.

Pour un seuil chronologique s donné et un ordre de voisinage k (utilisé pour les modifications spatio-chronologiques), nous comparons le classement des sommets obtenu avec les différentes

méthodes. Le tableau **6.2** suivant présente le pourcentage de graphes pour lesquels on a obtenu le même classement des sommets :

Tableau 6.2 Pourcentage de concordance des classements des sommets

	$s=1$, $k=1$, **MSP, 1 et 2 (%)**			$s=1$, $k=2$ **MSP, 1 et 2 (%)**			$s=2$, $k=1$ **MSP, 1 et 2 (%)**		
	MC	**MSSC**	**MSC**	**MC**	**MSSC**	**MSC**	**MC**	**MSSC**	**MSC**
MC	-	10	10	-	10	10	-	10	10
MSSC	10	-	100	10	-	30	10	-	90
MSC	10	100	-	10	30	-	10	90	-

On remarque tout d'abord que, pour les seuils s et k fixés, la concordance entre les méthodes est la même pour les trois modèles (*modèle sans perte* (`MSP`), *modèle* 1 et *modèle* 2). Lorsque les seuils changent, la concordance change aussi, mais pas entre toutes les méthodes. Par exemple, la concordance est de 10% pour tous les seuils choisis, entre la méthode de modification chronologique (`MC`) et les méthodes de modification spatiale avec seuil chronologique (`MSSC`) et les modifications spatio-chronologiques (`MSC`). Cette concordance est réalisée avec le graphe de taille 5.

Pour $s = 1$ et $k = 1$, les méthodes `MSC` et `MSSC` s'accordent à 100%. Lorsque $s = 1$ et que k passe de 1 à 2, la concordance chute à 30% (concordance réalisée avec les graphes de taille 5, 10 et 25). Par contre, elle passe à 90% (on n'a pas de concordance uniquement pour le graphe de taille 50) lorsque $k = 1$ et que s passe de 1 à 2. Ainsi, la concordance entre les méthodes `MSC` et `MSSC` dépend des seuils choisis.

On a aussi remarqué, lors des expériences sur les graphes simulés, qu'il n'y a aucune concordance entre nos méthodes et les autres mesures de centralité, sauf parfois une certaine concordance partielle sur certains graphes. Cela signifie que nos méthodes produisent des ordres d'importance différents de ceux des autres méthodes. Il est donc important de voir comment ces méthodes agissent sur la connectivité du graphe.

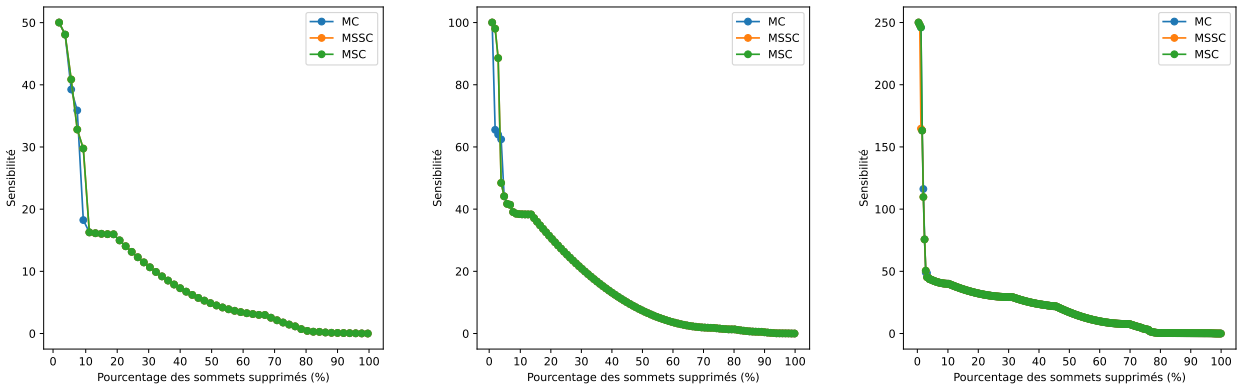
Comparaison des performances

Pour évaluer les performances, nous utilisons principalement les graphes de processus d'écriture simulés précédemment pour lesquels il n'y a pas de concordance entre les méthodes. Comme les seuils $s = 1$ et $k = 2$ sont les seuils pour lesquels il y a eu moins de concordances, alors on considère ces seuils. En prenant en compte cela, nous avons choisi pour les expériences les graphes de taille 50, 100 et 250. Pour chacun des graphes, nous évaluons les trois indicateurs de performance définis ci-dessous :

Sensibilité : Cette comparaison a pour objectif de voir comment la connectivité change dans le graphe pour les différentes méthodes lorsqu'on supprime les sommets un à un

suivant l'ordre décroissant d'importance (du sommet le plus important au sommet le moins important).

Tout d'abord, la figure **6.5** nous montre l'allure des courbes de sensibilité des trois méthodes de modification lorsque des sommets sont supprimés un à un. Nous pouvons alors constater que les trois méthodes ont les mêmes pics et que l'allure des courbes est similaire (dans certains cas, la courbe orange est complètement identique à celle verte), mais `MSC` descend un peu plus rapidement que les autres. Ainsi, suivant la sensibilité, la méthode `MSC` est légèrement préférable aux deux autres.



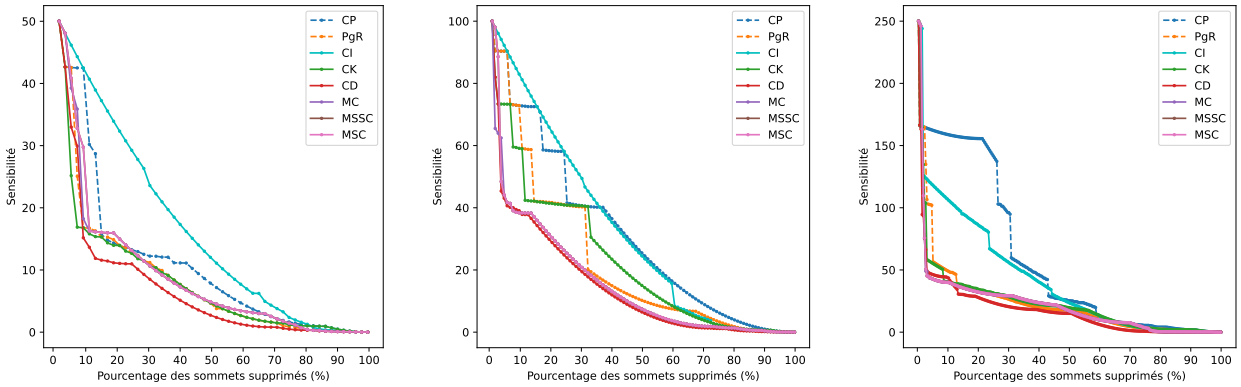
(a) Graphe taille 50 (b) Graphe taille 100 (c) Graphe taille 250

FIGURE 6.5 Sensibilité en fonction des pourcentages de sommets supprimés.

