République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

ESI ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
Ecole nationale Supérieure d'Informatique
ex. INI (Institut National de formation en Informatique)

Mémoire

Pour Obtention du diplôme de Master En Informatique

Option : Système Informatique (SIQ)

Compression de Graphes par extraction de motifs et k2-trees : étude et implémentation

---

Réaliser par :

Mlle. Hafsa Bousbiat
eh_bousbiat@esi.dz
ESI

Mlle. Sana Ihadadene
es_ihadadene@esi.dz
ESI

Encadreurs :

Dr. Karima Amrouche
k_amrouche@esi.dz
ESI

Dr. Hamida Seba
hamida.seba@univ-lyon1.fr
Université de Lyon

Dr. Mohammed Haddad
mail
Université de Lyon

Octobre 2018

Année Universitaire : 2018-2019

Remerciement

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nostrum tempore ea fugiat numquam autem saepe quas porro vitae ? Fugit commodi tempore voluptate sint fugiat, possimus optio ad ! Pariatur, obcaecati quidem. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Neque excepturi ducimus accusantium eius voluptatibus, quod velit, explicabo tenetur aliquid ipsam sapiente. Quibusdam quis ullam, saepe numquam molestias nobis recusandae labore ? Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nostrum tempore ea fugiat numquam autem saepe quas porro vitae ? Fugit commodi tempore voluptate sint fugiat, possimus optio ad ! Pariatur, obcaecati quidem. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Neque excepturi ducimus accusantium eius voluptatibus, quod velit, explicabo tenetur aliquid ipsam sapiente. Quibusdam quis ullam, saepe numquam molestias nobis recusandae labore ? Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nostrum tempore ea fugiat numquam autem saepe quas porro vitae ? Fugit commodi tempore voluptate sint fugiat, possimus optio ad ! Pariatur, obcaecati quidem. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Neque excepturi ducimus accusantium eius voluptatibus, quod velit, explicabo tenetur aliquid ipsam sapiente. Quibusdam quis ullam, saepe numquam molestias nobis recusandae labore ?

Résumé

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nostrum tempore ea fugiat numquam autem saepe quas porro vitae ? Fugit commodi tempore voluptate sint fugiat, possimus optio ad ! Pariatur, obcaecati quidem. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Neque excepturi ducimus accusantium eius voluptatibus, quod velit, explicabo tenetur aliquid ipsam sapiente. Quibusdam quis ullam, saepe numquam molestias nobis recusandae labore ? Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nostrum tempore ea fugiat numquam autem saepe quas porro vitae ? Fugit commodi tempore voluptate sint fugiat, possimus optio ad ! Pariatur, obcaecati quidem. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Neque exceptu

Abstract

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nostrum tempore ea fugiat numquam autem saepe quas porro vitae ? Fugit commodi tempore voluptate sint fugiat, possimus optio ad ! Pariatur, obcaecati quidem. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Neque excepturi ducimus accusantium eius voluptatibus, quod velit, explicabo tenetur aliquid ipsam sapiente. Quibusdam quis ullam, saepe numquam molestias nobis recusandae labore ? Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nostrum tempore ea fugiat numquam autem saepe quas porro vitae ? Fugit commodi tempore voluptate sint fugiat, possimus optio ad ! Pariatur, obcaecati quidem. Lorem ipsum dolor, sit amet consectetur adipisicing elit. Neque exceptu

Table des matières

Table des figures

Liste des tableaux

Première partie

Introduction

Chapitre 1

Théorie des graphes

Pour faciliter la compréhension d'un problème, nous avons tendance à le dessiner ce qui nous amène parfois même à le résoudre, la théorie des graphes est fondée, à l'origine sur ce principe, de nombreuses propriétés et méthodes ont été pensées ou trouvées à partir d'une représentation schématique pour être ensuite formalisées et prouvées.
La théorie des graphes est historiquement un domaine mathématique qui s'est développé au cours des années au sein des autres disciplines comme la chimie, la biologie, la sociologie et l'industrie. Elle constitue aujourd'hui un corpus de connaissance très important et un instrument efficace pour résoudre une multitude de problèmes.
De manière général le graphe sert à représenter les structures, les connexions entre différents composants, les acheminements possible pour un ensemble complexe composé d'un grand nombre de situations, en exprimant les dépendances et les relations entre ses éléments,(e.g. réseau routier ou ferroviaire, réseau de communication,diagramme d'ordonnancement, ..).
Dans ce chapitre nous introduisons les définitions et notions de base relatives aux graphes que nous utiliserons par la suite.

