الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د .مولاي الطاهر
كلية التكنولوجيا
قسم: الإعلام الآلي

Mémoire de Master

Spécialité : Modélisation Informatique des Connaissances et du Raisonnement

Appariement inexact de graphes
L'algorithme FastPFP

Présenté par :
Taibi Yacine

Dirigé par :
MR. Rahmani Mohamed

Année universitaire 2022-2023

Remerciements

Je voudrais exprimer remercier sincère aux professeurs du département d'informatique de la faculté des sciences et de technologie de l'Université Moulay Tahar de Saida.

Je tiens à exprimer mes plus sincères remerciements à Monsieur **Mohamed Rahmani**, qui a été mon superviseur pour ce projet, pour son suivi et pour ses remarques et suggestions.Je le remercie pour son engagement à m'enseigner et à me soutenir à tout moment et pour ses efforts considérables et continus dans l'amélioration de mes compétences et de mes connaissances. Je lui en suis très reconnaissant. Qu'Allah le récompense avec la meilleure récompense. Je lui suis également très reconnaissant pour les précieux conseils et orientations qu'il m'a prodigués lors de la préparation de cette mémoire.

Dédicaces

Je dédie ce travail à :

Surtout à mes parents qui m'ont beaucoup encouragé

Mes frères

mes soeurs

et

mes amis

Résumé

L'appariement de graphes fait référence au processus de recherche d'une correspondance ou d'une mappage entre deux nœuds de graphes. L'objectif de notre travail est de présenter une introduction et une vue d'ensemble de l'appariement inexact des graphes.Il est utilisé dans des domaines tels que la reconnaissance de formes, le traitement d'images, l'analyse de données et la biométrie. Le problème d'optimisation quadratique d'affectation vise à optimiser l'allocation de n objets sur n emplacements, en minimisant les flux et les distances. FastPFP est un nouvel algorithme d'apparition de graphes inexacts, basé sur une nouvelle méthode virgule fixe. Il a montré des performances améliorées par rapport aux algorithmes précédents, en utilisant une projection stochastique double et une approche fixe virgule.

Mots-clés : appariement de graphes, appariement inexacte de graphes, affectation quadratique,l'algorithme FastPFP.

Abstract

Graph matching refers to the process of finding a match or mapping between two graph nodes. The objective of our work is to present an introduction and overview of inexact graph matching. It is used in fields such as pattern recognition, image processing, data analysis and biometrics. The quadratic assignment optimization problem aims to optimize the allocation of n objects to n locations, minimizing flows and distances. FastPFP is a new algorithm for appearing inexact graphs, based on a new fixed-point method. It showed improved performance over previous algorithms, using a double stochastic projection and a fixed-point approach.

Keywords : graph matching, inexact graph matching, quadratic assignment, the FastPFP algorithm.

Table des matières

Liste des figure

Introduction générale

1 Introduction et problématique :

La correspondance inexacte des graphes fait référence au processus de recherche de similitudes ou de modèles entre deux graphes ou plus lorsqu'une correspondance exacte n'est pas requise. Dans plusieurs disciplines, dont l'informatique, la science des données et les mathématiques, les graphes sont un outil crucial. Ils sont utilisés pour représenter des structures de données, les relations entre les entités. Il peut être difficile de comparer et de faire correspondre des graphes, surtout lorsque les graphes ne sont pas des correspondances exactes. Dans de nombreuses applications du monde réel, l'appariement exact de graphes est trop contraignant et ne produit pas de résultats significatifs. L'appariement inexact de graphes permet des inadéquations et des correspondances imparfaites entre les graphes.

En raison de l'espace de recherche plus compliqué, le problème de l'appariement inexact des graphes est plus difficile que le problème de l'appariement exact des graphes. Sur la base de méthodes telles que la recherche heuristique, la réduction de dimensionnalité, les noyaux de graphes, l'inférence probabiliste, l'apprentissage automatique, etc., de nombreux algorithmes ont été imaginés. Cependant, il reste un besoin de travail sur la mise à l'échelle et les solutions d'appariement de graphes inexacts généralisables.

L'appariement approximatif de graphes recherche des correspondances entre des graphes imparfaits qui tiennent compte de l'incertitude et des inadéquations. Il s'agit d'un problème important mais difficile, et des algorithmes créatifs sont encore nécessaires pour résoudre ses difficultés.

2 L’objectif de l’étude :

Dans les cas où une correspondance exacte n'est pas requise, l'objectif de la correspondance approximative des graphes est de découvrir des modèles ou des similitudes entre deux graphes ou plus. Cela peut impliquer de comparer des graphes de différentes tailles, formes et topologies, et d'identifier des correspondances entre des nœuds et des arêtes qui ne sont pas nécessairement identiques. L'objectif est d'extraire des informations utiles des graphes et des modèles de points de similitude ou de différence qui sont pertinents pour une application ou un domaine particulier.

3 Organisation du document :

Cette introduction générale, dans laquelle nous avons discuté le problème de l’appariement inexact des graphes de ses défis et des solutions potentielles, sert d'introduction à ce travail.

Ensuite, au chapitre 1, nous avons énuméré les concepts fondamentaux de la théorie des graphes qui sont requis pour le reste de l'essai. Dans ce chapitre, nous avons également discuté de certaines définitions liées aux graphes. Et Etat de l'art sur

les méthodes d'appariement de graphe est présenté au chapitre 2. Dans le chapitre3 Nous présentons en détail l'appariement inexact des graphes.

Enfin, au chapitre 4 : Nous présentons les résultats de nos recherches que nous avons pu obtenir.

Chapitre 1 :

Concepts de base dans les graphes et leur appariement

1 Introduction

L'étude des graphes, qui sont des constructions mathématiques utilisées pour représenter des interactions par paires entre des choses, fait l'objet de la discipline mathématique connue sous le nom de théorie des graphes. Un graphe est composé d'un certain nombre de nœuds, également appelés sommets, et d'un certain nombre d'arêtes de connexion. Une grande variété de systèmes et d'événements, y compris les réseaux sociaux, les systèmes de transport, les composés chimiques et les algorithmes informatiques, peuvent être modélisés à l'aide des graphes.

L'appariement de graphes, qui nécessite de déterminer une correspondance entre les nœuds de deux graphes différents, est l'un des problèmes les plus importants de la théorie des graphes. De nombreux domaines, tels que la bioinformatique, la vision par ordinateur et la reconnaissance de formes, utilisent la correspondance de graphes.

Le problème de l'appariement de graphes peut être résolu à l'aide d'un large éventail d'algorithmes et de méthodes. L'algorithme hongrois, la méthode d'appariement de graphes bipartis et l'algorithme d'appariement de poids maximum sont quelques-uns des plus largement utilisés.

Comprendre les concepts et les techniques de base peuvent être utiles dans de nombreux domaines différents. Nous Dans ce chapitre, nous introduisons quelques notions de la théorie des graphes en définissant quelques concepts et termes propres à cette théorie.

1.1 Théorie de graphes

Un domaine des mathématiques appelé "théorie des graphes" concerne l'étude des graphes. La description mathématique d'une collection d'éléments, ou de nœuds, reliés par une collection d'arêtes, ou de liens, s'appelle un graphe. La modélisation et l'analyse de systèmes complexes, y compris les composés chimiques, les réseaux sociaux, les réseaux de transport et les réseaux informatiques, sont effectuées à l'aide de la théorie des graphes.

Certains termes clés de la théorie des graphes incluent la connectivité, qui décrit le degré de connexion d'un graphe, le degré, qui décrit le nombre d'arêtes liées à un sommet, et le chemin, qui est une série d'arêtes qui relient deux sommets. Les graphes Planaires, qui peuvent être représentés sur un plan sans qu'aucune arête ne se chevauche, sont l'un des autres concepts cruciaux de la théorie des graphes.

Comme mentionné au [23], les travaux de Leonardo Euler sur le problème du pont de Königsberg marquent le début de la théorie des graphes. Euler a utilisé un graphe pour modéliser ce problème : Chaque parcelle de terrain arrêté par la rivière a un sommet, et chaque pont qui les relie a une arête.

1.2 Les définitions :

Les définitions et les différents types de graphes ci-dessous sont tirés de [24, 25, 26 et 27]

1.2.1 Graphe :

Les ensembles finis V = (v1, v2, ..., vn) et E = (e1, e2 , ..., em) construisent un graphe fini G = (V, E), dont les constituants sont appelés respectivement sommets et arêtes.

Une paire de sommets non ordonnés appelés extrémités de e sert de définition d'une arête **e** de la collection **E**. Les sommets **a** et b sont dits adjacents ou incidents à l'arête e si elle les relie, ou si l'arête e est incident aux sommets **a** et **b**. La figure 1.1 représente un graphe non orienté à 8 sommets, notés par les lettres **a** à **h**, et 10 arêtes. Ce graphe est défini comme suit :

G= (V, E)

V= {a, b, c, d, e, f, g, h}

E= {(a, d), (b, c), (b, d), (d, e), (e, c), (e, h), (h, d), (f, g), (d, g), (g, h)}

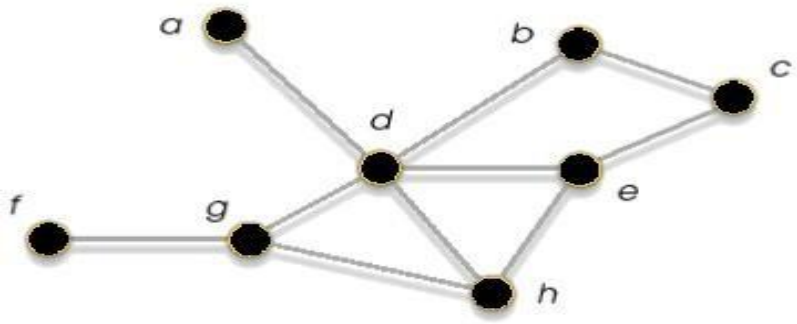

Figure 1.1 : *Un graphe non orienté*

La paire (x, y) diffère de la paire (y, x) en raison de leurs orientations différentes dans **un graphe orienté**, c'est-à-dire un graphe dont les arêtes sont orientées. Une flèche est utilisée pour symboliser **les arcs** d'un graphe orienté. Pour les distinguer des arêtes qui ne sont pas orientées, on les appelle des arcs.

Figure 1.2 : *Un graphe orienté*

1.2.2 Ordre, degré et distance d'un graphe :

Comme ce graphe a n sommets, on appelle son ordre |X|. La quantité d'arêtes qui coupent le degré d'un sommet, x, dans un graphe. Il est noté d(x). Le degré de x pour un graphe simple est également lié au nombre de sommets entourant

immédiatement x. Le nombre d'arêtes dans le chemin le plus court reliant deux sommets u et v représente la distance entre eux [23].

1.2.3 Boucle, chaîne, chemin, cycle et circuit :

Une boucle est une arête qui a des extrémités identiques. **Une chaîne simple** ou élémentaire est une chaîne dont la longueur maximale est égale au nombre d'arcs d'un graphe et qui ne peut passer qu'une seule fois au plus par arc. Une série d'arcs pointant tous dans la même direction s'appelle une route. Le chemin simple (f, g, d, b) en rouge et le chemin (f, g, d, h, e, d, b), qui n'est pas simple puisque le sommet d est visité deux fois, sont représentés sur la figure 1.3.

Un ensemble d'arcs commençant et se terminant au même sommet est appelé **un cycle** ou **circuit**. Gardez également à l'esprit qu'une chaîne simple résultera toujours d'un cycle simple ; sur la figure 1.3, le chemin (d, h, e, d) en vert désigne un cycle.

Observation : Dans le contexte des graphes orientés, les mots "chemin" et "circuit" sont appropriés. Nous discutons de la chaîne et du cycle en relation avec les graphes non orientés. Dans les deux cas, la structure (graphe orienté ou non) sur laquelle ils sont définis diffère, mais la définition formelle reste la même.

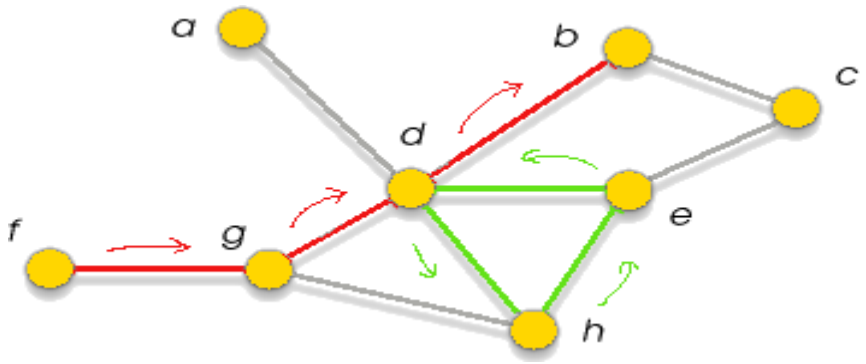

Figure 1.3 : *Exemple de chemin et cycle*

1.3 Types de graphes

Il existe différents types de graphes, nous énumérons ci-dessous quelques-uns des courants et des significatifs. Il convient de noter que le type de graphe à utiliser dépend du problème traité :

1.3.1 Multi-graphe

Un multigraphe est un type particulier de graphe dans lequel plusieurs arêtes peuvent joindre deux nœuds distincts. Cela implique qu'un multigraphe peut inclure de nombreuses arêtes, dont chacune peut avoir des propriétés uniques, reliant deux nœuds.

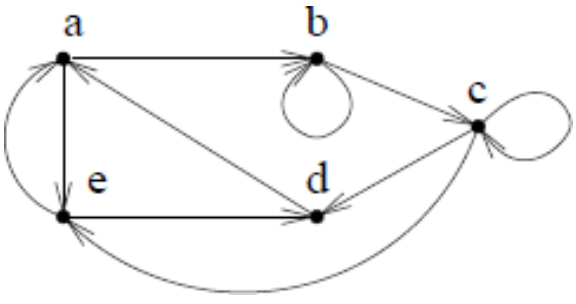

Figure 1.4 : Exemple d'un multi-graphe

1.3.2 Graphe simple

Chaque nœud et arête d'un graphe de base a une connectivité maximale de un. Pour le dire autrement, il n'y a pas de boucles ou d'arêtes qui relient un nœud à lui-même, et il n'y a pas d'arêtes multiples joignant deux nœuds.

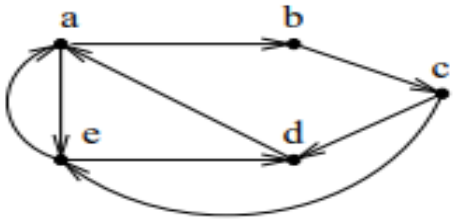

Figure 1.5 : *Un exemple de graphe simple*

1.3.3 Graphe connexe et non connexe :

Un graphe **connexe** est un graphe dans lequel chaque paire de nœuds peut être atteinte par un chemin. Autrement dit, en traçant les arêtes du graphe, il est possible de naviguer entre deux nœuds quelconques.

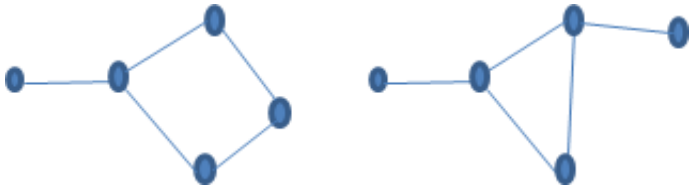

Figure 1.6 : Exemple de Graphe ayant deux composants connexe et non connexe

1.3.4 Graphe complet

Un graphe complet est un graphe où chaque sommet est adjacent à tous les autres sommets. Cela signifie que tous les nœuds sont connectés les uns aux autres.

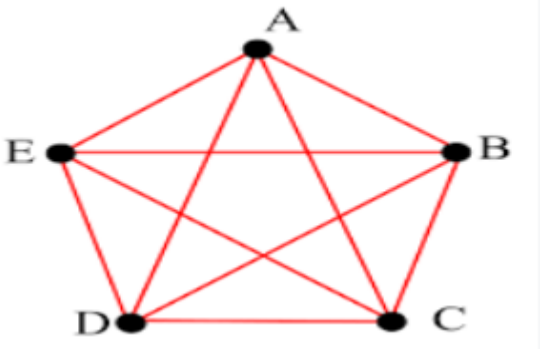

Figure 1.7 : *Un graphe complet*

1.3.5 Graphe planaire :

Un graphe planaire est un graphe qui a la propriété unique de pouvoir être représenté dans un plan sans arête, courbe ou lignes rectilignes sécantes (voir l'exemple de la figure 1.8 (a)).

1.3.6 Graphe biparti :

Comme le montre la Figure 1.8(b), un graphe est bipartite si ses sommets peuvent être divisés en deux ensembles, **X** et **Y**, et si toutes les arêtes du graphe relient un **X** à un **Y** sommet.