Ensuite, la figure **6.6** présente la sensibilité pour tous les autres indices de centralité sur les trois graphes de processus d'écriture en fonction du pourcentage de sommets supprimés. Globalement, les performances des indices de centralité d'intermédiarité (`CI`) et des indices de centralité de proximité (`CP`) sont les moins bonnes. Notre approche avec la méthode de modifications spatio-chronologiques `MSC` est quant à elle parmi les plus performantes. Pour le graphe de taille 50, elle a une allure similaire à la courbe obtenue avec la méthode de PageRank. Mais pour les deux autres, elle est semblable à la centralité de degré.

Efficacité : Cette comparaison a pour objectif de voir comment la communication entre les sommets se détériore dans le graphe pour les différentes méthodes lorsqu'on supprime les sommets un à un suivant l'ordre décroissant d'importance.

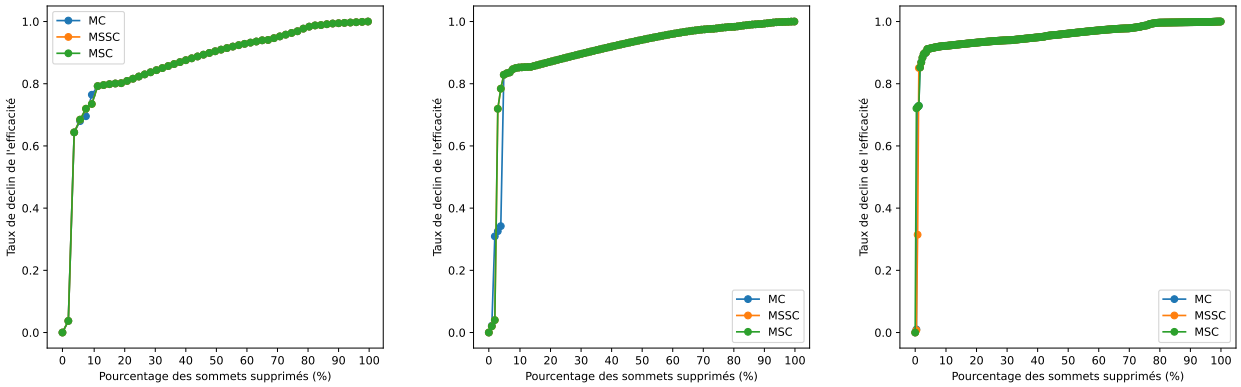
Les figures **6.7** et **6.8** donnent le taux de déclin de l'efficacité des trois graphes simulés en fonction des pourcentages de sommets supprimés. Dans la figure **6.7**, nous comparons les méthodes de modification entre elles. Parmi les trois méthodes, aucune ne se



(a) Graphe taille 50 (b) Graphe taille 100 (c) Graphe taille 250

FIGURE 6.6 Sensibilité des graphes en fonction des pourcentages de sommets supprimés.

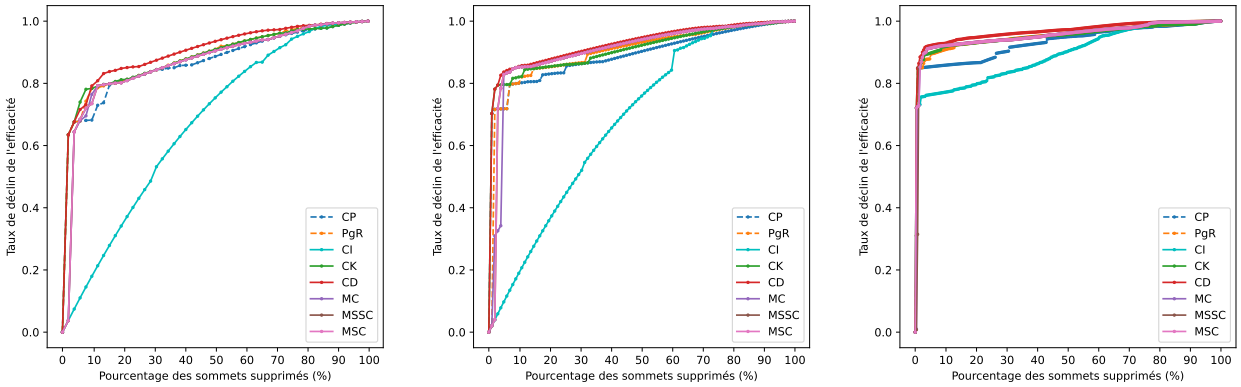
démarque considérablement. Ce qui signifie qu'elles ont une efficacité quasi similaire à déterminer les sommets importants pour la connectivité. Par contre, elles surpassent les autres méthodes telles que la centralité d'intermédiarité (`CI`), la centralité de proximité (`CP`), mais leurs performances sont quelque peu identiques avec les méthodes telles que la centralité de degré (`CD`), pageRank (`PgR`) et la centralité de Katz (`CK`).



(a) Graphe taille 50 (b) Graphe taille 100 (c) Graphe taille 250

FIGURE 6.7 Taux de déclin de l'efficacité en fonction des pourcentages de sommets supprimés.