1.1 Graphe non orienté

1.1.1 Définitions et généralités

Un graph non orienté G est la donnée d'un couple (V , E) où V = $\{v_1, v_2, ..., v_n\}$ est un ensemble fini dont les éléments sont appelés sommets ou nœuds (Vertices en anglais) et E=$\{ e_1, e_2, \ldots, e_m \}$ est un ensemble fini d'arêtes (Edges en anglais). Toute arête e de E correspond à un couple non ordonné de sommets $\{ v_i, v_j \} \in E \subset V \times V$ représentants ses extrémités (Müller, 2012) (Fages, 2014).
Soient $e = (v_i, v_j)$ et $e'=(v_k, v_l)$ deux arêtes de E, On dit que :

— v_i et v_j sont les extrémités de e et e est incident en v_i et en v_j (Hennecart et al., 2012).

— v_i et v_j sont voisins ou adjacents, car il y'a au moins une arête entre eux dans E (IUT, 2012).

— L'ensemble des sommets adjacents au sommet e est appelé le voisinage de e (Müller, 2012).

— e et e' sont voisins si ils ont une extrémité commune , i.e. : $v_i = v_k$ par exemple (Lopez, 2003).

— L'arête e est une boucle si ses extrémités coïncident, i.e. : $v_i = v_j$ (IUT, 2012).

— L'arête e est multiple si elle a plus d'une seule occurrence dans l'ensemble E.

1.1.2 Représentation graphique

Un graph non orienté G peut être représenter par un dessin sur un plan comme suit (Müller, 2012) :

— Les nœuds de G : $v_i \in$ V sont représenter par des points distincts.

— Les arêtes de G : $e = (v_i, v_i) \in$ E sont représenter par des lignes pas forcement rectilignes qui relient les extrémités de chaque arête e.

Exemple : Soit g=(V1 , E1) un graphe non orienté tel que : V1={ 1,2,3,5 } et E={ (1,2), (1,4), (2,2), (2,3) ,(2,5) ,(3,4) }. La représentation graphique de g est alors donnée

par le schéma de la figure 1.1.

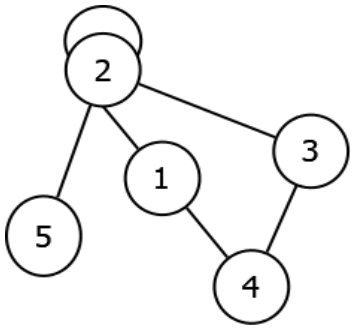

FIGURE 1.1 – Exemple de représentation graphique d'un graphe non orienté

1.1.3 Propriété d'un graphe

Ordre d'un graphe : On appel ordre d'un graphe le nombre de ses sommets.i.e. Card(V) (Roux, 2014).

Taille d'un graphe : On appel taille d'un graphe le nombre de ses arêtes.i.e. Card(E) (Roux, 2014).

Degré d'un graphe :

Degré d'un sommet : Le degré d'un sommet noté $d(v_i)$ est le nombre d'arêtes incidents a ce sommet, sachant qu'une boucle compte pour 2 (Müller, 2012) . Dans l'exemple de la figure 1.1, le degré du sommet (1) est : $d(1)$=2.

Degré d'un graphe : Le degré d'un graphe est le degré maximum de ses sommets. e.i. c'est max($d(v_i)$) (Müller, 2012). Dans l'exemple de la figure 1.1, le degré du graphe est $d(2)$=5.

Rayon et diamétre d'un graphe :

Distance : La distance entre deux sommets v et u est le plus petit nombre d'arêtes qu'on doit parcourir pour aller de v a u ou de u a v (Müller, 2012).

Diamètre d'un graphe : C'est la plus grande distance entre deux sommets de ce graphe (Müller, 2012).