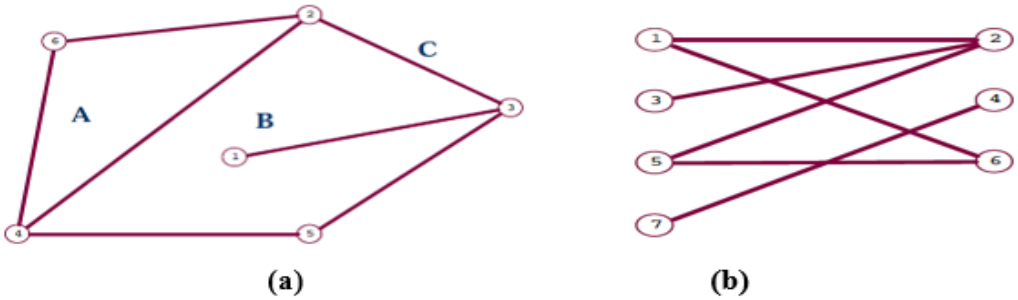

Figure 1.8 : *Exemple de graphe planaire (a) et de graphe biparti (b)*Graphe étiqueté :

Un graphe étiqueté est un graphe dans lequel chaque sommet et arête a une ou plusieurs étiquettes spécifiques qui lui sont attachées. Il peut s'agir de tout ce qui vous permet de les distinguer les uns des autres, y compris des lettres, des chiffres et des symboles.

1.3.7 Sous-graphe :

Un graphe a un sous-graphe. Par conséquent, tous les composants du graphe principal ne doivent pas nécessairement être présents dans un sous-graphe. Le terme « sous-graphe généré » fait référence à un sous-graphe qui est parfait, ou qui représente avec précision et totalement une certaine partie du graphe.

Dans un autre sens, un sous-graphe de **G** est un graphe **H= (V ; A(V))** où **V** est un sous-ensemble de **X** et **A(V)** sont les arêtes que **A** a induites sur **V** , c'est-à-dire les arêtes dont les deux extrémités sont les sommets de **V**.

1.3.8 Graphe partiel :

Considérons le graphe **G= (E, V).** Si **V1** est contenu dans **V,** le graphe **G1= (E1, V1)** est un graphe partiel de **G**. Autrement dit, on obtient **G1** en retirant une ou plusieurs arêtes du graphe **G**.

Considérons le graphe **G= (E, V)** de la figure 1.9, où **e**= {1, 2, 3, 4, 5, 6} et **V**= {(1,3), (1,4), (1, 5), (1,6), (2,3), (3,5), (5,6)}. Les sommets **E1**= {1, 2, 3, 4, 5, 6} et les arêtes **V1**= {(1,3), (1,4), (1,6), (2,3), (5, 6)} être né le graphe partiel **G1= (E1, V1)** de **G**. Les sommets **E2**= {1, 2, 4, 6} et les arêtes **V2**= {(1,3), (1,4), (1, 6)} de **G** fournissent le sous-graphe **G2= (E2, V2)** de **G**.

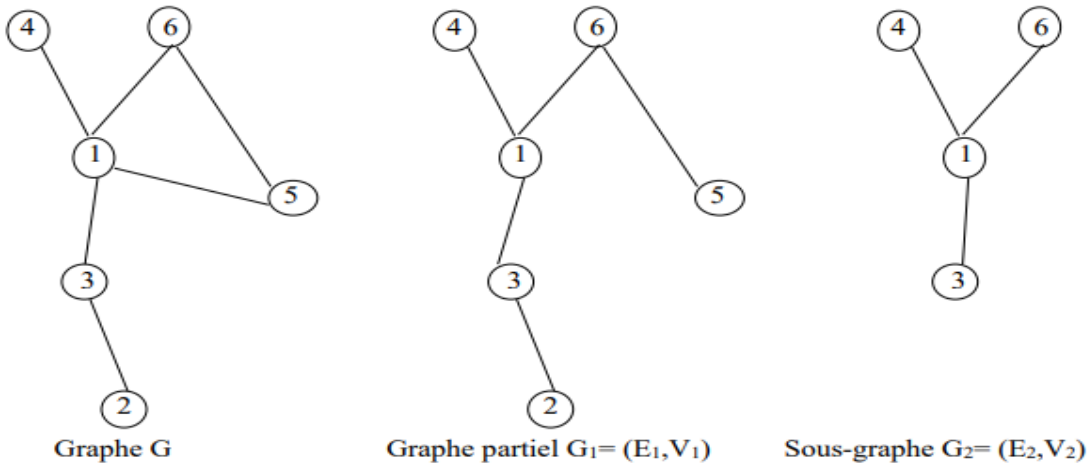

Figure 1.9 : Exemple de graphe partiel et sous graphe

1.3.9 Arbre et de foret :

Tout graphe connexe sans cycle est appelé **arbre**. **Une forêt** est un graphe qui n'a pas de cycles et qui n'est pas connexe.

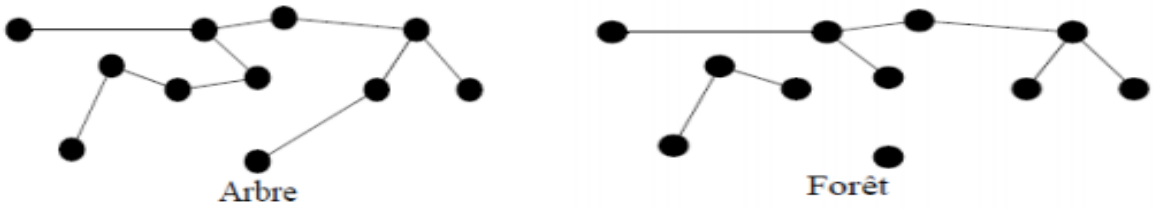

Figure 1.10 *:* Exemple d'arbre et de forêt

1.3.10 Parcours de graphes :

Navigation dans les graphes Il est courant que les problèmes de graphes appellent à traverser les sommets, les arcs et les arêtes du graphe. Les deux principales méthodes d'exploration sont les suivantes :

> **Le parcours en largeur** implique une enquête niveau par niveau des sommets du graphe en commençant à un certain sommet.

> **Le parcours en profondeur** implique de partir d'un certain sommet, de suivre le chemin aussi loin qu'il peut aller, puis de faire demi-tour pour continuer tous les chemins précédemment ignorés.

2 Représentation de graphe

L'ensemble des sommets et les voisins de chaque sommet suffisent pour représenter un graphe (sommets qui lui sont directement reliés par une arête). Selon le nombre d'arêtes, les opérations à effectuer, il existe plusieurs façons représenter des graphes. Tout graphe peut être représenté par une matrice d'incidence et d'adjacence ou par une liste d'adjacence ou par Listes d'arêtes.

2.1 Matrices d'adjacence

Une matrice d'adjacence est une matrice carrée qui représente un graphe. En théorie des graphes, un graphe est un ensemble de sommets et d'arêtes qui les relient. Une matrice d'adjacence est une matrice où les lignes et les colonnes représentent les sommets. Dans un graphe non pondéré, l'entrée (i, j) de la matrice de contiguïté est **1** s'il existe une arête entre le sommet **i** et le sommet **j**, et **0** sinon. Dans un graphe pondéré, l'entrée (i, j) représente le poids de l'arête entre le sommet **i** et le sommet **j.**

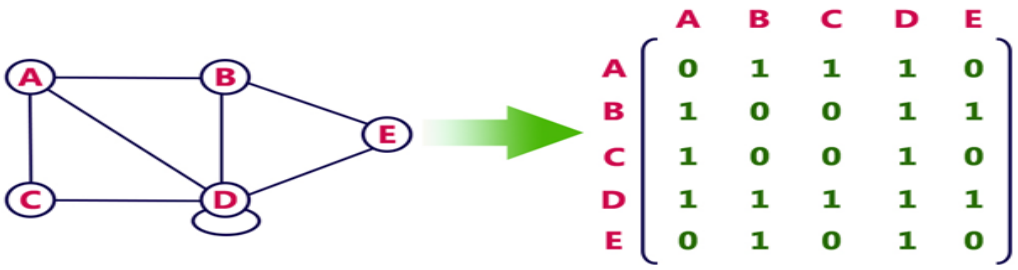

Figure 1.11 : Exemple de représentation d'un graphe par sa **matrice d'adjacence**

2.2 Matrices d'incidence

Une matrice de taille peut être utilisée pour représenter un graphe dans une matrice d'incidence : Nombre total de sommets par nombre total d'arêtes. Cela signifie qu'une matrice de classe 4X6 peut être utilisée pour représenter un graphe à 4 sommets et 6 arêtes. Les lignes de cette matrice représentent les sommets, tandis que les colonnes représentent les arêtes [28].

Soit **0**, soit **1**, soit **-1** remplit cette matrice. Où :

> Les arêtes de ligne qui ne sont pas connectées aux sommets de colonne sont représentées par le nombre **0**.

> L'arête de la ligne qui se connecte en tant que arête sortant au sommet de la colonne est représenté par le nombre 1.

> Les arêtes de ligne jointes aux sommets de colonne en tant qu'arêtes entrantes sont représentées par le nombre **-1**.

Figure 1.12 : Exemple de représentation d'un graphe par sa **matrice d'incidence**

2.3 Liste d'adjacence

Une représentation liée est appelée une liste d'adjacence. Avec ce modèle, nous gardons une trace des voisins de chaque sommet pour chaque nœud du graphe. Par conséquent, chaque sommet du graphe possède une liste des sommets qui lui sont immédiatement proches. Pour chaque sommet v, l'élément de tableau qui l'accompagne pointe vers une liste liée de manière simple des voisins de v dans un tableau de sommets qui est indexé par le numéro de sommet [28].

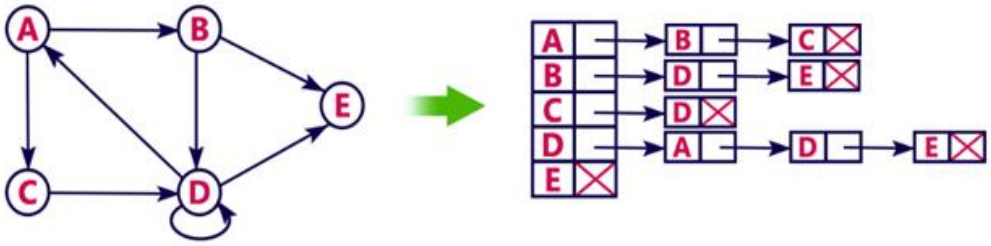

Figure 1.12 : *Exemple de représentation graphe d'une* ***liste d'adjacence***

3 Principes de base de la correspondance de graphe

Trouver des sommets correspondants par paires est l'objectif d'une recherche de correspondance de graphe. La correspondance dans un graphe G = (V, E) est une collection d'arêtes indépendantes **Ei**. L'absence de sommets incidents partagés entre les arêtes indique leur indépendance. Si tous les sommets de V0 sont couverts par **Ei**, ou si une arête de **Ei** relie chaque sommet de **Vi** à tous les autres sommets, alors V0 et V correspondent [4].

L'appariement de graphes fait référence au processus de recherche d'une correspondance ou d'un mappage entre les nœuds de deux graphes. Le but de l'appariement de graphes est d'identifier les nœuds similaires ou identiques dans différents graphes et d'établir une correspondance entre eux. Il existe différentes méthodes de correspondance de graphes, y compris des méthodes exactes et inexactes. Les méthodes exactes impliquent de résoudre le problème d'appariement de graphes en tant que problème d'optimisation, tandis que les méthodes inexactes utilisent des algorithmes heuristiques pour trouver une solution approximative au problème.

3.1 complexité de correspondance de graphe

L'appariement de graphes est considéré comme l'un des problèmes les plus difficiles. C'est sa nature combinatoire qui est à l'origine de cette complexité. Les problèmes d'appariement précis de graphes ont jusqu'à présent été classés comme P ou NP-complets. Lorsque les deux graphes à apparier sont d'un certain type ou ont un certain ensemble de restrictions, divers articles de la littérature tentent de démontrer qu'il est NP-complet. D'autre part, ils ont montré que la complexité du problème d'isomorphisme est polynomiale pour des types particuliers de graphes, comme l'isomorphisme des graphes planaires [29]. Pourtant, certains types de

graphes spécifiques peuvent avoir une complexité moins importante. Par exemple, considérons la situation unique dans laquelle le petit graphe est un arbre et le grand graphe est une forêt. La difficulté de l'appariement de graphes inexact s'est finalement révélée NP-complète [30].

3.2 Concepts de similarité de graphes

Il n'y a pas de formule pour déterminer à quel point deux graphes sont similaires. Souvent, cela dépend du type d'application. Les mesures les plus fréquemment utilisées pour comparer les graphes sont répertoriées ci-dessous.

3.2.1 La distance d'édition de graphes

La distance d'édition de graphe introduite par Bunke [31] recherche l'ensemble des opérations d'édition (insertion, suppression ou substitution de nœuds et d'arêtes) nécessaires pour convertir un graphe en un autre graphe au moindre coût possible. Les seules opérations utilisées sont la suppression et l'insertion si les graphées ne sont pas étiquetés. La fonction de coût de chaque opération varie en fonction du degré de distorsion introduit par la transformation.

La distance d'édition entre G1 et G2 est celle qui spécifie l'ensemble d'opérations le moins coûteux pour transformer G1 en un graphe isomorphe à G2.

Un exemple de l'ensemble des opérations d'édition qui peuvent être utilisées pour changer le graphe G1 en G2 est montré dans la Figure 1.14.

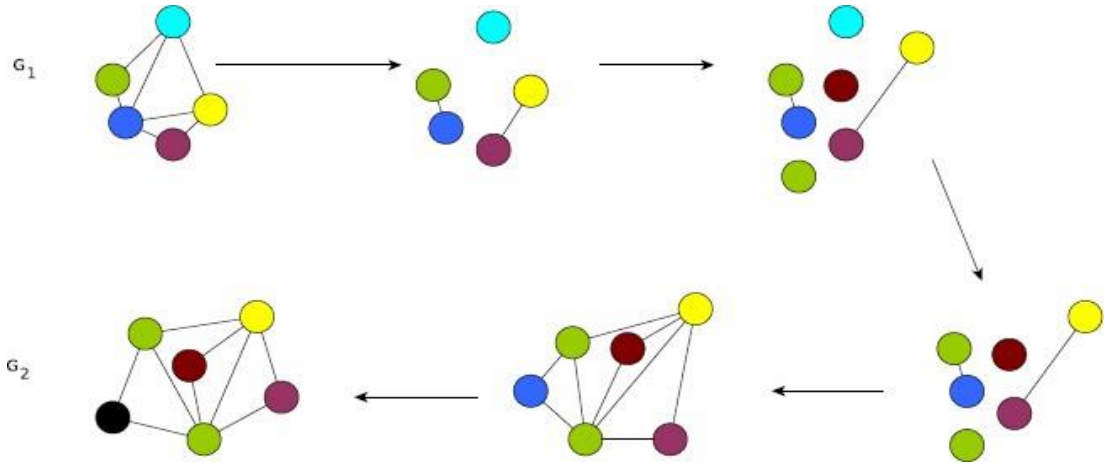

Figure 1.14 : Exemple d'une série d'étapes d'édition pour changer G1 en G2

3.2.2 Distance basée sur le concept maximum de sous-graphe (SCM)

Cette solution basée sur SCM surmonte le principal inconvénient des séquences d'édition basées sur la distance en évitant l'utilisation de fonctions de coût. Bunke et Shearer [32] ont démontré qu'une telle mesure est une métrique et donne une distance avec des valeurs dans la gamme [0,1].

La similarité basée sur le SCM est définie comme suit pour deux graphes donnés, G1 et G2 :

$$\mathrm{SimSCM(G1,G2)} = \frac{|SCM(G1,G2)|}{|max(G1,G2)|}$$

Où */max (*G_1, G_2*) /= max (/*G_1*/, /*G_2*/)* et */SCM (*G_1, G_2*) /* est le nombre d'arête dans *SCM (*G_1, G_2*)* [@5]. Un exemple d'un sous-graphe de maximum commun teinté de rouge entre deux graphes G1 et G2 est montré dans la Figure 1.15.

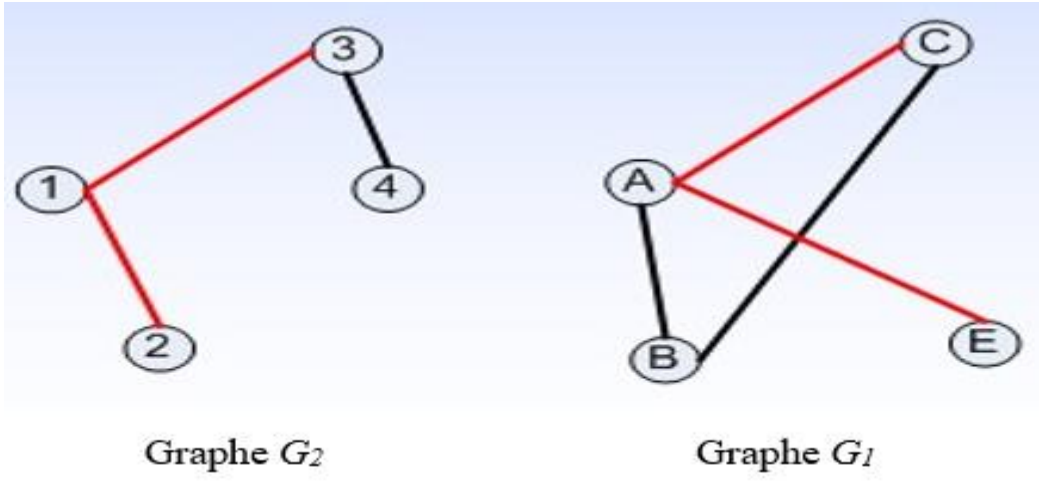

Figure 1.15 : Exemple de sous-graphe commun maximum entre deux graphes.