Robustesse : Cette comparaison a pour objectif de voir comment la connectivité change, mais en utilisant plutôt la taille des composantes connexes obtenues lorsque l'on supprime les sommets un à un suivant leur importance. Les résultats expérimentaux de la Figure **6.9** reflètent les performances des différentes méthodes de modification sur la connectivité du graphe en utilisant le coefficient de connectivité maximal. On constate



(a) Graphe taille 50 (b) Graphe taille 100 (c) Graphe taille 250

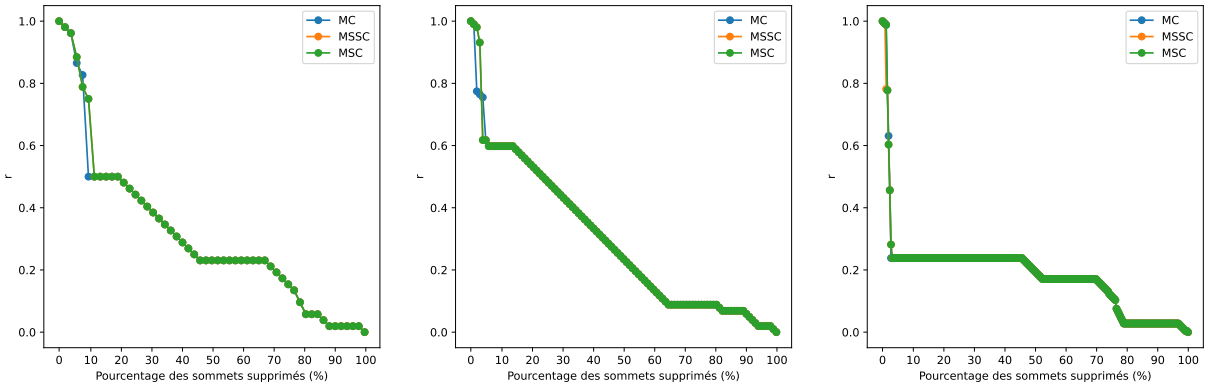
FIGURE 6.8 Taux de déclin de l'efficacité en fonction des pourcentages de sommets supprimés.

que, suivant le graphe et le pourcentage de sommets importants que l'on veut obtenir, l'utilisation des méthodes `MSSC` et `MSC` peut être avantageuse par rapport à la méthode `MC`. En effet, dans certaines situations, les méthodes `MSSC` et `MSC` modifient plus rapidement la taille de la plus grande composante du graphe. Par exemple, dans le graphe de taille 250, lorsque 2% des sommets les plus importants sont éliminés, les valeurs du coefficient de connectivité maximale (r) des méthodes `MC`, `MSSC` et `MSC` sont respectivement de 0.630, 0.603 et 0.603. Ce qui signifie que la connectivité du graphe devient moins bonne avec les méthodes `MSC` et `MSSC` que celle de `MC` lorsqu'on les utilise pour éliminer 2% des sommets les plus importants. Par contre, pour le graphe de taille 50, lorsque les 10% des sommets les plus importants sont éliminés, les valeurs du coefficient de connectivité maximal (`r`) des méthodes `MC`, `MSSC` et `MSC` sont respectivement de 0.5, 0.75 et 0.75. Ce qui veut dire dans ce cas que la connectivité du graphe devient moins bonne avec la méthode `MC` que celles des méthodes `MSC` et `MSSC`. Mais lorsqu'on supprime 20% des sommets importants : la connectivité devient de même qualité pour les trois méthodes.

Nous avons analysé la robustesse des trois méthodes `MC`, `MSSC` et `MSC` comme mesure d'importance sur la base de la connectivité. Le tableau **6.3** ci-dessous présente les résultats de l'évaluation de leur robustesse `R`.

TABLEAU 6.3 Robustesse des méthodes `MC`, `MSSC` et `MSC`

	MC	**MSSC**	**MSC**
50	0, 297	0, 297	0, 297
100	0, 286	0, 289	0, 289
250	0, 181	0, 180	0, 180



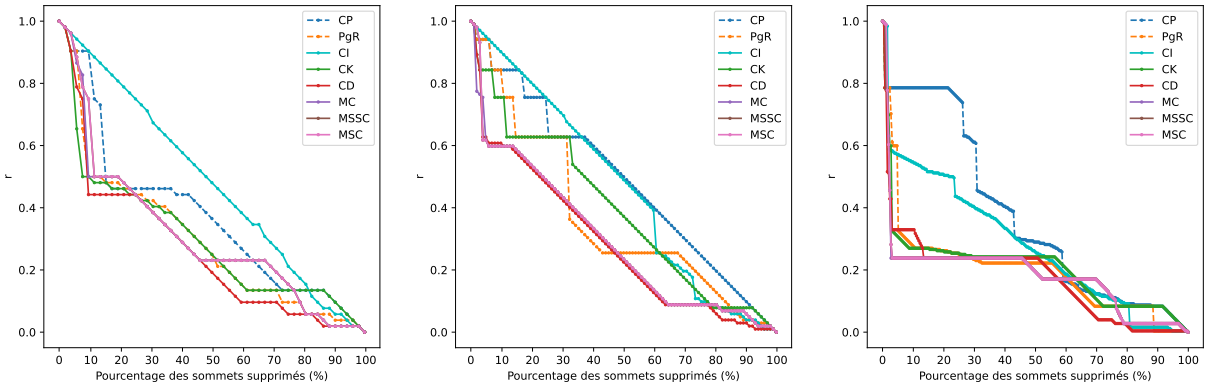
(a) Graphe taille 50 (b) Graphe taille 100 (c) Graphe taille 250

FIGURE 6.9 Coefficient de connectivité maximale des graphes en fonction des pourcentages de sommets supprimés.

Une faible robustesse traduit l'effondrement rapide du graphe et alors une meilleure performance de la méthode pour l'identification des sommets importants pour la connectivité. Le tableau montre que les trois méthodes ont des robustesses presque identiques, mais la méthode `MC` s'est démarquée sur le graphe de taille 50 en obtenant la plus faible valeur. Il résulte que les trois méthodes sont proches, tout dépendant du graphe choisi. Maintenant, comparons ces méthodes avec les autres méthodes. La figure **6.10** présente l'évolution du coefficient de connectivité maximal suivant le pourcentage des sommets supprimés. On remarque, en général, qu'avec la méthode de centralité de degré et la méthode `MSC` les graphes s'effondrent rapidement, ce qui traduit le fait que ces méthodes peuvent identifier les sommets les plus importants du graphe avec plus de précision que les autres méthodes. Par exemple, pour le graphe de taille 250, lorsqu'on supprime 4% des sommets les plus importants, `r` a pour valeurs $0,238$, $0,329$ et $0,325$ respectivement pour les méthodes `MSC`, `CD` et `CI`. Pour le graphe de taille 100, 6% des sommets importants supprimés correspondent à $r = 0,605$ pour la méthode `CD`, à $r = 0,843$ pour la méthode `CK`et à $r = 0,598$ pour la méthode `MSC`. Les méthodes `CI` et `CP`, quant à elles, sont celles qui identifient le moins bien les sommets importants dans les trois graphes de processus d'écriture.