Rayon d'un graph : C'est la plus pette distance entre deux sommets de ce graphe (Parlebas, 1972).

1.2 Graphe orienté

1.2.1 Définitions et généralités

Un graphe orienté G est la donnée d'un couple (V , E) où V est un ensemble fini dont les éléments sont appelés les sommets de G et E $\subset$ V x V est un ensemble de couples ordonnés de sommets dits arcs ou arêtes (Müller, 2012). G est appelé dans ce cas digraphe (directed graphe).

Pour tout arc e = (v_i , v_j) $\in$ E :

— v_i est dit extrémité initiale ou origine de e et v_j est l'extrémité finale de e (Müller, 2012).

— v_i est le prédécesseur de v_j et v_j est le successeur de v_i (IUT, 2012).

— les sommets v_i , v_j sont des sommets adjacents (Jean-Charles Régin, 2016).

— e est dit sortant en v_i et incident en v_j (Jean-Charles Régin, 2016).

— e est appelé boucle si $v_i = v_j$, i.e l'extrémité initiale et finale représente le même sommet (IUT, 2012).

1.2.2 Représentation graphique

Un graphe G = (V , E) peut être projeter sur le plan en représentant :

— dans un premier temps les nœuds $v_i \in$ V par des points disjoints du plan.

— et dans un second temps les arêtes e = (v_i , v_j) $\in$ E par des lignes orientées reliant par des flèches les deux extrémités de e.

Exemple :

Soit g = (V_1 , E_1) un digraphe tel que : V_1 = { 1,2,3,4 } et E_1 = {(1,2),(1,3),(3,2),(3,4),(4,3)}.

Le représentation graphique de g est alors donnée par le schéma de la figure ci-dessous.

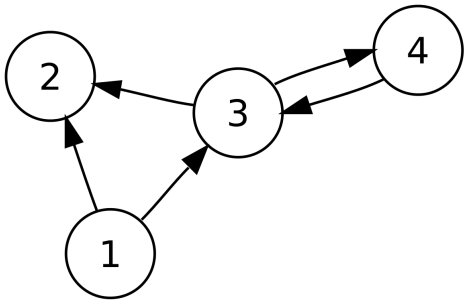

FIGURE 1.2 – Exemple de représentation graphique d'un digraphe.

1.2.3 Quelques Propriétés :

Ordre d'un digraphe : est le nombre de sommets n = Card(V) (Roux, 2014).

taille d'un digraphe : est le nombre d'arcs m = Card(A) (Roux, 2014).

Degré dans un digraphe :

Le degré d'un sommet $v_i \in$ V dans un digraphe G = (V , E) est donnée par la formule :

$$\mathrm{d}(v_i) = d^+(v_i) + d^-(v_i$$

où $d^+(v_i)$ est le nombre d'arcs sortants au sommet v_i et est appelé degré extérieure et $d^-(v_i)$ représente le nombre d'arcs incidents et est appelé degré intérieur (Müller, 2012).

Voisinage dans un digraphe :

Le voisinage d'un sommet $v_i \in$ V, noté V(v_i), dans un digraphe G = (V , E) est :

$$\mathrm{V}(v_i) = \mathrm{succ}(v_i) \bigcup \mathrm{pred}(v_i),$$

avec succ(v_i) qui est l'ensemble des successeurs de v_i et pred(v_i) qui l'ensemble de ses prédécesseurs (Rigo, 2010), i.e le voisinage de v_i est l'ensemble des sommets qui lui sont adjacents.

1.3 Quelques types de graphe

Avec les avancées technologique au fil du temps, plusieurs types de graphes ont connus le jours. En effet, La complexité et la variété des problèmes scientifiques existants modélisés par ces derniers ont poussé les chercheurs à adapter leurs structure selon le problème auquel ils font face. Durant cette section nous allons définir les principaux types existants.