3.2.3 Autres distances

on peut citer la distance de Wallis et tout [33] ,Parmi les autres distances qui existent pour calculer la similarité entre graphes, qui est une mesure basée sur l'union des graphes (UG). Cette distance est utilisée pour modéliser la taille du problème. Il est défini comme suit :

$$\mathrm{SimUG}(\mathrm{G1},\mathrm{G2}) = \frac{|SCM(G1,G2)|}{|G1| + |G2| - |SCM(G1,G2)|}$$

Afin de mesurer la distance entre deux graphes, Fernandez et al [34] ont proposé de combiner le plus grand sous-graphe commun et le super-graphe commun minimum. Cette distance indique que le sous-graphe commun le plus élevé et le super-graphe commun minimum de deux graphes G1 et G2 doivent être similaires [35].

4 Conclusion

Les graphes sont un outil de représentation et d'analyse efficace pour les systèmes complexes et les relations entre les entités. Ils sont constitués d'arêtes, qui représentent les connexions entre les nœuds, qui représentent les entités. Au fil du temps, les graphes sont devenus un sujet d'étude important en informatique. Ils sont actuellement souvent utilisés dans la représentation des données. L'avantage d'utiliser des graphes est qu'ils peuvent montrer les différences structurelles et les variations entre plusieurs objets.

Dans un certain nombre de disciplines, notamment l'informatique, l'optimisation des réseaux et l'analyse des données, il est essentiel de comprendre les principes fondamentaux des graphes et leur correspondance. Les nœuds, les arêtes,

les graphes orientés et non orientés sont quelques-unes des idées fondamentales de la théorie des graphes.

Dans les limites de ce chapitre, comme point de départ d'une étude, quelques idées de base en théorie des graphes sont révélées, qui ont été introduites à l'origine pour exprimer des problèmes combinatoires et qui ont ensuite été utilisées pour résoudre efficacement des problèmes réels. De plus, la complexité du problème d'appariement de graphes et les métriques de similarité qui permettent la comparaison de graphes entre deux graphes donnés ont également été déterminées.

Chapitre 2 :

Etat de l'art sur les méthodes *d'appariement de graphe*

1 Introduction

Les méthodes d'appariement graphie jouent un rôle important dans de nombreuses applications, notamment la vision par ordinateur, la bioinformatique. Ils sont utilisés pour trouver des correspondances entre les nœuds et les arêtes de deux graphes, généralement pour déterminer les similitudes entre eux. La correspondance des graphes est un problème difficile et de nombreuses façons ont été suggérées pour le résoudre. Nous proposons ici les dernières méthodes d'appariement des graphes et examinons les principales méthodes qui ont été proposées et utilisées pour résoudre ce problème.

Ces méthodes d'appariement de graphes sont réparties en deux classes d'approches : les approches exactes et les approches inexactes. Le but de la première classe est de trouver une correspondance parfaite entre les éléments de deux groupes afin d'améliorer un certain niveau de coût ou de profit. Le but de l'appariement inexact est de trouver la compatibilité entre les éléments de deux ensembles n'est pas nécessairement idéal, mais peut être considéré comme satisfaisant en termes de coût ou d'avantage. Cela peut être utilisé dans les cas où une conformité complète n'est pas nécessaire ou possible, ou pour traiter les incohérences ou l'incertitude dans les données.

Dans ce chapitre, nous présentons différentes méthodes et moyens de faire correspondre le graphe de chaque catégorie et les algorithmes testés pour trouver des similitudes entre les graphes.

2 Méthode de comparaison de graphes

Le processus d'évaluation de la similarité entre deux graphes se fait en comparant les étiquettes qu'ils ont en commun. Pour tenir compte des différences dans les graphes comparés, un sommet peut être associé à aucun, à un ou à un ensemble de sommets de l'autre graphe [1].

Le choix de la meilleure technique de comparaison de graphes est essentiel car elle affecte la fiabilité et l'exactitude des résultats. Distance d'édition de graphe, isomorphisme de graphe, isomorphisme de sous-graphe, et appariement de graphe spectral sont quelques-unes des approches qui sont fréquemment employées pour la comparaison de graphe.

Chacune de ces méthodes présente des avantages et des inconvénients, et la façon dont nous choisissons dépend des exigences particulières de notre mission Comme les types de graphes que nous comparons, les types de comparaisons que nous faites et les ressources de calcul disponibles.

2.1 Mesurer la similarité

La similarité est une mesure qui reflète à quel point deux objets ou caractéristiques sont similaires, et est généralement représentée sur une échelle allant de -1 à +1 ou normalisée à [0,1]. Lors de la mesure de la similarité entre des graphes étiquetés représentant des symboles graphes, des comparaisons et des combinaisons de divers types d'informations sont nécessaires. Cependant, la méthode

de Champin ne fonctionne qu'avec des attributs symboliques sur les arcs et les nœuds et n'est donc applicable que dans des cas spécifiques [1].

2.2 Mise en correspondance entre graphes

Tester toutes les correspondances potentielles dans les deux graphes et garder celle qui donne le plus de similitudes est la technique la plus simple pour identifier la meilleure correspondance. En raison de la croissance exponentielle de l'espace de recherche avec la taille de graphes, la demande de ressources informatiques augmente rapidement. Pour cette raison, des algorithmes d'une complexité moindre comme les procédures de séparation et évaluation ou les algorithmes gloutons sont fréquemment employés pour éviter une recherche approfondie, même si cela signifie obtenir un résultat moins-que-idéal [1]. La cartographie entre les graphes fournit un moyen de représenter et d'analyser les relations et les liens entre les objets, aide également à créer une correspondance entre deux ou plusieurs éléments des graphes, ce qui peut révéler des similitudes, des différences et des associations. C'est pourquoi il est important dans de nombreux domaines, y compris l'informatique et les mathématiques

2.3 Distances de graphes

Distance de graphe fait référence à la mesure de la similarité ou de la différence entre deux graphes. Il existe différentes façons de mesurer la distance entre les graphes, telles que la distance de Jaccard, la distance de Sørensen-Dice et la distance de Hamming. Chacune de ces méthodes utilise une combinaison de différences dans les nœuds et les arêtes du graphe pour déterminer leur similarité. La distance de graphe peut être utilisée pour diverses tâches telles que la reconnaissance de formes, la comparaison de graphes et l'analyse de graphe.

2.4 Isomorphisme et homomorphisme

Une relation un-à-un entre deux graphes qui respecte à la fois les propriétés des nœuds et les arêtes est connu comme "isomorphisme de graphe." Lors de l'examen de l'isomorphisme des sous-graphes, nous prenons en compte deux graphes de tailles différentes, |G' | = n' et |G| = n. Une relation entre les nœuds du graphe G' et tous les sous-graphes du graphe G causés par n' nœuds est recherchée afin de vérifier si G' est un sous-graphe de G. En général, nous cherchons à établir l'égalité entre deux graphes fournis. On définit les notions de l'homomorphisme et de l'isomorphisme [4].

2.4.1 Un homomorphisme

Un homomorphisme de graphe est une fonction entre deux graphes qui maintient leur intégrité structurelle, c'est-à-dire qu'il traduit les sommets adjacents d'un graphe en sommets adjacents de l'autre. Une carte de VG à VH qui prend des arêtes à arête est connue sous le nom de lissage du graphe G au graphe H [3].

Un homomorphisme de graphe F d'un graphe G = (A, B) vers un graphe G' = (A', B') s'écrit :

f : G -> G'

C'est une application f : A -> A' de l'ensemble des sommets de G vers l'ensemble des sommets de G' tel que {u, A} ∈ B ⇒ {f(u), f(A) ∈ B'.

2.4.2 Isomorphisme de graphes

Soient G1 = (V1, E1, α1, β1) et G2 = (V2, E2, α2, β2) deux graphes.
L'isomorphisme de graphes est une fonction bijective :

$$f : V1 \rightarrow V2$$

Cette fonction doit satisfaire les conditions suivantes :

1. ∀ u ∈ V1, α1(u) = α2(f(u)),
2. ∀ e1 = (u, v) ∈ E1, ∃ e2 = (f(u), f(v)) ∈ E2, tel que β1(e1) = β2(e2), et
3. ∀ e2 = (u, v) ∈ E2, ∃ e1 = (f −1 (u), f −1 (v)) ∈ E1, tel que β1(e1) = β2(e2)

Afin de déterminer si deux graphes sont isomorphes, nous devons identifier une fonction biunivoque (f) qui traduit les nœuds de deux graphes un à un tout en maintenant la connexité et les étiquettes de nœud. Formellement, si et seulement si leurs étiquettes sont identiques, un nœud u du graphe G1 et un nœud f(u) du graphe G2 sont mis en correspondance, soit α1(u) = α2(f(u)). Également, Si et seulement si leurs étiquettes sont égales, alors une arête e1 = (u, v) du graphe G1 correspond à une arête e2 = f(e1) du graphe G2 (e2). De plus, si et seulement si f(u) et f(v) sont adjacents, deux nœuds u et v contigus dans G1 sont appariés respectivement à f(u) et f(v) dans G2[5].

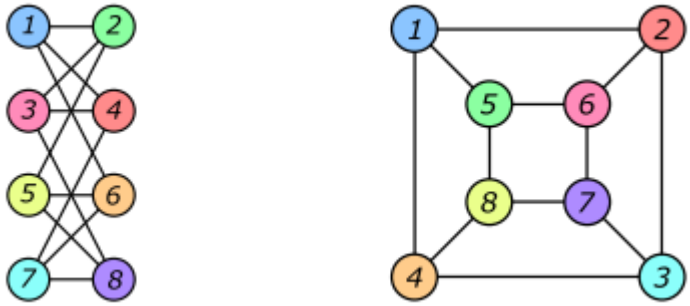

Figure 2.1 : Exemple de deux graphes isomorphes

2.4.3 Isomorphisme de sous-graphes

Soient G1 = (A1, B1, α1, β1) et G2 = (A2, B2, α2, β2) deux graphes. L'isomorphisme de sous-graphes est une fonction injective :

$$f : A1 \rightarrow A2$$

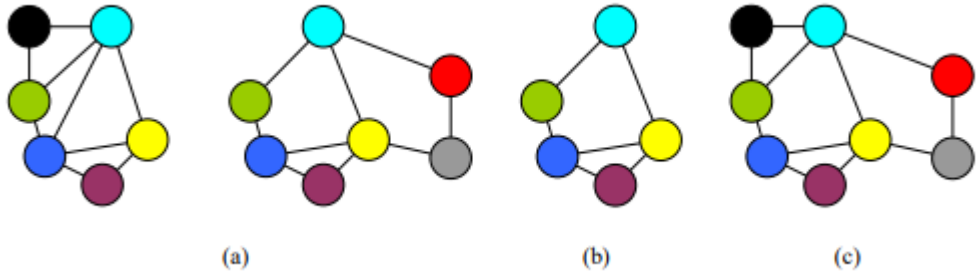

Figure 2.2 : (a) deux graphes g1 et g2, (b) un sous-graphe maximal commun entre g1 et g2, (c) un supergraphe minimal commun entre g1 et g2.

Les termes " supergraphe minimum commun" (SMC) et "sous-graphe maximal commun" (SMC) ont été proposés afin de définir les limites de l'isomorphisme des (sous-)graphes et d'établir une mesure de similarité entre les graphes [5].

3 Appariement de graphes

Le problème d'appariement de graphes a connu des progrès significatifs dans de nombreux domaines scientifiques entre les études initiales des années 1970 et des études plus récentes. Dans ce travail, des méthodes de calcul d'appariement supplémentaires ont été développées en plus des applications d'appariement de graphes. En particulier, la communauté de la reconnaissance de formes a accordé beaucoup d'attention à ce problème et de nombreuses solutions au problème de correspondance de graphes ont été proposées. Appariement exact de graphes et Appariement inexact de graphes peuvent être appliquées à ces œuvres. Nous passerons en revue les différentes méthodes de correspondance de graphes de chaque catégorie dans cette section [5].

3.1 Appariement exact de graphes

Il existe un certain nombre d'exigences pour la correspondance exacte des graphes. Trouver une fonction d'association entre les nœuds de deux graphes est la première étape de ce processus, qui consiste à maintenir la structure de sorte que chaque paire de nœuds adjacents dans le premier graphe corresponde à une paire de nœuds adjacents dans le deuxième graphe. Une correspondance parfaite entre deux graphes conserve à la fois la structure et les étiquettes, ce qui signifie que chaque nœud d'un graphe correspond à un nœud avec la même étiquette dans l'autre. La connexion entre les nœuds de deux graphes doit maintenir la contiguïté pour qu'ils soient mappés exactement, donc si une arête relie deux sommets du premier graphe, ces nœuds doivent également être mappés sur le graphe [2,5].

3.1.1 Arbre de recherche

La base de la majorité des algorithmes de comparaison exact est une représentation arborescente de recherche. Chaque correspondance est censée être représentée sous forme de chaîne dans un arbre de recherche. Pour ce faire, nous représentons chaque connexion entre deux sommets dans l'arbre de recherche

comme un nœud, et chaque correspondance entre deux arêtes comme un lien entre deux nœuds. Ce faisant, il est possible de diviser les calculs de correspondance et d'éviter de faire plusieurs fois le même calcul. En coupant des branches spécifiques de l'arbre, cela vous permet également d'éviter d'effectuer certains calculs. Il convient de noter qu'un besoin pour obtenir une solution exacte est que toutes les solutions peuvent être trouvées en naviguant dans l'arbre de recherche [6]. L'idée fondamentale est qu'une correspondance partielle, initialement vide, est étendue de manière itérative en y ajoutant de nouvelles paires de nœuds correspondants. La paire est sélectionnée en utilisant certaines conditions nécessaires qui garantissent sa compatibilité avec les contraintes imposées par le type de correspondance par rapport aux nœuds cartographiés jusqu'à présent. En fin de compte, l'algorithme trouve soit une correspondance parfaite, soit atteint le point où le mappage partiel actuel ne peut plus être étendu en raison de Restrictions de correspondance. L'algorithme s'arrête si tous les mappages possibles qui satisfont les contraintes ont déjà été essayés. Ce type d'algorithme est utilisé car plusieurs stratégies de mise en œuvre différentes diffèrent dans l'ordre des correspondances partielles visitées. Une grande caractéristique de ces algorithmes est qu'ils peuvent être très simples Il a été adapté pour tenir compte des attributs des nœuds et des arêtes dans la restriction correspondance souhaitable. Ceci est important pour les applications de relations publiques où les attributs jouent souvent un rôle majeur en réduisant le temps de calcul pour l'appariement [7].

3.1.2 L'algorithme d'Ullmann

Malgré son apparition en 1976, cet algorithme reste l'un des plus populaires et des plus efficaces pour les tests d'isomorphisme de graphes. Ullmann a implémenté une procédure de raffinement qui élague autant de branches que possible de l'arbre de recherche en profondeur d'abord. En fait, l'heuristique d'Ullmann filtre les correspondances suivantes au cours de chaque nœud de l'arbre pour exclure celles qui ne sont pas cohérentes avec la correspondance considérée [27].

L'algorithme d'Ullmann est largement utilisé comme algorithme de classe de recherche d'arbre primaire pour l'isomorphisme de sous-graphe. Il commence par obtenir des correspondances possibles pour chaque sommet u en sélectionnant tous les sommets v dans le graphe de données, puis en élaguant globalement l'espace de recherche à l'étape 2. Cela se fait en vérifiant que chaque sommet candidat v dans le graphe de données qui correspond au sommet u Dans la requête, le graphe doit avoir des enfants qui correspondent à tous les enfants de u. Ullmann suggère de traiter les sommets de requête dans l'ordre croissant, ce qui permet de rencontrer et d'élaguer davantage de branches au début du processus. L'algorithme garantit qu'aucun sommet n'est utilisé pour correspondre à plus d'un sommet de requête, et une fois que la procédure de recherche atteint la profondeur, en tenant compte des conditions de la procédure d'élagage, l'isomorphisme est trouvé. Les algorithmes concluent en effectuant un processus de retour en arrière jusqu'à ce que tous les chemins de recherche soient entièrement explorés [27].

La figure 2.3 résume les étapes de l'algorithme Ullmann [1].