Le tableau **6.4** des robustesses des méthodes confirme ces observations. En effet, on constate que la méthode `CD` et nos méthodes obtiennent les plus faibles robustesses pour les graphes de taille 250 et 100. Par contre, les méthodes `CP` et `CI` obtiennent les plus grandes robustesses.

En résumé, l'étude de la sensibilité, de l'efficacité et de la robustesse des méthodes des



(a) Graphe taille 5 (b) Graphe taille 100 (c) Graphe taille 250

FIGURE 6.10 Coefficient de connectivité maximale des graphes en fonction des pourcentages de sommets supprimés.

TABLEAU 6.4 Robustesse

	CD	**CK**	**CP**	**CI**	**PgR**	**MC**	**MSSC**	**MSC**
50	**0,254**	0, 290	0, 361	0, 473	0, 288	0, 297	0, 297	0, 297
100	**0,277**	0, 383	0, 479	0, 457	0, 361	**0,286**	0, 289	0, 289
250	**0,171**	0, 202	0, 370	0, 278	0, 196	0, 181	0, 180	**0,180**

indices de modifications sur les graphes simulés permet de constater qu'importe le modèle que l'on utilise, l'indice de modifications permet de trouver efficacement les sommets importants pour la connectivité du graphe de processus d'écriture. De plus, ces résultats expérimentaux montrent que les indices de modification, en général, et surtout l'indice de modification spatio-chronologique, peuvent être de bons candidats, après la centralité de degré, dans le classement des sommets des graphes de processus d'écriture, suivant leur importance pour la connectivité du graphe.

Nous avons aussi remarqué que nos méthodes ont un comportement un peu similaire à celui de la centralité de PageRank lors de l'analyse de l'efficacité. De plus, nous avons constaté que les méthodes d'indice de modification sont plus proches des indices de centralité locale, ce qui peut s'expliquer par le fait que, dans l'indice de modification, on ne prend en compte que les sommets qui sont proches, d'une certaine manière, du sommet considéré. Il en résulte alors que les indices de modification peuvent, dans une certaine mesure, être aussi utilisés comme indice de centralité locale du graphe de processus d'écriture.

6.3 Conclusion

Ce chapitre a permis de présenter les différentes manières d'utiliser les notions de communauté de modification et d'indice de modification pour analyser les processus d'écriture. Nous avons présenté au total quatre applications, deux pour chacune des deux notions. Pour la notion de communauté de modification, nous avons tout d'abord illustré comment utiliser les communautés de modifications trouvées pour mettre en exergue le lien entre les niveaux d'étude et la quantité de modifications. Cette observation révèle que, dans nos données, les élèves des niveaux supérieurs produisent et révisent davantage leurs écrits. Ensuite, nous avons présenté comment classifier les processus d'écriture en utilisant les séries chronologiques des valeurs de lisibilité obtenues durant les phases de modification. Ainsi, nous avons remarqué que la mesure de lisibilité basée sur les mots difficiles est celle qui permet le mieux de classer les processus d'écriture de notre corpus. Pour les indices de modification, nous avons présenté une nouvelle méthode de représentation synthétique d'un processus d'écriture et, ensuite, nous avons montré que les indices de modification peuvent être utilisés comme indice de centralité locale pour les graphes de processus d'écriture.

CHAPITRE 7 CONCLUSION

Les travaux réalisés dans cette thèse s'inscrivent globalement dans le cadre de l'extraction de connaissances à partir des graphes issus des processus d'écriture, dans le but de mieux comprendre la manière dont un rédacteur écrit. Dans cette optique, nous avons défini deux nouveaux concepts : les communautés de modifications et les indices de modifications. Nous présentons un résumé des résultats obtenus, les limites et des perspectives possibles pour nos travaux.

7.1 Synthèse des travaux

La principale contribution de cette thèse est d'avoir développé des outils qui permettent d'analyser les processus d'écriture à travers les graphes issus de ces processus.

Au chapitre 4, nous avons défini la notion de « communauté de modifications » ou « épisode de modifications », qui permet de diviser le processus d'écriture en épisodes, et nous avons également illustré comment représenter ces épisodes pour une compréhension visuelle rapide de ce que le rédacteur écrit à chaque étape du processus. La représentation choisie permet de visualiser l'étendue des changements effectués dans chaque épisode, leur position et leur composition (insertion ou suppression). On peut également observer le mouvement du curseur durant le processus d'écriture. Nous avons aussi présenté la fusion des communautés, ce qui permet d'avoir une vue plus globale du processus d'écriture. La fusion des communautés peut être réalisée lorsque les épisodes successifs sont de petite taille (ce qui peut indiquer que le rédacteur fait de petits ajustements ici et là dans le texte) ou que le curseur se déplace seulement d'un petit nombre de caractères (ce qui suggère que les modifications sont effectuées dans la même partie du texte). De plus, nous avons défini le diagramme de contribution, qui permet de synthétiser l'information contenue dans les communautés en vue d'une évaluation quantitative des contributions de chaque épisode de modification à la constitution du texte final. Nous avons aussi catégorisé les insertions et les suppressions à travers le diagramme de distribution, ce qui permet de connaître la tendance d'écriture et de révision d'un rédacteur. On peut ainsi savoir s'il a tendance à réécrire son texte plusieurs fois avant d'obtenir un résultat final qui lui convient, ou au contraire, si ce qu'il écrit a une forte probabilité de rester dans la version finale du texte.