- **Graphe Complet :** Un graphe G = (V , E) est un graphe complet si tous les sommets $v_i \in$ V sont adjacents (Jean-Charles Régin, 2016). Il est souvent noté K_n où n = card(V) (Roux, 2014).
- **Graphe étiqueté et graphe pondéré :** Un graphe étiqueté G = (V , E , W) est un graphe, qui peut être orienté ou non orienté, dont chacune des arêtes $e_i \in$ E est doté d'une étiquette w_i. Si de plus, w_i est un nombre alors G est dit graphe pondéré (valué) (Roux, 2014).
- **Graphe simple et graphe multiple :** Un graphe G = (V , E) est dit simple si il ne contient pas de boucles et tout pair de sommet est reliée par au plus une arête. Dans le cas contraire, G est dit multiple (IUT, 2012).
- **Graphe connexe :** Un graphe non orienté (resp. orienté) est dit connexe (resp. fortement connexe) si pour tout pair de sommets (v_i, v_j) il existe un chemin S les reliant (Müller, 2012).

1.4 Graphe partiel et sous graphe :

La quantité de donnée disponible aujourd'hui et sa croissance de manière exponentiel ont favorisé la décomposition des graphes en des entités plus petites afin de garantir une facilité de compréhension et d'analyse dans le but d'extraire l'information la plus pertinente. Dans cette partie nous allons définir de manière plus formelle ce que ces entités sont ainsi que leurs types.

1.4.1 Définitions :

Soient G = (V , E), G' = (V' , E') et G" = (V" , E") trois graphes.

- Le graphe G' est appelé graphe partiel de G si : V' = V et E' $\subset$ E (Roux, 2014). En d'autres termes, un graphe partiel est obtenu en supprimant une ou plusieurs arêtes de G.
- Le graphe G" est dit sous-graphe de G si : V" $\subset$ V et E"$\subset$ E $\cap$ (V" x V") (Rigo, 2010), i.e un graphe partiel est obtenu en enlevant un ou plusieurs nœuds du graphe initial ainsi que les arêtes dont ils représentent l'une des deux extrémités.

1.4.2 Quelques Types de sous graphes :

- **Une Clique :** est un sous graphe complet de G (Rigo, 2010).
- **Bipartie :** G' est un sous graphe bipartie si : V' = $V_1 \cup V_2$, tel que $V_1 \cap V_2 = \phi$, et E' = V_1 x V_2 (Rigo, 2010).
- **Étoile :** est un cas particulier de sous graphe bipartie ou X est un ensemble contenant le sommet central uniquement et Y contient le reste des nœuds (Koutra et al., 2015) .
- **Chemin (resp. Chaine) :** est une liste de sommets S= $(v_0, v_1, v_2, ..., v_k)$ telle qu'il existe un arc (resp. une arête) entre chaque couple de sommets successifs.
- **Cycle (resp. Circuit) :** est un chemin (resp. chaine) dont le premier et le dernier sommet sont identiques (Roux, 2014).

1.5 Représentation Structurelle d'un graphe

Bien que la représentation graphique soit un moyen pratique pour définir un graphe, elle n'est clairement pas adaptée ni au stockage du graphe dans une mémoire, ni à son traitement. Pour cela plusieurs structures de donnés ont été utilisés pour représenter un graphe, ces structures varient selon l'usage du graphe et la nature des traitements appliquer. Nous allons présenter dans cette partie les structures les plus utilisés.

Soit un graphe G(V,E) d'ordre n et de taille m dont les sommets v_1, v_2,..., v_n et les arêtes (ou arcs) v_1, v_2,..., v_m sont ordonnés de 1 à n et de 1 à m respectivement.

1.5.1 Matrice d'adjacence

La matrice d'adjacence de G est une matrice booléenne carré d'ordre n : $(m_{ij})_{(i,j)\in[0;n]^2}$, dont les lignes (i) et les colons (j) représentent les sommets de G, où les entrés (ij) prenent une valeur de "1" s'il existe un arc (une aréte dans le cas d'un graph non orienté) allant du sommet i au sommet j et un "0" sinon, e.i (Lehman et al., 2010) (SABLIK, 2018) (IUT, 2012) :

$$m_{ij} := \begin{cases} 1 & si \quad (v_i, v_i) \in E \\ 0 & sinon \end{cases}$$

Dans le cas d'un graphe non orienté, la matrice est symétrique par rapport a la diagonale descendante de gauche à droite . e.i. $m_{ij} = m_{ji}$, dans ce cas le graphe peut être représenter avec la composante triangulaire supérieure de la matrice d'adjacence (Müller, 2012).