1. **Début**

2. **pour** $k = i + 1$ à $|V|$

3. **pour** $j = 1$ à $|V'|$

4. **si** $P[k, j] = 1$ **alors**

5. **pour** $l = 1$ à $i - 1$

6. vérifier les contraintes suivantes de consistance des arcs:

(1) pour chaque arc $e = (v_k, v_l) \in E$
existe est-il un arc $e' = (w_j, w_{f[l]}) \in E'$ tel que $L_E(e) = L'_E(e')$

(2) pour chaque arc $e' = (w_j, w_{f[l]}) \in E'$
existe est-il un arc $e = (v_k, v_l) \in E$ tel que $L_E(e) = L'_E(e')$

7. si (1) ou (2) sont FAUX alors P[k, j] := 0

11. **finsi**

il existe une ligne k dans P telle que $P[k, j] = 0$ pour $j = 1 \ldots . |V'|$ **alorsretourner** FAUX;
sinon retourner VRAI;

12. **fin.**

Figure 2.3 : Algorithme d'Ullmann

Cependant, un inconvénient de cet algorithme est qu'il ne prend pas en compte les caractéristiques des nœuds et des arêtes lors du processus de mise en correspondance. D'autres algorithmes heuristiques visant à améliorer le temps d'exploration dans l'arbre de recherche ont été proposés, notamment l'algorithme VF et ses versions ultérieures (VF2, VF++ et VF3) [27].

3.1.3 Algorithme Vf2

L'algorithme VF2, qui est illustré sur la figure 2.4, vise à trouver un appariement entre deux graphes G1(N1, B1) et G2(N2, B2) tout en préservant leur structure en arc. La relation intime entre les deux graphes est décrite par la représentation de l'espace d'état. La solution est un ensemble de couples $M \subset N1 \times N2$ de correspondances bijectives (n, m), où $n \in G1$ et $m \in G2$. Le processus avance à travers une série d'états, dont chacun dénote une solution partielle de la correspondance du graphe. Les solutions partielles courantes et les sous-graphes correspondants G1(s) et G2(s) de G1 et G2 respectivement à l'état s sont représentés par le sous-ensemble M(s) de M [1].

Pour assurer l'efficacité du processus, l'algorithme utilise des contrôles de fiabilité pour chaque état, de l'état initial s0 à l'état solution sn. Si l'état s est susceptible d'être étendu à certaines étapes dans l'avenir, on garde cet état, et ceux qui ne remplissent pas les conditions sont préalablement éliminés. Ces conditions, appelées règles k-look-ahead (généralement k=1 ou k=2), dépendent de l'application spécifique. Les auteurs de l'étude proposent deux grands groupes de règles : les "règles de faisabilité syntaxique" et les "règles de faisabilité sémantique". Pour réduire l'espace de recherche, une fonction booléenne appelée fonction de fiabilité

F(s, n, m) est utilisée. Dans les cas où la valeur renvoyée est vraie, cela indique que l'état s' obtenu en ajoutant le couple candidat (n, m) à l'état s (s' = s ∪ p, où p= (n, m)) est isomorphe [1].

∧ (ET)

F(s, n, m) = Fsyn(s, n, m) ∧ Fsem (s, n, m)

Fonction de fiabilité syntaxique La fonction de fiabilité sémantique **Fsem** dépend des attributs des graphes, alors que **Fsyn** ne dépend que de la structure des graphes.

Algorithme VF2(s)

Entrée : un état intermédiaire s ou l'état initial s_0 tel que $M(s_0)$ = NULL
Sortie: Les appariements entre les deux graphes

1. **début**
2. **si** $M(s)$ comprend tous les sommets de G_2 **alors** renvoyer $M(s)$
3. **sinon**
4. Calculer $P(s)$ ensemble de couples candidats pour l'inclusion dans $M(s)$
5. **pour chaque** $(n, m) \in P(s)$
6. **si** $F(s, n, m)$ **alors**
7. Calculer l'état s' (obtenu en ajoutant le couple (n, m) à $M(s)$)
8. Appeler VF2 (s')
9. **fin si**
10. **fin pour**
11. Stocker la structure de données
12. **fin si**
13. **fin**

Figure 2.4 : Algorithme VF2

3.1.4 Algorithme Nauty

Selon divers auteurs, l'algorithme de McKay [37] est considéré comme l'approche la plus rapide pour l'isomorphisme de graphes. Cette méthode consiste à organiser les nœuds dans les deux graphes en fonction d'un étiquetage canonique. Pour ce faire, l'algorithme calcule une étiquette unique pour chaque nœud, qui décrit la connexion du nœud avec les autres nœuds du graphe. Par la suite, ces étiquettes sont attribuées à leurs nœuds respectifs et utilisées pour organiser les nœuds dans les graphes. Enfin, l'algorithme vérifie l'égalité entre les représentations canoniques pour déterminer l'isomorphisme des deux graphes [27].

L'approche de McKay pour l'appariement de graphes consiste à représenter chaque graphe avec sa forme canonique, qui est ensuite utilisée pour vérifier l'isomorphisme en comparant les deux représentations canoniques. Cette méthode est plus rapide que la construction d'un étiquetage, qui nécessite généralement un temps exponentiel. Malgré cela, la recherche a montré que cet algorithme fonctionne mieux que les autres algorithmes de correspondance de graphes [27].

3.2 Appariement inexact de graphes

L'appariement inexact de graphes est un type d'algorithme d'appariement de graphes utilisé pour faire correspondre des modèles dans des graphes. Parce qu'il permet un certain degré d'inexactitude dans les résultats, il diffère des algorithmes d'appariement exact de graphes et produit des résultats plus fiables et plus précis. Lorsque le calcul de la solution correcte prendrait trop de temps ou serait trop complexe, des procédures d'appariement de graphes inexacts seraient utilisées.

3.2.1 Arbre de recherche

Une correspondance inexacte peut également être obtenue à l'aide d'une recherche arborescente avec retour en arrière. Dans ce cas, le coût de la correspondance partielle qui a été obtenue jusqu'à présent et une estimation heuristique du coût de correspondance pour les nœuds restants servent souvent de principes directeurs de la recherche. L'algorithme A* utilise ces informations pour décider de l'ordre dans lequel l'arbre de recherche doit être exploré, tandis qu'un algorithme de branchement et de liaison peut l'utiliser pour élaguer les chemins improductifs. Si l'heuristique donne une bonne estimation du coût d'appariement futur Dans le dernier cas, l'algorithme trouve la réponse très rapidement ; mais, si ce n'est pas le cas, le besoin en mémoire est significativement plus élevé que pour la technique de branchement et de liaison [7].

La véritable distance d'édition du graphe a été tentée pour la première fois en 1983, où considéré comme opérations d'édition de base la substitution de nœuds et d'arêtes ainsi que la division et la fusion de nœuds [7]. Sur la base de l'algorithme A∗ et avec l'heuristique appropriée, un certain nombre de techniques ont été créées pour la détermination de la distance d'édition de graphe. Un autre problème NP-difficile est l'optimisation du coût d'appariement approximatif entre deux graphes. En réalité, la taille de l'arbre de recherche croît de manière exponentielle par rapport aux diamètres des graphes considérés. Des algorithmes sous-optimaux ont été suggérés comme solution à ce problème. L'idée directrice d'un algorithme d'appariement sous-optimal est de calculer un minimum local du coût d'appariement plutôt qu'un minimum global [5].

3.2.2 Les techniques basées sur la théorie spectrale

La méthode d'Umeyama est l'une des études originales que a utiliser la théorie spectrale pour l'appariement de graphes [5]. Les valeurs propres et les vecteurs propres de la matrice d'adjacence ou matrice laplacienne sont utilisés pour décrire la topologie des graphes. Le problème bien étudié du calcul des valeurs propres et des vecteurs propres peut être résolu en temps polynomial. Mais malgré les avantages de l'application de tous les outils mathématiques, les approches spectrales présentent deux inconvénients principaux. Certaines approches spectrales ne peuvent gérer que des graphes étiquetés non étiquetés ou contraints (par exemple, uniquement des alphabets d'étiquettes contraints ou des poids réels fournis aux arêtes), en raison de la faiblesse de leur représentation des attributs de sommet et d'arête de représentation en reconnaissance de formes. Deuxièmement, parce qu'ils ne peuvent pas gérer les graphes affectés par le bruit comme les défauts ou les sommets

supplémentaires, la décomposition propre est sensible aux problèmes structurels [10].

3.2.3 Les techniques basées sur l'apprentissage

Les réseaux de neurones récurrents sont utilisés dans une technique d'apprentissage dans le but de récupérer des images 2D à partir de grandes bases de données. De plus, ils utilisent des méthodes L'appariement inexact de graphes. Pour comparer les graphes, une version floue de la distance de Hausdorff qui ne prend en compte que les valeurs des attributs d'arête est suggérée. Un algorithme d'appariement de graphes bayésien qui exploite une extension de la technique de relaxation de Wilson et Hancock est utilisé pour effectuer la procédure d'appariement [4]. Les techniques d'apprentissage automatique peuvent être utilisées pour résoudre une variété de problèmes, tels que la classification, la régression, la prédiction, la reconnaissance de formes, la segmentation d'image, etc.

4 Conclusion

Les développements récents des techniques d'appariement de graphes se sont concentrés sur l'augmentation de l'évolutivité et de l'efficacité des algorithmes ainsi que sur l'incorporation de connaissances passées et de contraintes supplémentaires dans le résultat. De plus, des techniques d'apprentissage en profondeur ont été appliquées aux problèmes d'appariement de graphes, il est donc prévu que cette tendance se poursuive à l'avenir avec des progrès supplémentaires dans la précision et l'efficacité des algorithmes d'appariement de graphes.

Chapitre 3 :

Appariement inexact de graphes

1 Introduction :

L'appariement inexact de graphes est le problème de trouver des correspondances entre deux graphes qui ne sont pas nécessairement identiques, mais qui ont des structures similaires. C'est un problème fondamental en informatique qui consiste à trouver des correspondances entre les sommets et les arêtes dans deux graphes dissemblables. Ce problème se pose dans de nombreuses applications telles que la reconnaissance de formes, le traitement d'images, l'analyse de données et la biométrie.

En raison de la non-unicité des correspondances, du bruit des données et de la présence de valeurs aberrantes, l'appariement inexact de graphes est un défi difficile. Traiter des graphes énormes ou des graphes avec une architecture complexe peut rendre le problème encore plus difficile. En conséquence, un certain nombre d'algorithmes et de méthodes ont été suggérés pour résoudre ce problème.

Ce chapitre vise à donner un aperçu de l'appariement inexact de graphes, en commençant par une définition formelle du problème et de ses variantes. Il présente ensuite des techniques d'appariement de graphes inexacts et passe en revue certains des algorithmes populaires.

2 Appariement inexact de graphes :

Nous ignorons les nœuds des graphes avec moins valeur de centralité avant de calculer un score de similarité à l'aide de distance d'edition de graphe entre deux graphes et cela pour réduire le temps de calcul de l'appariement inexact de graphe [11] .

2.1 Définition :

La contraction des nœuds de t-centralité est le processus de contraction de t nœuds à partir d'un graphe G avec les valeurs de centralité les plus faibles d'une mesure de centralité donnée [11].

2.2 Isomorphisme de graphes :

De nombreuses études ont été menées au cours des dernières décennies, abordant les divers problèmes d'isomorphisme sous divers angles et utilisant un large éventail de méthodologies. La majorité des problèmes qui ont un impact pratique significatif sont NP-difficiles, mais pour l'isomorphisme des graphes non étiquetés, aucune solution à la question de savoir s'il est NP-complet ou se trouve dans P n'a pas encore été découverte [4].

2.3 Problèmes inexacts :

Même si deux graphes sont extraits de la même source, ils sont rarement parfaitement égaux en pratique. L'incertitude dans le processus de caractérisation des attributs pour l'extraction, les paramètres bruyants, etc., sont les principales causes des changements structurels. Par conséquent, les algorithmes

d'isomorphisme de graphe tolérants au bruit devraient être capables de reconnaître la correspondance entre deux graphes même lorsqu'il existe des différences minimes. Cela nous amène aux vagues difficultés d'isomorphisme cité par Conti et all [13], décrit dans sa classification [4].

2.4 Distances de graphes :

Trouver la fonction m : V → V' entre leurs nœuds pour qu'une métrique de dissimilarité soit réduite est problème d'isomorphisme inexact des graphes, étant donné deux graphes de même taille G = (V, α, β) et G' = (V', α' , β'). Le concept fondamental est de trouver un ensemble d'opérateurs qui convertit le premier graphe en le second. Les opérateurs sont l'insertion, la suppression et la substitution d'arêtes et de nœuds. Chacun de ces opérateurs a un coût, qui est compté sur toute la séquence. La séquence qui entraîne le coûts le plus faible est ensuite utilisée pour calculer la distance. Le problème de trouver cette séquence est NP-difficile. Trouver les valeurs appropriées pour les différents coûts est difficile et dépend de l'application. La distance d'édition est donc assez large, mais elle présente toujours un défi lorsqu'elle est appliquée à des problèmes réels [4].

2.5 Appariements multivoques:

Appariements multivoques, ou "nombreuses correspondances", est une sorte de correspondance de graphes qui implique deux graphes et permet à chaque nœud d'être mis en correspondance avec un groupe de nœuds dans l'autre graphe. La comparaison de deux types de graphes différents, tels que deux ontologies différentes ou deux photos, est facilitée par l'utilisation de ce type de correspondance. En raison de l'appariement plus flexible et efficace qu'il permet, il est également utile pour déterminer la solution la plus idéale. L'algorithme de hachage donné (et ses paramètres) peut être utilisé pour exécuter une application multithread.

Une expansion de distance d'édition de graphe est suggérée par Ambauen et all [14]. A la liste des opérations, elles incluent également la subdivision d'un sommet et la fusion de sommets. La définition de la division est qu'elle remplace un sommet par un ensemble de nouveaux sommets, alors que la définition de la fusion est qu'elle effectue l'opération inverse. Même s'il est désormais possible de gérer certaines instances de correspondance plusieurs-à-plusieurs, la stratégie est contrainte. En réalité, vous devez fusionner les deux graphes si vous voulez faire correspondre les sommets v1 et v2 du premier graphe avec v1' du second. Étant donné que v2 n'est pas également mappé sur v2', il est alors impossible de faire correspondre le sommet v1 avec un autre sommet v2' [4].

2.6 Appariements pour le recalage :

L'enregistrement de graphe est une autre application de correspondance, principalement utilisée dans un contexte visuel. Son but est d'aligner des graphes, permettant l'identification des sommets correspondants au niveau local. Cette technique est essentielle pour une compréhension plus approfondie des graphes, en

particulier lorsque vous tentez de localiser un emplacement spécifique sur une photo satellite avec des informations limitées [4].

3 Techniques d'appariement inexact :

Il est possible de trouver des correspondances entre deux ensembles de données même lorsqu'elles ne correspondent pas exactement à l'aide de Les Techniques d'appariement inexact, également connues sous le nom de Fuzzy Matching en anglais.

3.1 Les techniques basées sur l'optimisation continue :

L'appariement de graphes est un problème d'optimisation discret par nature, et la plupart des solutions qui ont été suggérées utilisent directement des graphes. Résoudre le problème d'appariement de graphes en résolvant un problème continu équivalent est une approche très différente. Les trois étapes qui composent cette approche sont les suivantes : Le problème d'appariement de graphes est d'abord reformulé comme un problème continu. Deuxièmement, une procédure d'optimisation est utilisée pour résoudre le problème continu. La solution continue découverte est ensuite renvoyée dans le domaine discret d'origine. Les méthodes d'optimisationcontinue des problèmes de la deuxième étape garantissent la découverte d'une solution moins qu'idéale. La solution finale, produite par l'approximation finale de l'étape de discrétisation, peut ne pas toujours atteindre l'optimalité locale. Pourtant, en raison de son coût de calcul extrêmement faible, qui est typiquement polynomial, l'approche basée sur l'optimisation continue est particulièrement utile dans de nombreuses situations [12].

3.2 Les techniques basées sur le clustering :

L'appariement inexact des graphes peut être réalisé à l'aide de diverses techniques de regroupement. Ils comprennent des techniques d'appariement de graphes spectraux, des techniques de regroupement par projection d'espace propre (EPC) et un regroupement hiérarchique agglomératif [15]. Les techniques EPC trouvent une variété de correspondances entre deux graphes en projetant des sous-espaces de vecteurs propres. Les clusters de nœuds de deux graphes sont regroupés à l'aide d'une technique appelée clustering hiérarchique agglomératif en fonction de leur similarité. Les vecteurs propres des deux graphes sont utilisés dans la méthode d'appariement des graphes spectraux pour déterminer quels graphes ont le meilleur accord [16].

3.3 Les techniques basées sur les noyaux de graphes :

compte tenu des variations entre les deux graphes. Afin d'identifier des correspondances approximatives entre les sommets et les arêtes des graphes, les méthodes basées sur les noyaux de graphes examinent les structures des graphes et leurs attributs. Ils s'appuient fréquemment sur des méthodes mathématiques telles que la factorisation matricielle et la transformation de Fourier. Ces techniques divisent les graphes en modèles, les comparent, puis pèsent les similitudes pour produire un score de similitude [17] [18].