Au chapitre 5, nous avons défini l'indice de modification, qui donne un aperçu du nombre de modifications effectuées en un endroit spécifique du texte. Pour ce faire, nous avons tout d'abord défini trois voisinages. Le voisinage entrant regroupe toutes les actions réalisées dans l'espace situé avant un caractère précis dans le texte. Le voisinage sortant, quant à lui, regroupe toutes les actions réalisées dans l'espace situé après un caractère précis dans le texte. Enfin, le voisinage, qui est la réunion des voisinages entrants et sortants, regroupe les sommets des insertions et suppressions effectuées dans l'espace autour du caractère considéré. Nous avons ensuite défini trois types de modifications : la modification chronologique, qui permet de connaître la succession des modifications dans l'ordre chronologique ; la modification spatiale avec seuil chronologique, qui permet en plus de connaître la position des modifications ; et enfin, la modification spatio-chronologique avec seuil chronologique, qui permet d'avoir une précision sur les distances entre les modifications effectuées. Afin de mieux évaluer l'indice de modification, nous avons construit deux nouveaux modèles de processus d'écriture à partir du *modèle sans perte*. Nous avons étudié les propriétés de ces modèles, ce qui nous a permis de constater qu'ils sont également sans perte et que l'indice de modification est égal à 1 ou différent de 1, selon que les modèles contiennent ou non un chemin hamiltonien. Enfin, nous avons présenté un algorithme pour le calcul de l'indice de modification.

Au chapitre 6, nous avons mis en avant quatre différentes façons possibles d'utiliser les notions que nous avons définies. Pour chaque notion, nous avons présenté deux applications. La notion de communauté de modifications peut être utilisée pour illustrer le lien entre le niveau d'étude et la quantité de modifications réalisées. Elle peut aussi permettre de créer la série chronologique des valeurs de lisibilité qui, à son tour, permet de classer les processus d'écriture. L'indice de modification quant à lui permet d'obtenir une nouvelle méthode de représentation synthétique d'un processus d'écriture. De plus, l'indice de modification peut être utilisé comme mesure de centralité locale pour les graphes de processus d'écriture.

En résumé, les communautés de modifications et les indices de modifications sont des outils qui apportent un plus dans l'étude des processus d'écriture. Ils peuvent être utilisés en association avec d'autres outils pour une analyse encore plus poussée.

7.2 Limitations de la solution proposée

Dans cette section, nous présentons les limites de nos recherches, notamment les limites de la notion de communauté de modifications et d'indice de modification.

La principale limite des communautés de modifications est qu'elle ne tient pas compte de certaines types d'actions qui sont présent dans le processus d'écriture, notamment :

- Le fait de copier une partie d'un texte et le coller à une autre place : cette action est couramment utilisée lorsque l'on est en train de relire un texte car elle permet d'ajuster le contenu du texte.
- Les pauses : lorsqu'on écrit on peut s'arrêter un moment pour réfléchir ou même lire le texte.

Les outils d'analyse des communautés de modifications sont essentiellement tournés vers la composition des communautés en termes de types d'événement effectué (insertion ou suppression) et ne traitent pas des éléments textuels tels que les mots et les phrases que les communautés de modifications peuvent contenir. En effet, dans nos outils, l'unité d'analyse est le caractère pour tant d'autres unités tel que la pause, la phrase, le jet textuel sont possibles.

Les données utilisées sont de trois types à savoir : données réelles, données simulées et données créées manuellement. Bien qu'elles ont été très utiles pour les exemples et les comparaisons, l'utilisation des données réelles de très grande taille serait souhaitable notamment pour la classification des processus d'écriture avec la méthode de lisibilité. Cela permettrait d'avoir des résultats encore plus pertinents.

7.3 Améliorations futures

Dans cette section nous présentons les extensions possibles et les travaux futurs.

Les premières extensions possibles sont d'améliorer les modèles pour qu'ils puissent tous combler les limites évoquées plus haut. Ainsi, il sera question par exemple de :

- Ne pas considérer le texte écrit comme une suite de caractères, mais utiliser plutôt d'autres éléments tels que les mots, les phrases, les paragraphes, voire les chapitres. De cette manière, le modèle s'adaptera mieux à certaines applications en linguistique où la phrase ou le paragraphe sont les unités de base.
- Ajouter d'autres types d'actions, comme par exemple les copier-coller et les pauses, lors de la construction des communautés afin de rendre le modèle plus robuste.
- Établir un modèle de classification des processus d'écriture basé sur les communautés et des mesures de lisibilité, reposant sur des données de taille considérable.

En plus de cela, il serait aussi possible d'examiner les autres pistes d'analyse de processus d'écriture que les communautés de modifications peuvent offrir, comme par exemple : la quantité d'informations qui se relaient d'un épisode de modification à l'autre.

Dans un tout autre ordre d'idées, il serait aussi possible d'analyser les *modèles* 1 et 2 pour trouver des éléments qui pourraient aider à caractériser encore plus les processus d'écriture. En effet, les communautés n'ont été déterminées qu'avec le *modèle sans perte* (graphe original), mais pas dans les *modèles* 1 et 2. Ainsi, il serait aussi intéressant de savoir comment des communautés dans ces graphes peuvent être définies.

RÉFÉRENCES

[1] D. Alamargot and J.-L. Lebrave, "The study of professional writing : A joint contribution from cognitive psychology and genetic criticism," *European Psychologist*, vol. 15, 01 2010.

[2] R. Alba, "A graph-theoretic definition of a sociometric clique†," *Journal of Mathematical Sociology*, vol. 3, pp. 113–126, 1973.

[3] L. K. Allen, M. E. Jacovina, M. Dascalu, R. Roscoe, K. Kent, A. Likens, and D. McNamara, "ENTERing the time series SPACE : Uncovering the writing process through keystroke analyses," 2016, pp. 22–29.

[4] J. M. Anthonisse, "The rush in a directed graph," *Journal of Computational Physics*, pp. 1–10, 1971. [Online]. Available : https://api.semanticscholar.org/CorpusID:118421505

[5] M. Atzmueller, S. Günnemann, and A. Zimmermann, "Mining communities and their descriptions on attributed graphs : a survey," *Data Mining and Knowledge Discovery*, vol. 35, pp. 661–687, Feb. 2021.

[6] I. M. Azpiazu and M. S. Pera, "Is cross-lingual readability assessment possible ?" *Journal of the Association for Information Science and Technology*, vol. 71, 2019.

[7] ——, "Multiattentive recurrent neural network architecture for multilingual readability assessment," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 421–436, 2019.