Note :

- Cette représentation est valide pour le cas d'un graphe non orienté et orienté.
- Dans le cas d(un graphe pondéré, les "1" sont remplacés par les poids des arêtes (ou arcs) (Lopez, 2003).
- Ce mode de représentation engendre des matrices très creuses (comprenant beaucoup de zero) (Hennecart et al., 2012).

Exemple : La figure 1.4 représente un exemple de matrice d'adjacence pour le graphe G ci-contre (figure 1.3) :

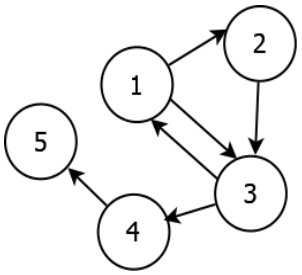

FIGURE 1.3 – Graphe orientée G

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

FIGURE 1.4 – Matrice d'adjacence du graphe G

Place occupé en mémoire : n^2 pour un graphe d'ordre n (Lopez, 2003).

1.5.2 Matrice d'incidence

La matrice d'incidence d'un graphe orienté G est une matrice de taille $n \times m$, dont les lignes représentent les sommets (i ∈ V) et les colons représentent les arcs (j ∈ E) et dont les coefficients (m_{ij}) sont dans {-1, 0, 1}, tel que (Hennecart et al., 2012) (SABLIK, 2018) :

$$m_{ij} := \begin{cases} 1 & si \text{ le sommet i est l'extrémité final de l'arc j} \\ -1 & si \text{ le sommets i est l'extrémité initial de l'arc j} \\ 0 & sinon \end{cases}$$

Pour un graphe non orienté, la coefficients (m_{ij}) de la matrice sont dans {0, 1}, tel que (Hennecart et al., 2012) :

$$m_{ij} := \begin{cases} 1 & si \text{ le sommet i est une extrémité de l'arête j} \\ 0 & sinon \end{cases}$$

Exemple : La figure 1.6 représente un exemple de matrice d'incidence pour le graphe G ci-contre (figure 1.5) :

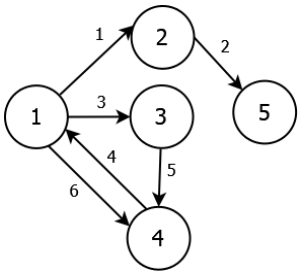

FIGURE 1.5 – Graphe orientée G

$$M = \begin{pmatrix} -1 & 0 & -1 & 1 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

FIGURE 1.6 – Matrice d'incidence du graphe G

Place occupé en mémoire : $n \times m$

1.5.3 Liste d'adjacence

La liste d'adjacence d'un graphe G est un tableau de n listes, où chaque entrée (i) du tableau correspond a un sommet et comporte la liste T[i] des successeurs (ou prédécesseur) de ce sommet, c'est à dire tous les sommets j tel que (i,j) ∈ E (SABLIK, 2018).

Dans le cas d'un graphe non orienté on aura : j ∈ la liste T[i] $\iff$ i ∈ la liste T[j] (IUT, 2012).

Exemple : La figure 1.8 représente un exemple de matrice d'incidence pour le graphe G ci-contre (figure 1.7) :

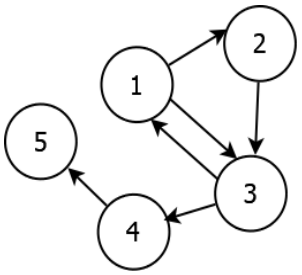

FIGURE 1.7 – Graphe orientée G

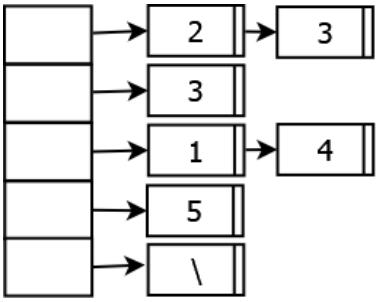

FIGURE 1.8 – Liste d'adjacence du graphe G

Place occupé en mémoire : Dans le cas orienté l'espace est de n + m. Dans le cas

non orienté, l'espace est de n + 2m car une arête est représenté deux fois.