3.4 Les techniques basées sur Réseaux de neurones artificiels :

Il existe également une classe de techniques d'appariement de graphes basées sur des réseaux de neurones artificiels selon [21]. En maximisant un critère d'énergie d'entrée, a utilisé le réseau de neurones de Hopfield pour l'appariement de graphes homomorphes en 1995. En 1997, l'idée de structure récursive généralisée a été utilisée pour créer des réseaux de neurones supervisés pour classer les graphes. En combinant des modèles de réseaux de neurones artificiels et de réseaux de croyances, et en présentant un cadre complet en 1998, où les connexions entre les variables de données sont représentées par des graphes acycliques dirigés, le traitement structurel de l'information est rendu possible. ont présenté la généralisation de la carte des caractéristiques auto-organisatrices en 2003 en substituant un graphe non orienté à la grille de neurones conventionnelle .En utilisant une stratégie de raffinement neuronal pour condenser l'espace de recherche, puis en effectuant un processus de réduction d'énergie.

3.5 les Techniques basées sur l'incorporation de graphes :

La base des techniques d'intégration de graphes est une application sur un espace vectoriel. Deux catégories largement distinctes sont principalement couvertes par les techniques d'incorporation de graphes. Le premier comprend des techniques pour mapper la collection de sous-structures d'un graphe sur un groupe de points dans un espace vectoriel, où les sous-structures seront mappées sur des points d'espace vectoriel proches. La deuxième catégorie comprend les méthodes qui mappent des graphes entiers sur des points de l'espace vectoriel ; dans ce cas, les graphes associés seront transférés sur des points spatiaux vectoriels proches. La première stratégie est l'intégration isométrique, dans laquelle les approches suggérées utilisent une métrique de similarité de graphe et cherchent à identifier une application aux vecteurs qui préserve la métrique. La deuxième stratégie est connue sous le nom d'intégration spectrale et base ses techniques suggérées sur l'utilisation des caractéristiques spectrales des graphes, telles que celles liées aux valeurs propres et aux vecteurs propres. La troisième stratégie est l'intégration de sous-modèles, dans laquelle les techniques suggérées cherchent à catégoriser certains types de sous-modèles particuliers dans les graphes à intégrer [12].

3.6 Autres techniques d'appariement inexact :

Une stratégie plus récente consiste à construire une fonction noyau entre les graphes ou, pour le dire autrement, une fonction de similarité qui se comporte comme un produit scalaire. Il s'agit d'une extension de plusieurs approches initialement énoncées pour la correspondance exacte qui ont été conçues pour gérer les instances erronées. Etant donné que les méthodes reposent fréquemment sur l'une des stratégies précédentes, plus ou moins modifiée pour s'adapter aux critères mathématiques, cette approche est plus difficile à décrire [6]. De nombreuses méthodes ont été développées, il existe plus de catégories d'appariement de graphes imparfaits qui n'ont pas été décrites auparavant. L'utilisation d'algorithmes évolutifs et de méthodes basées sur propriétés locales et techniques de recherche taboue [12].

4 Algorithmes d'appariement inexact :

Des algorithmes pour une mise en correspondance approximative de chaînes ou de motifs pouvant inclure des erreurs ou des variations sont utilisés. Vous trouverez ci-dessous quelques exemples d'algorithmes de correspondance inexactes populaires.

4.1 Algorithme de Greedy :

Selon sa définition, les algorithmes greedy sont des moyens de résoudre les problèmes d'optimisation en faisant des choix qui, quel que soit le résultat, produisent l'avantage le plus évident et le plus immédiat. Il fonctionne lorsque la minimisation ou la maximisation donne le résultat souhaité [19]. L'objectif fondamental d'un algorithme glouton est de construire une solution en décidant du meilleur choix disponible à chaque étape. L'algorithme prend une succession de décisions, choisissant celle qui semble la meilleure du moment à chaque étape. L'algorithme passe ensuite à la phase suivante et répète la procédure, prenant une nouvelle décision en fonction de l'état actuel de la solution.

Algorithme de Greedy est l'algorithme de base pour la cartographie des couleurs Cet algorithme est utilisé toujours le strict minimum de couleurs. De plus, la séquence dans laquelle les sommets sont traités peut parfois affecter le nombre de couleurs utilisées. Prenez les deux graphes suivants G1 et G2 de la figure 3.1, par exemple. Sachez que les sommets 3 et 4 du graphe G2 sont inversés. Nous pouvons colorer le graphe G1 en utilisant 3 couleurs si nous prenons en compte les sommets 0, 1, 2, 3 et 4. Mais, si nous prenons en compte les sommets du graphe G2 (0, 1, 2, 3 et 4) nous avons besoin de 4 couleurs [19].

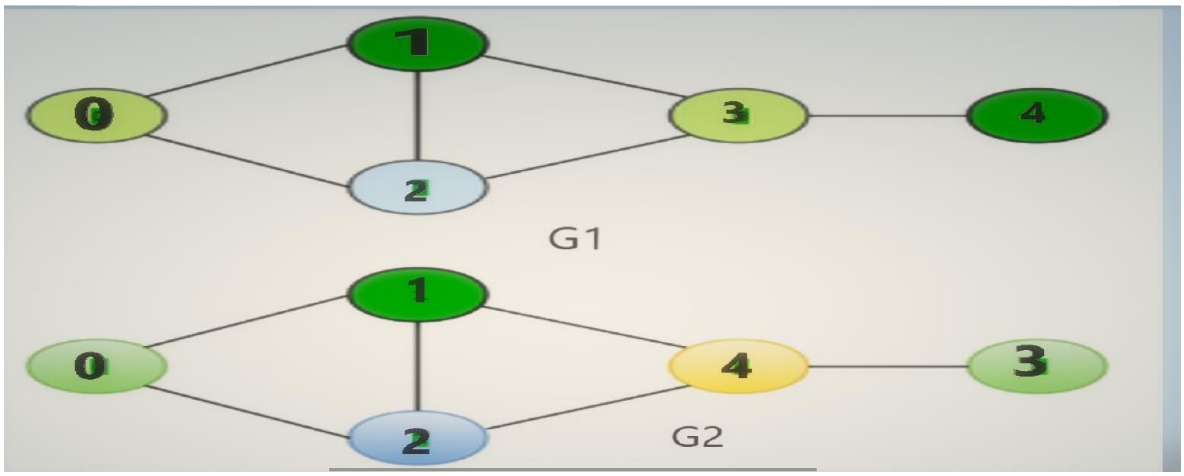

Figure 3.1 : Deux graphe pondéré G1,G2

4.2 Algorithme de recherche locale:

L'algorithme de recherche locale est un outil d'optimisation qui cherche à améliorer une solution en étudiant son environnement immédiat. Le concept fondamental est de commencer par une première réponse, puis d'étudier les solutions environnantes en faisant de petits ajustements pour trouver une meilleure solution.

Algorithm Local Search.

1. $S \leftarrow$ an arbitrary feasible solution in S.
2. While $\exists\ S' \in B(S)$ such that cost $(S') <$ cost(S),

   do $S \leftarrow S'$.
3. return S.

Figure 3.2 : Algorithme de recherche locale

Un algorithme de recherche local général est illustré à la figure 3.2 . Il consiste en un ensemble S de toutes les solutions potentielles une fonction de cost cost : S → R, une structure de voisinage B : S → 2S ,et un oracle qui, étant donné toute solution S, trouve une solution S' ∈ B(S) (si possible) telle que cost(S ') < cost(S). La réponse Si cost(S) > cost(S') pour tout S' ∈ B(S), alors S∈ S est appelée la solution localement optimale ; la technique de la figure 3.2 donne toujours une telle réponse. Pour divers problèmes et techniques, une fonction de coût et une structure de voisinage B différentes seront utilisées [20].

4.3 Algorithme Séparation et évaluation (Branch and Bound) :

Les problèmes d'optimisation combinatoire sont généralement résolus en utilisant le paradigme de la branche et de la construction d'algorithmes. Dans les pires scénarios, il peut être nécessaire d'explorer toutes les combinaisons potentielles de solutions, car ces problèmes sont souvent exponentiels en termes de complexité temporelle. L'approche Algorithme de Branch and Bound résout ces problèmes assez rapidement [19]. Le concept fondamental Trouver le chemin le plus court dans un graphe ou le débit maximum dans un grapheé sont deux exemples de problèmes d'optimisation qui peuvent être résolus à l'aide de l'algorithme de résolution de problèmes branch and bound. Il fonctionne en examinant un espace de recherche et en développant progressivement une solution tout en supprimant les branches défavorables de l'arbre de recherche.

La figure 3.3 montre un pseudo-code pour l'algorithme Branch and Bound où **L** représente la liste d'attente et **S** l'espace de recherche global [22].

Algo B&B

Tant que L $\neq \emptyset$ faire

S = suivant (L)

S_1, S_2 ..... S_n = séparer (S)

Pour chaque S_i faire

Si $V(S_i) > U$ ou S_i non réalisalble

éliminer S_i

Sinon si S_i réalisable

$S_{best} = S_i$

$U = V(S_i)(=f(S_i))$

Sinon

$L = L \cup \{ S_i \}$

Fin

Figure 3.3 : Algorithme Séparation et évaluation (Branch and Bound).

4.4 La algorithme FastPFP les grandes Appariement de graphes

Virgule fixe projetée rapide(fastPFP), et un nouvel algorithme d'appariement rapide de graphes qui peut gérer grand graphes avec plus de 1000 nœuds, est proposé dans [36]. Avec une nouvelle projection partielle doublement stochastique et une nouvelle approche en virgule fixe projetée, l'algorithme a une complexité temporelle O (n^3) par itération et une complexité spatiale O (n^2). En plus d'être évolutive, l'approche est également simple à utiliser, fiable et capable de faire correspondre des graphes attribués pondérés non orientés de différentes tailles, et à l'exception de la situation d'isomorphisme graphe, où la technique d'Umeyama est l'algorithme le plus rapide, fastPFP a constamment surpassé les algorithmes antérieurs de l'état de l'art. fastPFP, par exemple, peut faire correspondre deux graphes avec 1000 nœuds en quelques secondes. dans le chapitre suivant nous détaillerons cet algorithme suivi de son implémentation.

Conclusion :

Trouver des correspondances approximatives entre les graphes tout en se concentrant sur les variations de taille, de structure et de nomenclature est l'objectif de faire correspondre des graphes inexact. Ce domaine utilise une variété de stratégies pour résoudre ce problème, telles que les algorithmes d'appariement de graphes et les noyaux de graphes.

Étant donné que l'efficacité des différentes méthodes peut varier en fonction des propriétés uniques des graphes appariés, il n'y a pas de "meilleur" moyen pour une correspondance inexactés des graphes. Alors que les noyaux de graphes sont utiles pour capturer les similitudes structurelles entre de grands graphes avec des degrés de similitude variables , Les utilisateurs peuvent effectuer une mise en correspondance des données à l'aide de méthodes d'alignement imparfaites sans investir dans des ajustements longs et coûteux. Cependant, ces techniques doivent être utilisées avec prudence et les résultats doivent toujours être validés par une analyse plus approfondie.

L'appariement inexact de graphes est un domaine de recherche actif, avec de nombreux problèmes ouverts et des opportunités de travaux futurs. Il est important de développer des méthodes efficaces et précises pour faire correspondre des graphes inexacts afin de répondre aux exigences croissantes des applications pratiques.

Chapitre 4 :
Implémentation et Résultats

1 Introduction :

Dans ce chapitre, nous commençons par un bref aperçu des outils pouvant être utilisés pour développer notre application, tels que le langage python et quelques ses bibliothèques qui nous aident dans ce travail. On présente ensuite notre expérimentation qui vise à implémenter Appariement inexact de graphes qu'on a détaillé dans le chapitre précédent. En utilisant la théorie hungarian, et en implémentant son propre algorithme et qui sont considérés parmi les algorithmes de référence pour toute comparaison de performance avec les algorithmes d'isomorphisme de sous-graphes. Notre expérience consiste alors à trouver la symétrie de deux sous-graphes. Et au final, nous présentons le résultat que nous avons obtenu grâce à cette application.

2 Histoire de La Langage Python :

Python est un langage de programmation de haut niveau publié pour la première fois en 1991 par Guido van Rossum. C'est un langage interprété, ce qui signifie que le code est exécuté directement, sans avoir besoin de compilation. Python a été conçu pour être facile à lire et à écrire, en mettant l'accent sur la simplicité et l'élégance. Il est devenu l'un des langages de programmation les plus populaires, avec une communauté importante et active de développeurs qui contribuent à son développement et l'utilisent pour un large éventail d'applications.

Python possède une grande bibliothèque standard qui fournit une large gamme de modules pour des tâches telles que le développement Web, le calcul scientifique et l'analyse de données. Il peut fonctionner sur plusieurs plates-formes, notamment Windows, Mac OS et Linux.

Python possède une communauté importante et active de développeurs qui ont créé une large gamme de bibliothèques et d'outils pour différentes applications. Certaines bibliothèques populaires incluent NetworkX, NumPy, pandas et Matplotlib pour l'analyse de données, Flask et Django pour le développement Web, et TensorFlow et PyTorch pour l'apprentissage automatique.

Figure 4.1 : Logo de langage python.

3 Les bibliothèques utiliser :

3.1 NumPy

Le package Python NumPy est utilisé pour manipuler des tableaux. Il offre des opérations rapides et efficaces sur des tableaux de données homogènes. De nombreuses fonctions mathématiques utiles pour les opérations sur les tableaux sont également incluses dans NumPy.

À l'aide de pip, le gestionnaire de packages Python, vous pouvez installer NumPy. Vous pouvez entrer la commande suivante sur une invite de commande ou un terminal Si Python est installé sur votre machine

pip install **numpy**

Après avoir installé NumPy, vous pouvez l'importer dans votre code Python en utilisant l'instruction suivante :

import **numpy** as **np**

Cette instruction importe NumPy et lui donne l'alias "np", qui est une convention courante utilisée par les programmeurs Python. Vous pouvez ensuite utiliser les fonctions et les objets NumPy en les préfixant avec "np.".

3.2 NetworkX :

NetworkX est un package Python pour la création, la manipulation et l'étude de réseaux complexes. Il fournit des outils pour analyser et visualiser les structures de graphes et peut être utilisé pour une grande variété d'applications, y compris l'analyse des réseaux sociaux, l'analyse des réseaux biologiques.

Utilisez pip, le programme d'installation du package Python, pour installer NetworkX. Tapez la commande suivante dans un terminal ou une invite de commande :

pip install **networkx**

La version la plus récente de NetworkX et de ses dépendances sera téléchargée et installée. Après l'installation, utilisez la commande suivante pour importer NetworkX dans des sessions interactives ou des scripts Python :

import **networkx** as **nx**

Après cela, vous pouvez commencer à utiliser les classes et les fonctions du module NetworkX pour créer et modifier des graphies.

3.3 Matplotlib :

Le package Matplotlib de Python permet aux utilisateurs de créer des visualisations interactives, animées et statiques. Il fournit une variété d'interfaces de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs. L'analyse de données,

l'apprentissage automatique et le calcul scientifique sont tous des domaines qui utilisent largement Matplotlib.

Vous pouvez utiliser pip, le programme d'installation du package Python, pour installer Matplotlib. Tapez la commande suivante dans un terminal ou une invite de commande :

pip install **matplotlib**

La version la plus récente de Matplotlib et ses dépendances seront téléchargées et installées. Après l'installation, utilisez la commande suivante pour importer Matplotlib dans des sessions interactives ou des scripts écrits en Python :

import **matplotlib.pyplot** as **plt**

Après cette importation du module pyplot de Matplotlib, vous pouvez commencer à utiliser ses fonctions pour créer différents types de tracés.

3.4 Tkinter :

Tkinter est une bibliothèque Python utilisée pour créer des applications GUI (Graphical User Interface). Il s'agit d'une interface Python standard pour la boîte à outils Tk GUI, qui est largement utilisée pour créer des interfaces graphiques pour les applications de bureau. Tkinter fournit un ensemble d'outils et de widgets pour créer des fenêtres, des boutons, des menus, des zones de texte et d'autres éléments de l'interface graphique.

Tkinter peut être utilisé sur Windows, Linux et macOS car il s'agit d'une bibliothèque multiplateforme. Son principal avantage est qu'il est simple et simple à utiliser.

Utilisez le morceau de code suivant pour importer Tkinter dans votre script Python :

import **tkinter** as **tk**

Exemple d'un programme Tkinter simple qui crée une fenêtre :

```
import tkinter as tk
root = tk.Tk()
root.geometry("300x200")
root.title("Fenêtre tkinter")
label = tk.Label(root, text=" Bonjour mes amis !")
label.pack()
root.mainloop()
```

La méthode Tk() est utilisée pour créer une fenêtre lorsque ce programme importe le package tkinter. La taille de la fenêtre est définie par la méthode geometry() et le titre de

la fenêtre est défini par la méthode title(). La méthode Label() crée un widget d'étiquette avec le texte " Bonjour mes amis !", et le widget est ensuite compressé dans la fenêtre à l'aide de la méthode pack(). La boucle d'événements est lancée par la méthode mainloop(), qui actualise également l'interface graphique en réponse aux actions de l'utilisateur. La figure 4.2 est une implémentation de ce programme simple.