[8] V. Baaijen, D. Galbraith, and K. Glopper, "Keystroke analysis reflections on procedures and measures," *Written Communication*, vol. 29, pp. 246–277, 07 2012.

[9] N. Ballier, D. Galbraith, and K. Glopper, "Investigating keylogs as tim-stamped graphemics," 2019, pp. 353–365.

[10] A. Bavelas, "Communication patterns in task-oriented groups," *The journal of the acoustical society of America*, vol. 22, no. 6, pp. 725–730, 1950.

[11] H.-S. Bécotte-Boutin, "Analyse et visualisation du processus d'écriture à l'aide des graphes," Ph.D. dissertation, Polytechnique Montréal, juin 2019. [Online]. Available : https://publications.polymtl.ca/4047/

[12] L. L. Bekius, "'behind the computer screens' : the use of keystroke logging for genetic criticism applied to born-digital works of literature," Ph.D. dissertation, University of Amsterdam and University of Antwerp, 2023.

[13] J.-P. Benoit, "Revue critique des formules de lisibilité (60 ans de formules de lisibilité : qu'en reste-t-il ?)," *Pratiques*, vol. 52, no. 1, pp. 45–63, 1986.

[14] P. Berkhin, "A survey on pagerank computing," *Internet mathematics*, vol. 2, no. 1, pp. 73–120, 2005.

[15] V. W. Berninger, *Past, present, and future contributions of cognitive writing research to cognitive psychology*. Psychology Press, 2012.

[16] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *Journal of mathematical sociology*, vol. 2, no. 1, pp. 113–120, 1972.

[17] ——, "Power and centrality : A family of measures," *American journal of sociology*, vol. 92, no. 5, pp. 1170–1182, 1987.

[18] J. Bondy and U. Murty, "Traduit de l'anglais par f," *Havet Théorie des Graphes*, p. 202, 2008.

[19] J. A. Bondy, U. S. R. Murty *et al.*, *Graph theory with applications*. Macmillan London, 1976, vol. 290.

[20] S. P. Borgatti and M. G. Everett, "A graph-theoretic perspective on centrality," *Social Networks*, vol. 28, no. 4, pp. 466–484, 2006.

[21] J. R. Bormuth, "Development of readability analysis," *ERIC Clearinghouse*, 1969.

[22] C. Bothorel, J. D. Cruz, M. Magnani, and B. Micenkova, "Clustering attributed graphs : models, measures and methods," 2015.

[23] U. Brandes, *Network analysis : methodological foundations*. Springer Science & Business Media, 2005, vol. 3418.

[24] U. Brandes and D. Fleischer, "Centrality measures based on current flow," in *Annual symposium on theoretical aspects of computer science*. Springer, 2005, pp. 533–544.

[25] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107–117, 1998, proceedings of the Seventh International World Wide Web Conference.

[26] B. Bruce, A. Rubin, and K. Starr, "Why readability formulas fail," *IEEE Transactions on Professional Communication*, vol. PC-24, no. 1, pp. 50–52, 1981.

[27] G. Caporossi and C. Leblay, "Online writing data representation : A graph theory approach," in *Advances in Intelligent Data Analysis X*, J. a. Gama, E. Bradley, and J. Hollmén, Eds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, pp. 80–89.

[28] G. Caporossi, C. Leblay, and H. Usoof, "Genographix-log version 2.0 user guide," *Les Cahiers du GERAD ISSN*, vol. 711, p. 2440, 2020.

[29] M. Coleman and T. L. Liau, "A computer readability formula designed for machine scoring." *Journal of Applied Psychology*, vol. 60, no. 2, p. 283, 1975.

[30] D. Combe, C. Largeron, E. Egyed-Zsigmond, and M. Géry, "Combining relations and text in scientific network clustering," in *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 2012, pp. 1248–1253.

[31] E. Dale and J. S. Chall, "A formula for predicting readability," *Educational Research Bulletin*, vol. 27, no. 1, pp. 11–28, 1948. [Online]. Available : http://www.jstor.org/stable/1473169

[32] ——, "The concept of readability," *Elementary English*, vol. 26, no. 1, pp. 19–26, 1949. [Online]. Available : http://www.jstor.org/stable/41383594

[33] S. Dereich and P. Mörters, "Random networks with sublinear preferential attachment : The giant component," *The Annals of Probability*, vol. 41, no. 1, Jan. 2013. [Online]. Available : http://dx.doi.org/10.1214/11-AOP697

[34] T. Deutsch, M. Jasbi, and S. Shieber, "Linguistic features for readability assessment," in *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, J. Burstein, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, H. Yannakoudakis, and T. Zesch, Eds. Seattle, WA, USA → Online : Association for Computational Linguistics, jul 2020, pp. 1–17.

[35] C. Doquet, "Étude génétique de l'Écriture sur traitement de texte d'Éléves de cours moyen 2, année 1995-1996," Ph.D. dissertation, Université de Paris 3, 2003.

[36] C. Doquet-Lacoste, "Le temps d'écrire : stratégies d'écriture et chronologie des événements dans des processus d'écriture d'élèves de cm2," *Repères. Recherches en didactique du français langue maternelle*, vol. 11, no. 1, pp. 59–72, 1995.

[37] W. DuBay, "The principles of readability," *Impact Information*, 2004.

[38] J. A. Emig, "On teaching composition : Some hypotheses as definitions," *Research in the Teaching of English*, 1967. [Online]. Available : https://api.semanticscholar.org/CorpusID:141178254

[39] ——, "The composing processes of twelfth graders," *National Council of teachers of English research report*, 1971.

[40] I. E. Fang, "The "easy listening formula"," *Journal of Broadcasting & Electronic Media*, vol. 11, no. 1, pp. 63–68, 1966.

[41] R. Flesch, "A new readability yardstick." *Journal of applied psychology*, vol. 32, no. 3, p. 221, 1948.

[42] L. Flower and J. R. Hayes, "A cognitive process theory of writing," *College Composition and Communication*, vol. 32, no. 4, pp. 365–387, 1981.

[43] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, p. 399–404, 1956.

[44] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[45] J. Foucambert, "La recherche genèse du texte," *Repères. Recherches en didactique du français langue maternelle*, vol. 11, no. 1, pp. 47–57, 1995.

[46] J.-C. Fournier, *Théorie des graphes et applications : avec exercices et problèmes.* Lavoisier, 2011.