1.6 Les domaines d'application

La diversité des domaines faisant appel à la modélisation par des graphes ne cesse d'augmenter, allant des réseaux sociaux aux réseaux électriques et réseaux biologiques et arrivant jusqu'aux World Wide Web. Dans cette partie nous allons décrire trois domaines d'application les plus répandus des graphes.

1.6.1 Graphes des réseaux sociaux :

Les réseaux sociaux représentent un lieu d'échange et de rencontre entre individus (entités) et dont l'utilisation est devenue de nos jours une nécessité. Pour représenter les interactions entre ces individus, nous avons généralement besoin de faire recours aux graphes où les sommets sont des individus ou des entités et les interactions entre eux sont représenté par des liens. Vue la diversité des interactions sociales, la modélisation de ces réseaux nécessite différents types de graphes : graphes non orientés pour pour les réseaux sociaux avec des relations non orientées, graphes orientés pour représenter des relations non symétriques comme c'est la cas dans les réseaux de confiance, graphes pondérés pour les réseaux sociaux qui contiennent différents niveaux d'intensités dans les relations, ... etc (Lemmouchi, 2012).

1.6.2 Graphes en Bioinformatique :

La bio-informatique est un domaine qui se trouve à l'intersection des deux grands domaines celui de l'informatique et celui de la biologie. Elle a pour but d'exploiter la puissance de calcule des équipements informatiques pour effectuer des traitements sur des données moléculaires massives (Pellegrini et al., 2004).

Elle est largement utilisée pour l'analyse des séquences d'ADN et des protéines à travers leurs modélisation sous forme de graphe. A titre d'exemple, les graphes non orientés multiples sont un outil modélisation des réseaux d'interaction protéine-protéine (Pelle-

grini et al., 2004), le but dans ce cas est donc l'étude du fonctionnement des protéines par rapport à d'autre.

1.6.3 Le Graphe du web :

Le graphe du Web est un graphe orienté dont les sommets sont les pages du web et les arêtes modélise l'existence d'un lien hypertexte dans une page vers une autre (Brisaboa et al., 2009). Il représente l'un des graphes les plus volumineux : en juillet 2000 déja, on estimait qu'il contenait environ 2,1 milliards de sommets et 15 milliards d'arêtes avec 7,3 millions de pages ajoutées chaque jour (Guillaume and Latapy, 2002). De ce fait, ce graphe a toujours attiré l'attention des chercheurs. En effet, l'étude de ses caractéristiques a donné naissance à plusieurs algorithmes intéressants, notamment l'algorithme PageRank de classement des pages web qui se trouve derrière le moteur de recherche le plus connu de nos jours : Google.

1.7 Conclusion

Dans ce chapitre nous avons présenter les notions et les concepts généraux qui touchent a la théorie de graphes : définitions de graphes, leurs principales propriétés, leurs représentations et leurs domaines d'application.
Le point important qu'on a put tirer de cette partie est que les graphes sont devenue un moyen crucial et indispensable dans la modélisation des problèmes dans plusieurs domaines. Cependant ils devient de plus en plus complexe et volumineux suit a la grande quantités de données, ce qui rend leurs stockage, visualisation et traitement difficile. La compression de graphe est nait comme solution a ce problème. Dans le chapitre suivant nous allons présenter la compression de graphe, son rôle et ses différents méthodes.

Chapitre 2

Compression de graphe

2.1 Compression de données :

2.2 Compression appliquée aux graphes :

la compression de graphe a été proposé comme solutions pour le traitement et le stockage des graphe volumineux, elle permet de transformer un grand graphe en un autre plus petit tout en préservant ses propriétés générales et ses composants les plus importants. Les principaux avantages de la compression sont (?) :

— Réduction de la taille des données et de l'espace de stockage : De nos jours, les graphes représentant les bases de donnés, les réseaux sociaux et tout types de données numérique accroissent d'une manière exponentiel, ce qui rend leur stockage difficile et couteux en terme d'espace mémoire, les techniques de compression produisent des graphes plus petits qui nécessitent moins d'espace que leurs graphe d'origines. Cela permet aussi de décroitre le nombre d'opérations d'E/S, de réduire les communications entre nœuds dans un environnement distribué et de charger le graphe en mémoire central.