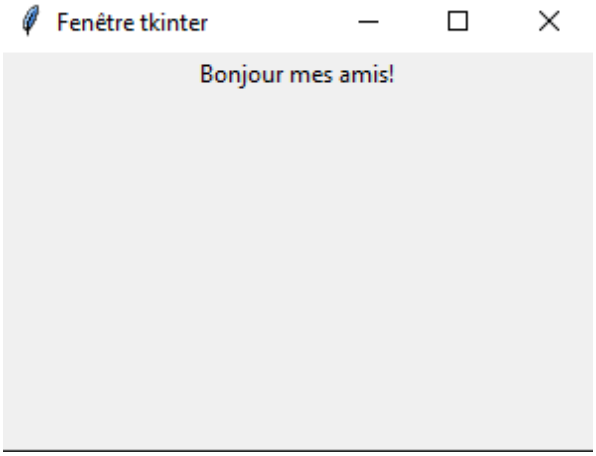

Figure 4.2 : programme Tkinter simple

4 Le problème d'affectation :

4.1 Quelques d´efinitions :

problème d'affectation:

Le problème d'affectation est un problème d'optimisation combinatoire dans lequel un ensemble de travailleurs doit être affecté à un ensemble de tâches, avec un certain coût ou une certaine efficacité associée à chaque affectation de tâche de travailleur. L'objectif est de trouver l'affectation qui minimise le coût total ou maximise l'efficacité totale.

Exemple :

Affectation	Tâche1	Tâche2	Tâche3	Tâche4
Agent1	5	6	5	2
Agent2	8	5	4	2
Agent3	5	5	7	5
Agent4	8	5	4	5

4.2 Méthode Hongroise :

La méthode hongroise, également connue sous le nom d'algorithme de Kuhn-Munkres, est un algorithme permettant de résoudre le problème d'affectation en optimisation combinatoire. La méthode hongroise est basée sur le concept d'une matrice de coût, qui représente le coût d'affectation de chaque ressource à chaque tâche. L'algorithme

fonctionne en trouvant de manière itérative un ensemble de zéros indépendants dans la matrice de coût, puis en utilisant ces zéros pour construire un ensemble de chemins d'augmentation. Un chemin d'augmentation est un chemin qui commence et se termine par des zéros et alterne entre les bords horizontaux et verticaux dans la matrice de coût.

Rappelons que la technique hongroise ne peut être utilisée que lorsque le nombre d'agents est égal au nombre d'emplois [38].

4.2.1 Étapes de la méthode hongroise :

Selon [38], nous allons exposer une série d'étapes

4.2.1.1 ÉTAPE 1 : RÉDUCTION DU TABLEAU INITIAL

Le plus petit élément est soustrait de chaque ligne du tableau initial. Ensuite, nous soustrayons également le plus petit élément de chaque colonne.

Exemple :

15	13	7	3	10
14	14	8	3	8
10	13	12	9	3
2	6	12	15	11
11	7	6	10	15

→

12	10	4	0	7
11	11	5	0	5
7	10	9	6	0
0	4	10	13	9
5	1	0	4	9

→

12	9	4	0	7
11	10	5	0	5
7	9	9	6	0
0	3	10	13	9
5	0	0	4	9

4.2.1.2 ÉTAPE 2 : ENCADRER ET BARRER DES ZÉROS

Nous cherchons la ligne avec le moins de zéros non barrés (En cas d'égalité, on tire au hasard la ligne la plus haute)

Nous choisissons arbitrairement le zéro le plus à gauche sur cette ligne.

Tous les zéros dans la même ligne ou colonne que le zéro encadré sont barrés.

Tant qu'on ne peut plus encadrer ou barrer de zéros, on répète l'opération :

Le tableau de l'exemple précédent devient :

12	9	4	0	7
11	10	5	0	5
7	9	9	6	0
0	3	10	13	9
5	0	0	4	9

Si un zéro a été inséré dans chaque ligne et colonne, c'est terminé, Nous avons la solution idéale.

Sinon, on passe à l'étape 3.

4.2.1.3 ÉTAPE 3 : MARQUER ET COLORER DES LIGNES ET DES COLONNES

a) Nous biffons toutes les lignes qui ne contiennent pas de zéros enfermés.

b) Marquez n'importe quelle colonne avec un zéro barré sur une ligne marquée.

c) Toute ligne contenant un zéro dans une colonne marquée est marquée

Répéter les opérations b) et c) alternativement jusqu'à ce qu'il n'y ait plus de lignes à marquer. Nous colorons ensuite toutes les lignes non marquées et toutes les colonnes marquées.

Continuons l'exemple précédent :

12	9	4	0	7
11	10	5	0	5
7	9	9	6	0
0	3	10	13	9
5	0	0	4	9

→

12	9	4	0	7
11	10	5	0	5
7	9	9	6	0
0	3	10	13	9
5	0	0	4	9

4.2.1.4 ÉTAPE 4 : MODIFICATION DU TABLEAU

Un tableau partiel est composé des cases non colorées.

Le plus petit élément de ce tableau partiel est soustrait de chaque cellule.

Le même élément est ajouté à toutes les cases de la tableau colorée deux fois.

Après cela, nous obtenons une nouvelle table afin de pouvoir recommencer

les étapes 1 à 3.

Application de l'étape 3 à l'exemple précédent

8	5	0	0	7
7	6	1	0	5
7	9	9	10	0
0	3	10	17	9
5	0	0	8	9

donc, dans le cas, la valeur d'affectation minimale est 9 + 5 + 5 + 4 + 9 = 32

La méthode hongroise décrite est efficace pour résoudre les problèmes d'affectation minimale. Pour résoudre un problème d'affectation maximale, le tableau initial doit être transformé en soustrayant les éléments les plus élevés du tableau de tous les éléments du tableau.

4.2.2 Complexité de l'algorithme original

Le nombre d'opérations nécessaires pour terminer chaque étape sera indiqué ci-dessous.

*** Étape 1 :** Le minimum est déterminé pour chaque ligne et colonne en utilisant n opérations, puis et opérations pour le soustraire de chaque élément. Donc O(n^2).

*** Étape 2 :** Pour parcourir tous les éléments de la matrice et vérifier si elle contient un zéro et si sa ligne et sa colonne contiennent un zéro sélectionné, il suffit de garder en mémoire un booléen pour chaque ligne et colonne indiquant si elle contient déjà un zéro sélectionné. Cela peut être fait en temps O(n^2).

*** Étape 3 :** Lorsqu'une colonne est trouvée, ajoutez les zéros nouvellement découverts à la liste des zéros non couverts. Ensuite, nous vérifions si les zéros non couverts restants sont toujours couverts. Nous ne pouvons couvrir que n lignes à la fois, et chaque couverture de ligne nécessite de traverser 2n cellules, ou n lignes et n colonnes. Donc O(n^2).

*** Étape 3' :** Les mêmes nombres d'opérations qu'à l'étape 3 sont effectuées ici. Donc O(n2).

*** Étape 4 :** De même pour l'étape 4, alors O(n^2).

Au début de l'étape 4 pour la k-iéme fois, nk est le nombre maximal de zéros indépendants dans la matrice. Si nk = nk+1, il n'y aura pas de pas 3' entre le k-ième et le (k + 1)-ième pas4. Au début de la (k + 1)-ième étape 4, nous aurons plus de lignes couvertes qu'au début de la k-ième étape 4, en supposant nk = nk+1. Nous n'effectuons des couvertures d'opérations de ligne qu'entre la k-ième étape 4 et la (k + 1)-ième étape 4.

L'algorithme présenté comporte une séquence successive d'étapes 1, 2 et 3 visant à obtenir une matrice à n zéros indépendants. Au plus O(n^2) étapes sont exécutées résultant en une complexité totale de O(n^4).

5 problème d'affectation quadratique

Le problème d'affectation quadratique est un problème bien connu en optimisation combinatoire où l'objectif est d'optimiser l'allocation de n objets, y compris les usines et les matériaux, sur n emplacements. Cela nécessite de minimiser le coût à la fois des flux inter-objets et des distances entre locations. Il s'agit d'identifier la répartition qui se traduit par le coût total minimum [39].

Il y a un ensemble de six objets et six zones dans l'exemple de la figure 4.3 ci-dessous. Le but est d'affecter tous les objets à toutes les zones, tout en minimisant la fonction objectif, qui affiche le coût global généré.

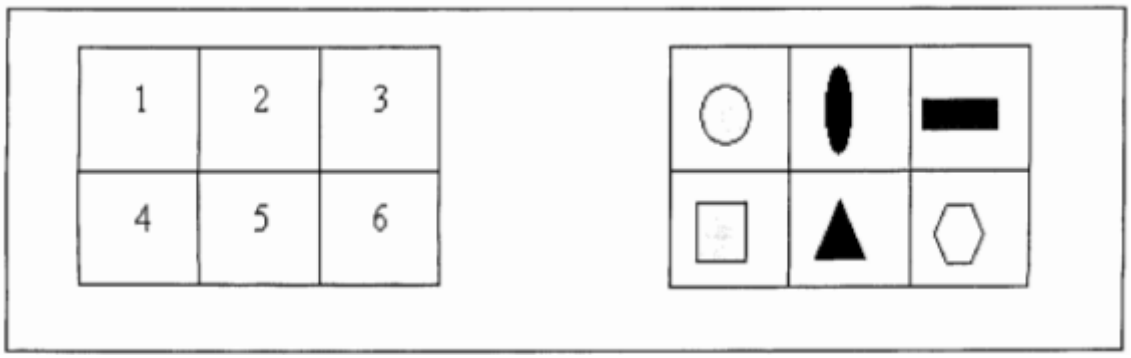

Figure 4.3 : Exemples d'emplacements et emplacement de l'équipement à chaque emplacement

Soient:

- D(**n** x **n**) La matrice avec la notation d(**i**, **J**) dénote la séparation entre la location **j** et l'emplacement **i**.
- F(**n** x **n**) Matrice de dimension (**n** x **n**) qui capture le flux entre les objets **k** et **I** sous la forme F(**k**,**l**).

Apres cela Le problème peut être écrit Le problème sous la forme suivante :

La fonction de objectif:

$$Min\ L\ =\ \sum_{n=i}^{n-1}\sum_{j=i}^{n} f(i,j)\ x\ d\big(p(i),p(j)\big)$$

Les illustrations suivantes montrent comment le problème d'implantation industrielle est formulé pour déterminer la division la plus efficace des départements :

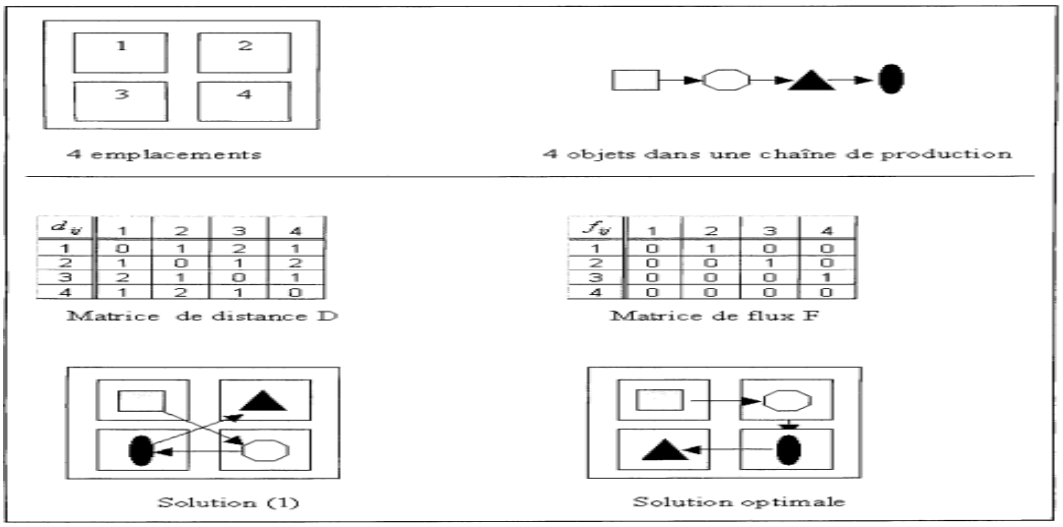

Figure 4.4 : Un exemple d'affectation quadratique

Le coût associé à la solution (1) est :

$$L = 2+1+2 = 5$$

le coût de la solution (2) est :

$$L = 1+1+1 = 3$$

Cela démontre que l'option (2) est le meilleur arrangement et l'une des solutions optimales.

6 Applications principales

Bien que le problème d'affectation quadratique ait de nombreuses applications pratiques en informatique, en productique électronique et en architecture, il est incroyablement difficile à résoudre [39].

6.1 Les applications :

- **Les services hospitaliers :**

Le problème est apparu pour la première fois en 1972 lors de la planification du Klinikum Regensburg, un hôpital universitaire allemand. Le problème Krarup30a nécessite que 30 emplacements soient attribués à 30 équipements [39].

- **Domaine de l'électronique:**

Le problème (PAQ) est utilisé dans le domaine de l'électronique pour optimiser le positionnement et le routage des circuits électroniques [39].Figures 4.5 et 4.6

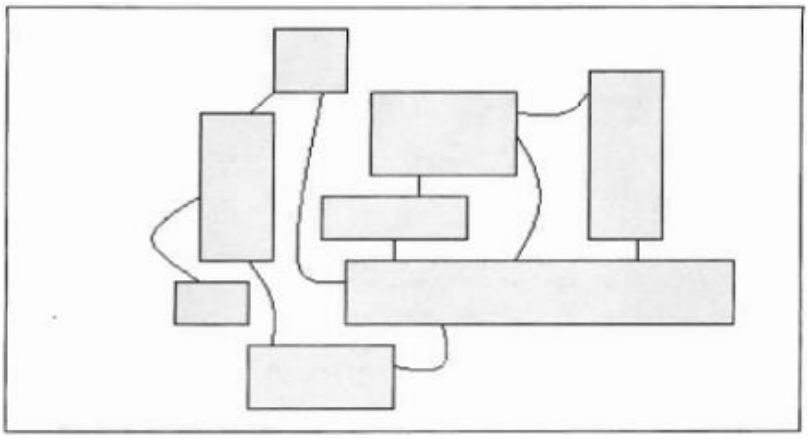

Figure 4.5 : Emplacement des composantes en désordre

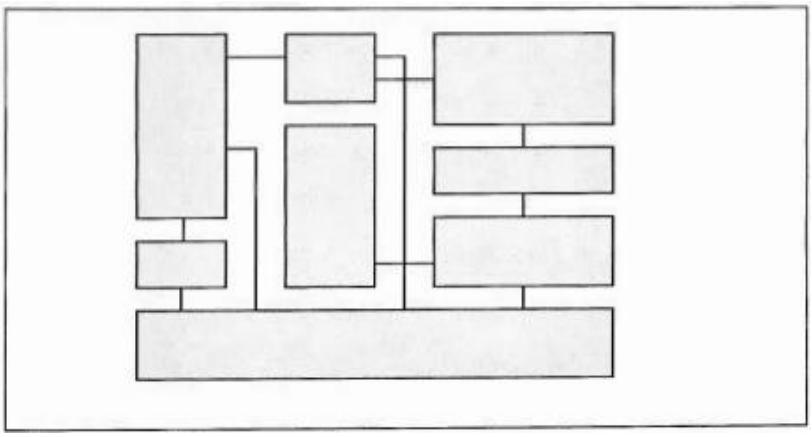

Figure 4.6 : Emplacement des composantes en ordre par le (PAQ)

7 La algorithme FastPFP les grandes Appariement de graphes

Selon [36], Un nouvel algorithme d'appariement de graphes appelé Fast Projected Fixed-Point (FastPFP) a été proposé capable de traiter de grands graphes de plus de 1 000 nœuds. L'algorithme a une complexité temporelle O(n^3) par itération, une complexité spatiale O(n^2) et est capable de faire correspondre des graphes attribués pondérés non orientés de différentes tailles. FastPFP est facile à mettre en œuvre, robuste et à une garantie de convergence linéaire, il a également montré des performances améliorées par rapport aux algorithmes précédents dans la plupart des cas.