[47] T. François, "Les apports du traitement automatique du langage à la lisibilité du français langue étrangère," Ph.D. dissertation, Ph. D. thesis, Université Catholique de Louvain., 2011.

[48] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.

[49] L. C. Freeman, S. P. Borgatti, and D. R. White, "Centrality in valued graphs : A measure of betweenness based on network flow," *Social Networks*, vol. 13, pp. 141–154, 1991. [Online]. Available : https://api.semanticscholar.org/CorpusID:383472

[50] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[51] A. Grésillon, *Éléments de critique génétique. Lire les manuscrits modernes.* CNRS éditions, 2016.

[52] S. Günnemann, B. Boden, and T. Seidl, "Db-csc : A density-based approach for subspace clustering in graphs with feature vectors," in *ECML/PKDD*, 2011. [Online]. Available : https://api.semanticscholar.org/CorpusID:15228316

[53] S. Günnemann, I. Färber, B. Boden, and T. Seidl, "Subspace clustering meets dense subgraph mining : A synthesis of two paradigms," *2010 IEEE International Conference on Data Mining*, pp. 845–850, 2010. [Online]. Available : https://api.semanticscholar.org/CorpusID:6165228

[54] S. Günnemann, I. Färber, S. Raubach, and T. Seidl, "Spectral subspace clustering for graphs with feature vectors," *2013 IEEE 13th International Conference on Data Mining*, pp. 231–240, 2013. [Online]. Available : https://api.semanticscholar.org/CorpusID:1021448

[55] R. Gunning, "The technique of clear writing," 1952.

[56] P. Hage and F. Harary, "Eccentricity and centrality in networks," *Social networks*, vol. 17, no. 1, pp. 57–63, 1995.

[57] J. Hayes and L. Flower, *Identifying the organization of writing processes*, 01 1980, pp. 3–.

[58] J. R. Hayes, "A new framework for understanding cognition and affect in writing," in *The science of writing*. Routledge, 1996, pp. 1–27.

[59] C. A. Hill, D. L. Wallace, and C. Haas, "Revising on-line : Computer technologies and the revising process," *Computers and Composition*, vol. 9, no. 1, pp. 83–109, 1991.

[60] C. H. Hubbell, "An input-output approach to clique identification," 1965. [Online]. Available : https://api.semanticscholar.org/CorpusID:120923022

[61] J. Kalofolias, M. Boley, and J. Vreeken, "Discovering robustly connected subgraphs with simple descriptions," in *IEEE International Conference on Data Mining (ICDM)*, 2019.

[62] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, pp. 39–43, 1953. [Online]. Available : https://api.semanticscholar.org/CorpusID:121768822

[63] S. Kemper, "Measuring the inference load of a text." *Journal of educational psychology*, vol. 75, no. 3, p. 391, 1983.

[64] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[65] D. Klein, "M. randić,"resistance distance,"," *J. Math. Chem*, vol. 12, no. 1, pp. 81–95, 1993.

[66] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[67] P. Kollberg, "S-notation as a tool for analysing the episodic structure of revisions," 1997. [Online]. Available : https://api.semanticscholar.org/CorpusID:17942739

[68] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski, "Centrality indices," *Network analysis : methodological foundations*, pp. 16–61, 2005.

[69] T. K. Landauer, "Pearson's text complexity measure," *Pearson's White Papers.–2011*, 2011.

[70] C. Leblay and G. Caporossi, "La dynamique d'Écriture dans la description linguistique. nouveaux modes de visualisation de l'Écriture enregistree," 01 2016.

[71] R. Lempel and S. Moran, "The stochastic approach for link-structure analysis (salsa) and the tkc effect," *Computer Networks*, vol. 33, no. 1-6, pp. 387–401, 2000.

[72] B. A. Lively and S. L. Pressey, "A method for measuring the vocabulary burden of textbooks," *Educational administration and supervision*, vol. 9, no. 7, pp. 389–398, 1923.

[73] F. Lorrain and H. C. White, ""structural equivalence of individuals in social networks"," *The SAGE Encyclopedia of Research Design*, 2022. [Online]. Available : https://api.semanticscholar.org/CorpusID:16625374

[74] D. Lucf, "Connectivity and generalized cliques in sociometric group structure," *Psychometrika*, vol. 15, pp. 169–190, 1950. [Online]. Available : https://api.semanticscholar.org/CorpusID:40990156

[75] C. Mahlow, M. A. Ulasik, and D. Tuggener, "Extraction of transforming sequences and sentence histories from writing process data : a first step towards linguistic modeling of writing," *Reading and Writing*, vol. 37, no. 2, pp. 443–482, 2024.

[76] D. Manik, M. Rohden, H. Ronellenfitsch, X. Zhang, S. Hallerberg, D. Witthaut, and M. Timme, "Network susceptibilities : Theory and applications," *Phys. Rev. E*, vol. 95, p. 012319, Jan 2017. [Online]. Available : https://link.aps.org/doi/10.1103/PhysRevE.95.012319

[77] M. Martinc, S. Pollak, and M. Robnik-Šikonja, "Supervised and unsupervised neural approaches to text readability," *Computational Linguistics*, vol. 47, no. 1, pp. 141–179, 2021.

[78] A. Miletić, C. Benzitoun, G. Cislaru, and S. Herrera-Yanez, "Pro-text : An annotated corpus of keystroke logs," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, 2022, pp. 1732–1739.

[79] R. J. Mokken, "Cliques, clubs and clans," *Quality and Quantity*, vol. 13, pp. 161–173, 1979. [Online]. Available : https://api.semanticscholar.org/CorpusID:122414757

[80] D. M. Murray, "Internal revision : A process of discovery," *Research and Composing/-National Council of Teachers of English*, 1978.

[81] F. Nadeem and M. Ostendorf, "Estimating linguistic complexity for science texts," in *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, 2018, pp. 45–55.

[82] J. Neville, M. Adler, and D. D. Jensen, "Clustering relational data using attribute and link information," 2003. [Online]. Available : https://api.semanticscholar.org/CorpusID:163995

[83] M. E. J. Newman, "A measure of betweenness centrality based on random walks," 2003. [Online]. Available : https://arxiv.org/abs/cond-mat/0309045

[84] ——, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, p. 066133, Jun 2004. [Online]. Available : https://link.aps.org/doi/10.1103/PhysRevE.69.066133

[85] ——, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[86] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, Feb 2004. [Online]. Available : https://link.aps.org/doi/10.1103/PhysRevE.69.026113

[87] M. E. J. Newman, "Networks : an introduction," 2010.