— Exécution rapide des algorithmes de traitement et des requêtes sur les graphe : l'exécution des différents algorithmes de traitement sur des graphes volumineux peut avérer couteux en terme de temps et peut ne pas donner les résultats attendues.

La compression permet d'obtenir de petits graphes qui peuvent être traiter, analyser et interroger plus efficacement et dans un temps raisonnable.

— Facilité d'analyse et visualisation du graphe : Les techniques de compression permettent de représenter les donnés et les structures des graphes massives d'une manière plus significative permettant ainsi leurs analyse et leurs visualisation contrairement au graphes d'origines qui ne peuvent même pas être charger en mémoire.

— Élimination du bruit : les grands graphes du web sont considérablement bruité, il contiennent des liens et des nœuds erronés, ce bruit peut perturber l'analyse en faussant les résultats et en augmentant la charge de travail liée au traitement des donnés. la compression permet donc de filtrer le bruits et de ne mettre en évidence que les donnés importantes.

2.2.1 Les types de compression :

2.2.2 Les métriques d'évaluation des algorithmes de compression :

2.3 Classification des méthodes de compression :

2.4 Méthodes de compression basés sur les k2-trees :

2.5 Méthodes de compression basés sur l'extraction de motifs :

2.5.1 VOG : Vocabulary Based Summarization of Graphs

Principe général : VOG est une méthode de base sur laquelle s'appuient plusieurs autres méthodes de compression par extraction de motifs, elle décrit le graphe en se basant sur un ensemble appelé vocabulaire composé d'un ensemble de structures communes qui apparaissent souvent dans les graphes du monde réel et qui ont généralement une signification sémantique.De plus VoG utilise le principe de longueur de description minimale (MDL) pour minimiser le cout de la description du graphe en terme de bits.

La problématique traité par VoG peut étre résumé comme suite : étant donné un graph statique non orienté G, on cherche a trouver un ensemble de sous-graphes chevauchés pour décrire de manière succinct le graphe donné en entré tout en minimisant le cout du codage en utilisant le principe MDL.

Le principe Minimum Description Length MDL est un concept de la théorie de l'information qui permet de codifier un ensemble de donnés en se basant sur un modèle, tout en minimisant le cout de codage par une fonction objective. Il peut être définit comme suite : étant donné un ensemble de modèles M, choisir celui qui minimise la fonction suivante : $min(D,M) = \mathrm{L}(M) + \mathrm{L}(D \mid M)$ où $\mathrm{L}(M)$ est la longueur, en bits de la description de M et $\mathrm{L}(D \mid M)$ est la longueur, en bits de la description des donnés en utilisant le modèle M. Pour utiliser le principe MDL dans VoG, il est d'abord essentiel de définir l'ensemble de modèles M, comment un modèle M peut il décrire un graphe et comment l'encoder. Nous notant aussi que pour une comparaison équitable entre les différents modèles, MDL nécessite que la description soit sans perte.

Soit un graphe non orienté G(V,E), sans boucle avec n nœuds et m arêtes et soit A sa matrice d'adjacence. le résultat de VoG est une liste ordonnée de structures noté M, on note par Ω le vocabulaire composé de six structures qui sont : clique (fc) et quasi-clique (nc), noyau bipartie (cb) et quasi-noyau bipartie (nb), étoile (st) et chaine (ch). On aura $\Omega = \{ fc,nc,cb,nb,st,ch \}$. Chaque structure $s \in M$ identifie une partie de la matrice d'adjacence A. Nous notons cette partie area(s). On peut avoir un chevauchement au niveau des nœuds, les liens quand a eux sont servit selon un ordre FIFO et ne peuvent pas étre chevauchés, e.i. la première structure $s \in M$ qui décrit l'arête dans A détermine sa valeur.