7.1 Formulation du problème

L'appariement de graphes consiste à trouver l'appariement optimal entre deux graphes non orientés de taille n et n', représentés par leurs matrices d'adjacence A (binaire pour les graphes non pondérés et à valeur réelle pour les graphes pondérés) et A' (également symétrique) [36]. Ces matrices A, A' ont des tailles de n × n et n' × n',

respectivement, la fonction objective de l'appariement de graphes est défini par

$$\mathrm{Min}\frac{1}{2}\|X^{T}\,\mathrm{AXA}'\|_{f}^{2}+\lambda\|\mathrm{B}-\mathrm{XB}'\|_{f}^{2} \qquad (1)$$

$$s.t. \quad X\mathbf{1} \le \mathbf{1},\ X^{T}\mathbf{1} = \mathbf{1},\ X \in \{0,1\}^{n\times n'} \qquad (2)$$

$\|.\|_F$ est la norme matricielle de Frobenius, λ est un paramètre de contrôle, et 1 est un vecteur dont tous ses éléments sont égaux à un. Les contraintes de (2) garantissent que X est une matrice de permutation partielle. La constante $\frac{1}{2}$ est incluse pour plus de commodité. Sans perte de généralité, nous supposons que n ≥ n'. Résoudre le problème de minimisation exprimé en (1)(2) équivaut à

$$\mathrm{Max}\frac{1}{2}\mathrm{tr}(X^{T}\mathrm{AXA}') + \lambda\mathrm{tr}(X^{T}\,\mathrm{K}) \qquad (3)$$

$$s.t. \quad X\mathbf{1} \le \mathbf{1},\ X^{T}\mathbf{1} = \mathbf{1},\ X \in \{0,1\}^{n\times n'} \qquad (4)$$

où tr représente la trace matricielle et K désigne n×n' la matrice BB'^{T}.Il est généralement admis que ce problème est un problème d'affectation quadratique (QAP) et est en général NP-difficile [36].

7.2 Algorithme

La relaxation continue du problème discret d'origine est le fondement de l'algorithme, FastPFP. L'espace des matrices de permutation partielles, qui est le domaine du problème discret d'origine [36] est :

$$\{\mathrm{X} \mid X\mathbf{1} \le \mathbf{1},\ X^{T}\mathbf{1} = \mathbf{1},\ X \in \{0,1\}^{n\times n'}\},$$

à l'espace matriciel bi-stochastique

$$\{\mathrm{X} \mid X\mathbf{1} \le \mathbf{1},\ X^{T}\mathbf{1} = \mathbf{1},\ X \ge 0\},$$

L'espace relaxé est alors défini pour une technique de point fixe projeté [36]. Nous expliquerons comment l'algorithme a été dérivé dans les sous-sections qui suivent.

7.3 Point fixe projeté rapide

Afin de résoudre le problème original du QAP (3)(4), il a été relaxé son domaine sur l'espace des matrices partielles doublement stochastiques [36].

$$\mathrm{Max}\frac{1}{2}\mathrm{tr}(X^{T}\,\mathrm{AXA}') + \lambda\mathrm{tr}(X^{T}\,\mathrm{K}) \qquad (5)$$

$$s.t. \qquad X\mathbf{1} \le \mathbf{1},\ X^{T}\mathbf{1} = \mathbf{1},\ X \ge 0 \qquad (6)$$

Notez que pour la dérivation de l'algorithme :

$$\mathrm{f(X)} = \frac{1}{2}\,\mathrm{tr}(\mathrm{X}^{T}\mathrm{AXA}') + \lambda\mathrm{tr}(X^{T}\,\mathrm{K}) \qquad (7)$$

Où $\mathrm{X} \in R^{\mathrm{n}\times\mathrm{n}'}$, le gradient de f(X) est

$$\nabla f(X) = AXA' + \lambda K \quad (8)$$

Nous fournissons un nouvel algorithme à virgule fixe projeté qui est décrit comme

$$X^{(t+1)} = (1 - \alpha)X^{(t)} + \alpha Pd(\nabla f(X^{(t)})) \quad (9)$$

$$\text{s.t. } \alpha \in [0, 1], \quad (10)$$

la projection partielle doublement stochastique Pd(), définie par :

$$Pd(X) = \arg\min d \| x - d \|F , \quad (11)$$

$$s.t. \quad \boldsymbol{d1} \leq \mathbf{1}, \boldsymbol{d}^T \mathbf{1} = \mathbf{1}, \boldsymbol{d} \geq 0 \quad (12)$$

L'algorithme Fast Projected Fixed-Point (FastPFP) a une garantie de convergence linéaire et introduit plus de douceur avec une méthode de point fixe projeté proportionnellement mise à jour. A est le paramètre de taille de pas, et pour $\alpha = 1$, l'algorithme est une simple méthode à virgule fixe projetée. Chaque X(t) reste dans l'espace des matrices doublement stochastiques partielles si l'état initial X(0) est une matrice doublement stochastique partielle [36].

$$Z = (1 - \alpha)X + \alpha Y, \text{ s.t. } \alpha \in [0, 1], \quad (13)$$

est encore une autre matrice doublement stochastique partielle.

7.4 Interprétation et travaux connexes

Selon [36], en utilisant la définition de la projection sur l'espace des matrices de permutation partielles dans (9), remplacer Pd() par et laisser $\alpha = 1$ .défini comme

$$P_{perm}(X) = \arg\min P \| X - P \|^2 F , \quad (14)$$

$$s.t. \quad \boldsymbol{P1} \leq \mathbf{1}, \boldsymbol{P}^T \mathbf{1} = \mathbf{1}, \boldsymbol{P} \in \{0, 1\}^{n \times n'} \quad (15)$$

qui peut être résolu par l'approche hongroise. Alors (9) devient essentiellement un IPFP simplifié.

$$X^{(t+1)} = Pperm(\nabla f(X^{(t)})). \quad (16)$$

Chaque étape de l'IPFP entraîne une augmentation de la fonction d'objectif (3) [40]. Par conséquent, FastPFP peut être considéré comme une variante plus douce d'IPFP.

L'algorithme Projected Gradient (PG), d'autre part, est décrit comme

$$X^{(t+1)} = Pd\,(X^{(t)} + \alpha \nabla f(X^{(t)})). \quad (17)$$

L'algorithme Projected Gradient (PG) est algorithmiquement similaire à FastPFP dans le problème d'appariement de graphes, mais il reste facilement bloqué dans des optima locaux médiocres. Il s'exécute également plus lentement que FastPFP. Le taux de convergence de l'algorithme PG pour résoudre le problème d'appariement de graphes est

inconnu malgré sa garantie de convergence [41]. Nous verrons comment obtenir la solution de projection Pd() dans la sous-section suivante.

7.5 Projection doublement stochastique partielle

En introduisant des variables d'écart, il est possible de transformer la projection doublement stochastique partielle en une projection doublement stochastique. La matrice relâchée de **X** peut être représentée par **Y** [36].

$$Y=\begin{pmatrix} X11 & X12 & \dots & X1n' & Y1(n'+1) & \dots & Y1n \\ X21 & X22 & \dots & X2n' & Y2(n'+1) & \dots & Y2n \\ \dots & \dots & & \dots & \dots & & \dots \\ Xn1 & Xn2 & \dots & Xnn' & Yn(n'+1) & \dots & Ynn \end{pmatrix} \quad (18)$$

Selon [36], on écrit Y1:n,1:n' = X, où Y1:n,1:n' désigne la matrice formée par les n premières lignes et les **n'** premières colonnes de Y. la projection d'une matrice Y sur l'espace des matrices doublement stochastiques sous le frobenius la norme est la solution du problème suivant

$$P_D(Y) = \arg\min \|Y - D\|_F , \quad (19)$$

$$s.t. \quad D\mathbf{1} \le \mathbf{1},\ D^T\mathbf{1} = \mathbf{1},\ D \ge 0 \quad (20)$$

Nous pouvons démontrer la convergence linéaire de FastPFP dans la sous-section suivante en utilisant cette projection doublement stochastique. Définissez deux problèmes (projections) de la période (19-20) comme

$$P_1(Y) = \arg\min \|Y - D\|_F, \quad s.t. \quad D\mathbf{1} \le \mathbf{1},\ D^T\mathbf{1} = \mathbf{1} \quad (21)$$

$$P_2(Y) = \arg\min \|Y - D\|_F, \ D \ge 0 \quad (22)$$

Il existe une solution de forme fermée pour P1 et P2.

$$P1(Y) = Y + \left(\frac{1}{n}I + \frac{1^T x 1}{n^2}I - \frac{1}{n}Y\right)11^T - \frac{1}{n}11^T Y \quad (23)$$

$$P2(Y) = \frac{Y + |Y|}{2} . \quad (24)$$

La méthode de projection successive pour dériver une projection partielle double stochastique sous la norme de Fronbenius est expliquée. La solution à la projection partielle doublement stochastique est PD (x) = Pd (y) 1: n, 1: n' [36].

7.6 Analyse de convergence

Selon [36], Nous énonçons maintenant le théorème de convergence FastPFP. Le lemme 1 est la clé de voûte de la preuve du théorème.

Le lemme 1 : Définir $\Omega = \{ X \mid X\mathbf{1} \le \mathbf{1}, X^T\mathbf{1} = \mathbf{1}, X \ge 0\}$.

Pour deux matrices réelles X1 et X2, $\|P_d(X1) - P_d(X2)\|_F \le \|(X1) - (X2)\|_F$

Preuve: Pd() est une projection non expansive puisque est une convexe fermée, ce qui indique que $\|(X1)-(X2)\|_F \geq \|Pd(X1)-Pd(X2)\|_F$.

Théorème 1 : Étant donné la matrice réelle $X^{(0)}$, $\alpha \in [0, 1]$ et $\| A \otimes A' \|_2 < \varepsilon$, la série

$$X^{(t+1)} = (1-\alpha)X^{(t)} + \alpha Pd(AX^{(t)}A' + \lambda K) , \quad (25)$$

converge au taux $1 - \alpha + \alpha\,\varepsilon$.

Preuve: Par x, y ,k et A, notons respectivement vec(X), vec(Y) et $A \otimes A'$, où vec() est la vectorisation d'une matrice et $\otimes$ est le produit de Kronecker. Puis $Y^{(t)} = Ax^{(t)} A' + \lambda K$ est équivalent à $Y^{(t)} = Ax^{(t)} + \lambda K$

Ainsi

$$\|X^{(t+1)} - X^{(t)}\|_F \leq (1-\alpha) \cdot \|X^{(t)} - X^{(t-1)}\|_F \quad (26)$$

$$+ \alpha \cdot \|Pd(AX^{(t)}A' + \lambda K) \quad (27)$$

$$- Pd\,(AX^{(t-1)}A' + \lambda K)\,\|_F . \quad (28)$$

Aussi

$$\|Pd(AX^{(t)}A' + \lambda k) - Pd(AX^{(t-1)}A' + \lambda K)\|_F \quad (29)$$

$$\leq \|AX^{(t)}A' - AX^{(t-1)}A'\|_F = \|AX^{(t)} - AX^{(t-1)}\|_2 \quad (30)$$

$$\leq \|A\|_2 . \|X^{(t)} - X^{(t-1)}\|_2 \quad (31)$$

$$< \varepsilon . \|X^{(t)} - X^{(t-1)}\|_F \quad (32)$$

Ainsi

$$\frac{\|X^{(t-1)} - X^{(t)}\|_F}{\|X^{(t)} - X^{(t-1)}\|_F} < 1 - \alpha + \alpha\,\varepsilon. \quad (33)$$

Remarques : Il n'est pas nécessaire de mettre à l'échelle A et A' en pratique pour que $\| A \otimes A' \|_2$. L'instabilité numérique peut-être évitée en laissant X = X/ max(X) à la fin de chaque itération.

7.7 Discrétisation

Selon [36], suite à la convergence de la méthode du point fixe projeté, la matrice de sortie X est discrétisée pour produire la matrice de permutation partielle P. Au lieu de la coûteuse technique hongroise, nous employons un algorithme glouton. Voici comment fonctionne l'algorithme de discrétisation gloutonne :

Étape 1.Pour commencer, créez une matrice nulle P(n x n'). Après avoir créé la matrice, créez un ensemble L qui contient toutes les paires d'indices (i, j) pour la matrice P.

Étape 2 .soit $P_{i*j*} = 1$ pour obtenir l'indice $(i*, j*)$ de L tel que $X_{i*j*} = \arg\max_{(i,j)\in L} X_{ij}$ étant donné une matrice X. Ensuite, enlevez tous les indices (i, j) dans L où $i=i*$ ou $j=j*$.

Étape 3 .Continuez à effectuer l'étape 2 jusqu'à ce que L soit vide. Retour P.

dans la Figure 4.7 Algorithme 1 résume l'ensemble de l'algorithme. Il est important de noter que n ≥ n' est supposé. Pour garantir la précision d'appariement de FastPFP, nous recommandons d'utiliser l'initialisation $X = 11^T_{n\times n'}/nn'$, $Y=0_{n\times n}$, comme nous l'avons fait dans toutes nos expériences. L'étape 9 devient nécessaire pour la stabilité numérique. Pour n = n', quel que soit le calcul de matrice rapide et clairsemé, l'étape 3 de l'algorithme 1 nécessite $O(n^3)$ opérations par itération. Les étapes 4 à 7 nécessitent $O(n^2)$ opérations par itération, tandis que l'étape 8 nécessite $O(n^2)$ opérations par itération. L'algorithme a une complexité temporelle de $O(n^3)$ par itération et une complexité spatiale de $O(n^2)$ [36].

Algorithm 1: Fast Projected Fixed-Point

Input: A, A′,K

Output: P

1 Initialize X and Y

2 **repeat**

3 $Y_{1:n,1:n'} = AX A' + \lambda K$

4 **repeat**

5 $Y = Y + (\frac{1}{n}I + \frac{1^T Y_1}{n2}I - \frac{1}{n}Y)11^T - \frac{1}{n}11^T Y$

6 $Y=\frac{Y+|Y|}{2}$

7 **until** Y converges;

8 $X = (1 - \alpha)X + \alpha Y_{1:n,1:n'}$

9 X = X/ max(X)

10 **until** X converges;

11 Discretize X to obtain P.

Figure 4.7 : Algorithm Fast Projected Fixed-Point

8 Présentation de l'application :

L'interface utilisateur principale de l'application est illustrée à la figure 4.8. Les graphes peuvent être chargés, affichés et appariés à l'aide des boutons et des menus de cette interface.

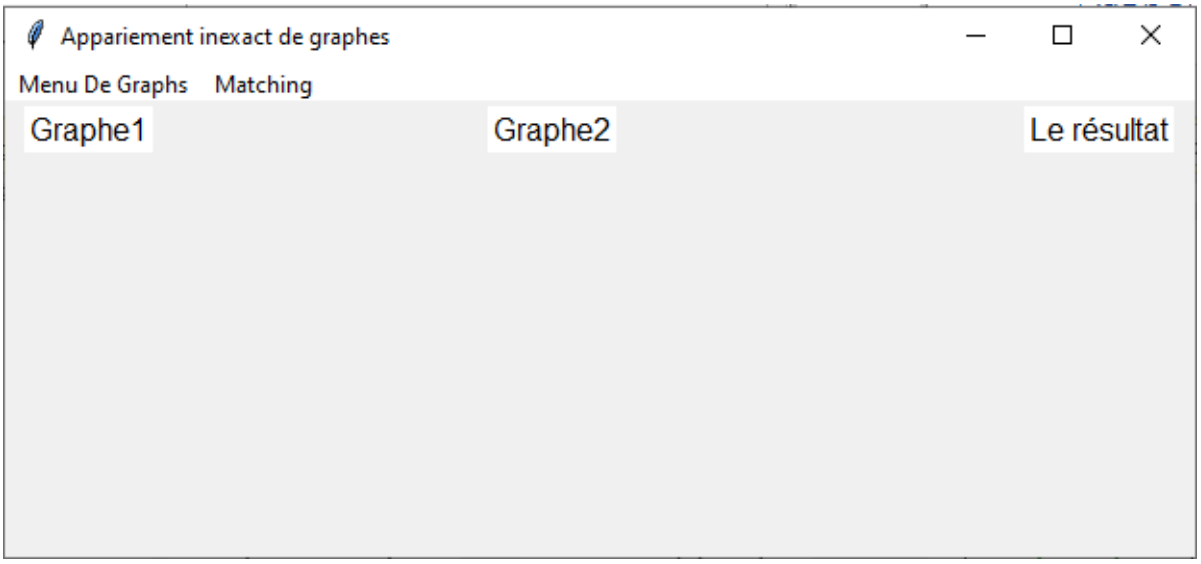

Figure 4.8 : L'interface principale de l'application

8.1 Des exemples

Exemple 1 :

dans le graphe 1 nous avons placé le graphe G1, qui a 4 sommets, avec la matrice adjacente représentée comme suit :

$$G1=\begin{pmatrix}0 & 1 & 3 & 6\\ 1 & 0 & 0 & 0\\ 3 & 0 & 0 & 5\\ 6 & 0 & 5 & 0\end{pmatrix}$$

dans le graphe 2 nous avons placé le graphe H1, qui a 3 sommets, avec la matrice adjacente représentée comme suit :

$$H1=\begin{pmatrix}0 & 6 & 3\\ 6 & 0 & 5\\ 3 & 5 & 0\end{pmatrix}$$

La figure 4.9 montre le résultat des graphes d'appariement G1 et H1

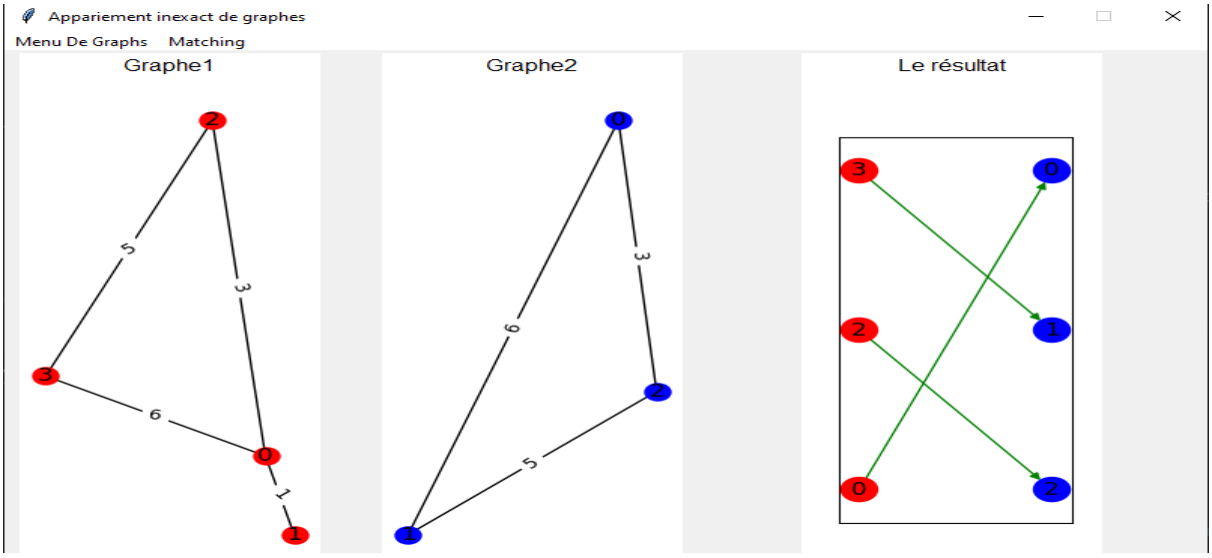

Figure 4.9 : l'appariement des graphes G1 et H1

Exemple 2 :