[88] L. Page, "The pagerank citation ranking : Bringing order to the web," Technical Report, Tech. Rep., 1999.

[89] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[90] F. Parés, D. G. Gasulla, A. Vilalta, J. Moreno, E. Ayguadé, J. Labarta, U. Cortés, and T. Suzumura, "Fluid communities : A competitive, scalable and diverse community detection algorithm," in *Complex Networks & Their Applications VI : Proceedings of Complex Networks 2017 (The Sixth International Conference on Complex Networks and Their Applications)*. Springer, 2018, pp. 229–240.

[91] D. Perrin, "Progression analysis (pa) : investigating writing strategies at the workplace," *Journal of Pragmatics*, vol. 35, no. 6, pp. 907–921, 2003.

[92] S. Pool, F. Bonchi, and M. van Leeuwen, "Description-driven community detection," *ACM Trans. Intell. Syst. Technol.*, vol. 5, pp. 28 :1–28 :28, 2014. [Online]. Available : https://api.semanticscholar.org/CorpusID:15995433

[93] C. H. Proctor and C. P. Loomis, "Analysis of sociometric data," *Research methods in social relations*, vol. 2, pp. 561–585, 1951.

[94] C. Rianne, R. Jens, and M. van Zaanen, "Understanding the keystroke log : The effect of writing task on keystroke features," *Reading and Writing*, vol. 32, pp. 2353–2374, 2019.

[95] F. Richaudeau *et al.*, *La lisibilité*. FeniXX, 1969.

[96] D. G. Rohman and A. O. Wlecke, "Pre-writing, the construction and application of models for concept formation in writing." 1964.

[97] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[98] S. E. Schwarm and M. Ostendorf, "Reading level assessment using support vector machines and statistical language models," in *Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05)*, 2005, pp. 523–530.

[99] S. B. Seidman, “Network structure and minimum degree,” *Social Networks*, vol. 5, no. 3, pp. 269–287, 1983. [Online]. Available : https://www.sciencedirect.com/science/article/pii/037887338390028X

[100] J. Selzer, “Readability is a four-letter word,” *The Journal of Business Communication (1973)*, vol. 18, no. 4, pp. 23–34, 1981.

[101] L. Sharp, “Acts of writing : A compilation of six models that define the processes of writing,” *International Journal of Instruction*, vol. 9, pp. 77–90, 06 2016.

[102] K. M. Sheehan, M. Flor, and D. Napolitano, “A two-stage approach for generating unbiased estimates of text complexity,” in *Proceedings of the Workshop on Natural Language Processing for Improving Textual Accessibility*, 2013, pp. 49–58.

[103] K. M. Sheehan, I. Kostin, D. Napolitano, and M. Flor, “The textevaluator tool : Helping teachers and test developers select texts for use in instruction and assessment,” *The Elementary School Journal*, vol. 115, no. 2, pp. 184–209, 2014.

[104] E. A. Smith and R. Senter, *Automated readability index*. Aerospace Medical Research Laboratories, Aerospace Medical Division, Air . . ., 1967, vol. 66, no. 220.

[105] K. Stephenson and M. Zelen, “Rethinking centrality : Methods and examples,” *Social Networks*, vol. 11, no. 1, pp. 1–37, 1989.

[106] H. Sun, H. Du, J. Huang, Z. Sun, L. He, X. Jia, and Z. Zhao, “Detecting semantic-based communities in node-attributed graphs,” *Computational Intelligence*, vol. 34, 04 2018.

[107] ——, “Detecting semantic-based communities in node-attributed graphs,” *Computational Intelligence*, vol. 34, no. 4, pp. 1199–1222, 2018.

[108] Y. Sun, K. Chen, L. Sun, and C. Hu, “Attention-based deep learning model for text readability evaluation,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[109] F. Tanoï Namio, G. Caporossi, and A. Hertz, “Visualising the stages of a writing process,” in *An Introduction to data visualisation of the writing process*, C. Leblay, G. Caporossi, and H. Usoof, Eds. Brill, soumis pour publication.

[110] L. Timbal-Duclaux, “Textes "inlisable" et lisible,” *Communication & Langages*, vol. 66, no. 1, pp. 13–31, 1985.

[111] A. Todirascu, T. François, D. Bernhard, N. Gala, and A.-L. Ligozat, “Are cohesive features relevant for text readability evaluation ?” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics : Technical Papers*, 2016, pp. 987–997.

[112] M. A. Ulasik, C. Mahlow, and M. Piotrowski, “Sentence-centric modeling of the writing process,” *Journal of Writing Research*, vol. 16, no. 3, pp. 463–498, 2025.

[113] T. W. Valente and R. K. Foreman, "Integration and radiality : Measuring the extent of an individual's connectedness and reachability in a network," *Social networks*, vol. 20, no. 1, pp. 89–105, 1998.

[114] T. A. Van Dijk, "Text and context : Explorations in the semantics and pragmatics of discourse," 1977.

[115] I. Vragović, E. Louis, and A. Díaz-Guilera, "Efficiency of informational transfer in regular and complex networks," *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, vol. 71, no. 3, p. 036122, 2005.

[116] S. White and P. Smyth, "Algorithms for estimating relative importance in networks," in *Knowledge Discovery and Data Mining*, 2003. [Online]. Available : https://api.semanticscholar.org/CorpusID:406311

[117] H. Wiener, "Structural determination of paraffin boiling points," *Journal of the American chemical society*, vol. 69, no. 1, pp. 17–20, 1947.

[118] M. Wininger, "Measuring the evolution of a revised document," *Journal of Writing Research*, vol. 6, no. 1, pp. 1–28, 2014.

[119] M. Xia, E. Kochmar, and T. Briscoe, "Text readability assessment for second language learners," *arXiv preprint arXiv :1906.07580*, 2019.

[120] H. Yang and S. An, "Critical nodes identification in complex networks," *Symmetry*, vol. 12, no. 1, p. 123, 2020.