La figure 4.10 montre le résultat de l'appariement des graphes G2 et H2 où G2 est placé dans le graphe1 et H2 est dans le graphe2 . Ils ont le même nombre de sommets . les matrices adjacentes de G1 et de H1 représentée comme suivent :

$$G2=\begin{pmatrix} 0 & 80 & 30 & 50 \\ 80 & 0 & 0 & 0 \\ 30 & 0 & 0 & 30 \\ 50 & 0 & 30 & 0 \end{pmatrix} \quad ET \quad H2=\begin{pmatrix} 0 & 30 & 50 & 0 \\ 30 & 0 & 30 & 0 \\ 50 & 30 & 0 & 80 \\ 0 & 0 & 80 & 0 \end{pmatrix}$$

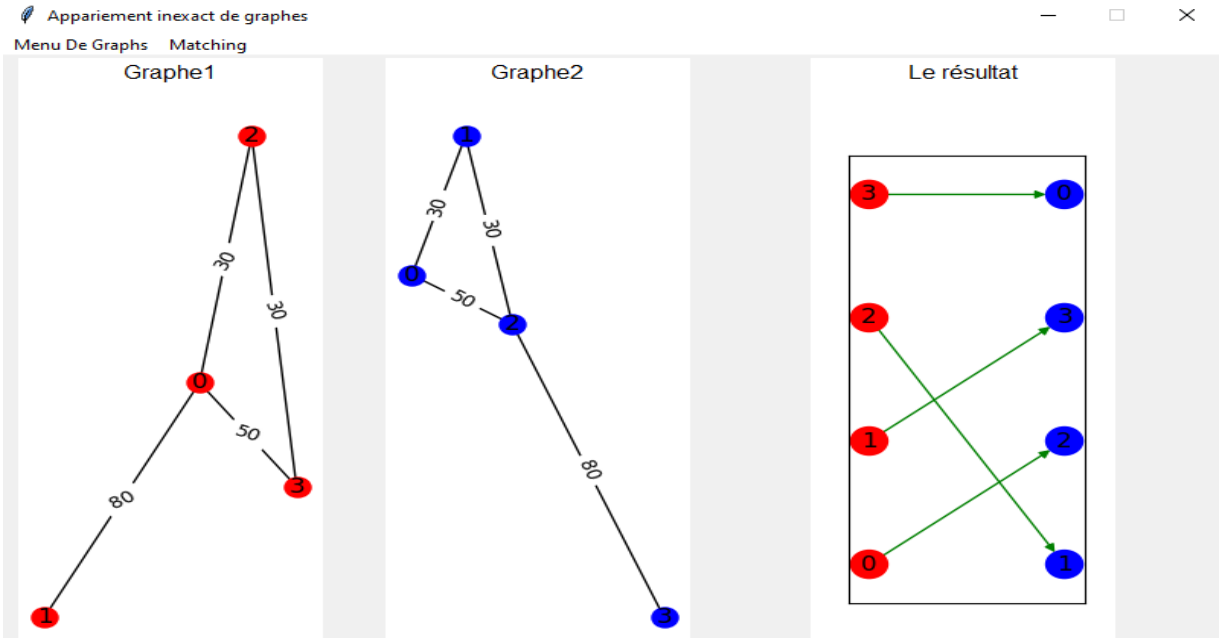

Figure 4.10 : l'appariement des graphes G2 et H2

Exemple 3 :

Le résultat de l'appariement des graphes GH et G2 est illustré à la Figure 4.11, où GH est situé dans le graphe 1 et a 5 sommets, G2 est situé dans le graphe 2 et a 4 sommets, et les matrices adjacentes de GH et G2 sont représentées comme suit :

$$GH=\begin{pmatrix} 0 & 30 & 30 & 30 & 0 \\ 30 & 0 & 40 & 0 & 0 \\ 30 & 40 & 0 & 30 & 50 \\ 30 & 0 & 30 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 \end{pmatrix} \quad ET \quad G2=\begin{pmatrix} 0 & 80 & 30 & 50 \\ 80 & 0 & 0 & 0 \\ 30 & 0 & 0 & 30 \\ 50 & 0 & 30 & 0 \end{pmatrix}$$

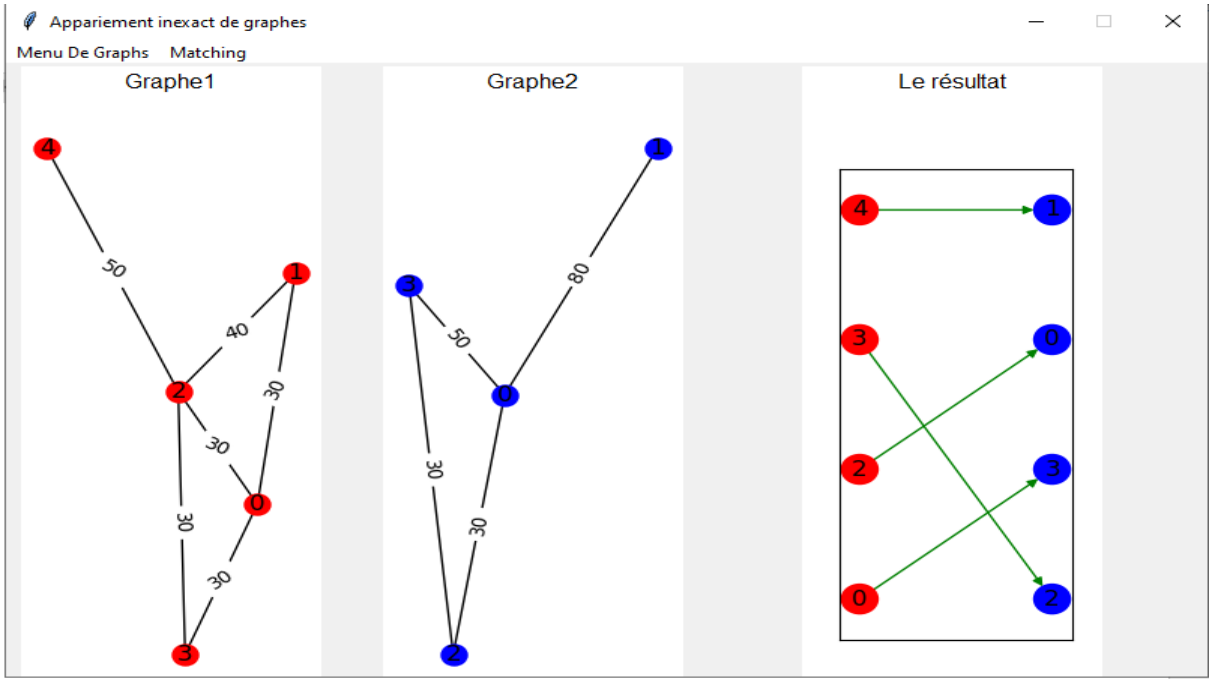

Figure 4.11 : l'appariement des graphes GH et G2

9 Conclusion:

Afin de faire appariement inexact de graphes, nous avons utilisé dans ce chapitre l'algorithme Fast Projected Fixed-Point (FastPFP).Cet algorithme est utilisé pour l'appariement de graphes basé sur une nouvelle méthode à virgule fixe et est bien adapté à l'appariement de grands graphes. Cet algorithme repose sur des techniques de comparaison de sous structures de graphes, telles que la comparaison de sous-graphes, la comparaison de modèles, la comparaison de noyaux, etc. Cet algorithme peut être appliqué pour résoudre des problèmes tels que la reconnaissance d'objets dans des images et la détection de similarités.

Conclusion Générale

Conclusion Générale:

Les résultats d'une correspondance de graphe inexacte peuvent varier en fonction de l'algorithme ou de l'approche spécifique utilisée. Certaines techniques courantes pour la mise en correspondance de graphes inexactes incluent la distance d'édition de graphes, les noyaux de graphes et les méthodes spectrales. Ces méthodes visent à quantifier la similarité entre les graphes et à trouver la meilleure correspondance possible entre leurs nœuds et leurs arêtes.

L'algorithme Fast Projected Fixed-Point (FastPFP) a une complexité temporelle O($n3$) par itération, une complexité spatiale O($n2$) et est capable de faire correspondre des graphes attribués pondérés non orientés de différentes tailles. FastPFP est facile à mettre en œuvre, robuste et à une garantie de convergence linéaire.

Bibliographie

[1] R. J. QURESHI, Artist, Reconnaissance de formes et symboles graphiques complexes. [Art]. University of Tours, le 4 mars 2008.

[2] V. Carletti, Artist, Exact and Inexact Methods for Graph Similarity in Structural Pattern Recognition. [Art]. univ de Caen, 2016.

[3] Basic, «GFG Weekly Coding Contest,» Sovereign Corporate Tower, 4 jul 2021. [En ligne]. Available: https://www.geeksforgeeks.org/graph-homomorphism/. [Accès le 2 février 2023].

[4] T. Bärecke, Artist, Isomorphisme Inexact de Graphes par Optimisation Évolutionnaire. [Art]. Université Pierre et Marie Curie - Paris , 2009.

[5] Salim Jouili Indexation de masses de documents graphiques : approches structurelles Université Nancy France 2011.

[6] J. Lebrun, Artist, Appariement inexact de graphes appliqué à la recherche d'image et d'objet 3D. [Art]. Université de Cergy Pontoise, 2011.

[7] F. a. M. V. CONTE, Artist, THIRTY YEARS OF GRAPH MATCHING. [Art]. Universit`a di Napoli "Federico II", 2004.

[8] p. fehér, Artist, Methods for Improving and Applying Graph Rewriting-Based Model Transformations. [Art]. Budapest University of Technology and Economics Department of Automation and Applied Informatics, 2017.

[9] P. Madarasi, Artist, Algorithms for Matching. [Art]. Eotvos Lorand University, 2016.

[10] Z. Abu-Aisheh, Artist, Approches anytime et distribu´ees pour l'appariment de graphes Anytime and Distributed Approaches for Graph Matching. [Art]. universite francois rabelais de tours, 2016.

[11] S. P. Dwivedi, «Inexact Graph Matching Using Centrality Measures,» Social and Information Networks (cs.SI, 21 dec 2021. [En ligne]. Available: https://arxiv.org/abs/2201.04563v1. [Accès le 23 02 2023].

[12] K. Madi, Artist, Inexact graph matching : application to 2D and 3D. [Art]. Université de Lyon, 2016.

[13] F. a. M. A. a. M. M. Conti, «Conti F, Minelli A, Melone M. GABA transporters in the mammalian cerebral cortex: localization, development, pathological implications. Brain Res Brain Res Rev 45: 196-212,» Brain research. Brain research reviews, vol. 45, n° %110.1016/j.brainresrev.2004.03.003, pp. 196-212, 2004.

[14] S. F. &. H. B. R. Ambauen, «Graph Edit Distance with Node Splitting and Merging, and Its Application to Diatom Identification,» the Lecture Notes in Computer Science , vol. LNCS, n° %12726, p. pp 95–106, 2003.

[15] A. S. D. M. R. H. David Knossow, «Inexact Matching of Large and Sparse Graphs Using,» 7th International Workshop on Graph-Based Representations, Vols. %1 sur %2ff10.1007/978-3-642-02124-4_15f, n° %100446989f, pp. pp.144-153, May 2009.

[16] R. SOLOH, «Graphs and Binary Linear Programming for Natural Object Modeling,» l'Université Le Havre Normandie, Liban, 2022.

[17] J.-E. H. S. PHILIPP-FOLIGUET1, «Recherche d'objets par appariement de graphes combinant contours et régions,» 6avenue du Ponceau, vol. f95014, n° %144, pp. 637-644, January 2010.

[18] J.-B. F. H. M. I. B. R. D. Jérémy Chopin, Artist, Méthode d'analyse sémantique d'images combinant. [Art]. Rencontres des Jeunes Chercheur×ses en Intelligence Artificielle, Jun 2020.

[19] S. Jain, «GeeksforGeeks,» A-143, 9th Floor, Sovereign Corporate Tower,, 0 0 2009. [En ligne]. Available: https://www.geeksforgeeks.org/ . [Accès le 25 02 2023].

[20] S. J. COMPUT, «LOCAL SEARCH HEURISTICS FOR k-MEDIAN AND,» Society for Industrial and Applied Mathematics, vol. 33, n° %13, p. 544–562, 2004.

[21] P. K. Jain, Écrivain, Some Algorithms on Exact,Approximate and Error-Tolerant Graph. [Performance]. Banaras Hindu University (BHU), 2019.

[22] R. LOUCIF, «Parallélisation d'Algorithmes d'Optimisation Combinatoire,» Université colonel HADJ LAKHDAR –BATNA, Algérie –BATNA, 2014

[23] N. Gastineau, «Partitionnement, recouvrement et colorabilité dans les graphes,» Universit´e de Bourgogne, France, 2014.

[24] M. Rigo, «Théorie des graphes,» Université de Liège Faculté des sciences , 2009–2010.

[25] A. Gournay, «Théorie des graphes,» Institut de Mathématiques, Université de Neuchâtel Suisse, Septembre 2013.

[26] J. B. e. U. Murty, «Théorie des graphes,» Traduit de l'anglais par F. Hayet, 2008.

[27] N. Saadia. B. Ghezala, «Etat de l'art des algorithmes d'appariement des graphes,» Université Dr. Tahar Moulay , SAIDA, 2017 / 2018.

[28] Javatpoint, «Javatpoint,» India, 2011. [En ligne]. Available: https://www.javatpoint.com/graph-theory-graph-representations. [Accès le 20 03 2023].

[29] Hopcroft J. E. et Wong J. K., Linear Time Algorithm for Isomorphism of Planar Graphs, In Sixth ACM Symposium on Theory of Computing, 1974.

[30] Abdulkader A. M., Parallel Algorithms for Labelled Graph Matching, PhD thesis, Colorado School of Mines, 1998.

[31] H. Bunke. On a relation between graph edit distance and maximum common sub-

graph. Pattern Recognition Letters, 1997.

[32] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters, 1998.

[33] W. D.Wallis, P. Shoubridge, M. Kraetzl, and D. Ray. Graph distances using graph union. Pattern Recognition Letters, 2001.

[34] M.-L. Fernandez and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recognition Letters, 2001.

[35] Salim Jouili, Indexation de masses de documents graphiques : approches structurelles - Université Nancy 2, Mars 2011.

[36] K. H. a. C.-L. L. Yao Lu, *A Fast Projected Fixed-Point Algorithm for Large,* Chines: Chinese Academy .

[37] B. McKay. Practical graph isomorphism. Congressus Numerantium, 30(30) :47– 87, 1981.

[38] R. F. Tanjona, «RESOLUTION DU PROBLEME D'AFFECTATION PAR LA METHODE DU SIMPLEXE,» UNIVERSITÉ D'ANTANANARIVO, Madagascar, 2021.

[39] A. KAMIL, «Application d'un algorithme hybride à colonie de fourmis au problème d'affectation quadratique,» L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI, ABITIBI-TÉMISCAMINGUE, 2008.

[40] M. Leordeanu, M. Hebert, and R. Sukthankar, «An Integer Projected Fixed Point Method for Graph Matching and MAP Inference, » Proc. Advances in Neural Information Processing Systems, pp. 1114-1122, 2009.

[41] P.H. Calamai, «Projected Gradient Methods for Linearly Constrained Problems, » Mathematical Programming, vol. 39, no. 1, Oct 1987.