On note par $\mathcal{C}_x$ l'ensemble de tous les sous-graphes possible de type $x \in \Omega$, et $\mathcal{C}$ l'union de tous ces ensembles, $\mathcal{C} = \cup_x \mathcal{C}_x$. La famille de modèles noté $\mathcal{M}$ représente tous les permutations possibles des éléments de $\mathcal{C}$. Par MDL, on cherche $M \in \mathcal{M}$ qui minimise le mieux le cout de stockage du modèle et de la matrice d'adjacence.

Pour calculer le cout totale de description général, on construit d'abord une approximation $\mathbf{M}$ de la matrice d'adjacence, en utilisant le modèle M : nous considérons de

manière itérative chaque structure $s \in M$ et complétons la connectivité de area(s) dans **M**. Comme on a $\mathbf{M} \neq A$ et comme notre méthode est une méthode sans perte, on aura besoin d'une matrice d'erreur E pour reconstituer la matrice d'adjacence tel que E= A $\bigoplus$ **M**.

la fonction objective deviens donc : $min(D,M)$ = L(M) + L(E).

Pour l'encodage du modèle, on a pour chaque $M \in \mathcal{M}$:

$$\mathrm{L}(M) = L_{\mathbb{N}}(|\mathrm{M}|+1) + \log \begin{pmatrix} |M| + |\Omega\| - 1 \\ |\Omega| - 1 \end{pmatrix} + \sum_{s \in M} (-logPr(x(s)|M) + L(s))$$

Au début, on calcule le nombre de structures dans le modèle avec $L_{\mathbb{N}}$, et on encode le nombre de structures par type $x \in \Omega$. Ensuite pour chaque structure $s \in M$, on encode son type x(s) avec un code de préfixe optimal et enfin on encode la structure.

Le codage des structures se fait selon leurs type :

Clique : Pour l'encodage d'une clique, on calcule le nombre des nœuds de celle-ci, et on encode leurs ids :

$$\mathrm{L}(fc) = L_{\mathbb{N}}(|fc|) + \log \begin{pmatrix} n \\ |fc| \end{pmatrix}$$

Quasi-Clique : Les quasi cliques sont encoder comme des cliques complètes, toute en identifiant les arêtes ajoutés et manquantes en utilisant des codes de préfixe optimaux, on utilise $||nc||$ et $||nc||$' pour transmettre respectivement le nombre d'arêtes ajoutés et manquantes.

$$\mathrm{L}(nc) = L_{\mathbb{N}}(|nc|) + \log \begin{pmatrix} n \\ |nc| \end{pmatrix} + \log(|\mathrm{area}(nc)|) + ||nc||l_1 + ||nc||'l_0$$

Où l_1 = - log($||nc||/(||nc||+||nc||')$) et analogique à l_0 sont les longueurs des codes de préfixe optimaux des arêtes ajoutés et manquantes.

Noyau bipartie : notant par A et B les deux ensemble du noyau bipartie, On encode leurs tailles, ainsi que les ids de leurs sommets :

$$\mathrm{L}(fb) = L_{\mathbb{N}}(|A|) + L_{\mathbb{N}}(|B|) + \log \begin{pmatrix} n \\ |A| \end{pmatrix} + \log \begin{pmatrix} n - |A| \\ |B| \end{pmatrix}$$

Quasi-Noyau bipartie : Comme les quasi-cliques, les noyau bipartie sont codé comme suit :

$$\mathrm{L}(nb) = L_{\mathbb{N}}(|A|) + L_{\mathbb{N}}(|B|) + \log \begin{pmatrix} n \\ |A| \end{pmatrix} + \log \begin{pmatrix} n-|A| \\ |B| \end{pmatrix} + \log(|\mathrm{area}(nb)|) + ||nb||l_1 + ||nb||'l_0$$

Étoile : L'étoile est un cas particulier des noyau bipartie où l'un des ensembles est constituer d'un seule élément (appelé hub) et l'autre ensemble est constituer des sommets restants (appelé spokes). Le codage est calculer comme suit, d'abord on calcule la nombre de spokes de l'étoile, ensuite on identifie le hub parmi les n sommets et les spokes parmi les n-1 restants.

$$\mathrm{L}(st) = L_{\mathbb{N}}(|st|\text{-}1) + \log \mathrm{n} + \log \begin{pmatrix} n-1 \\ |st|-1 \end{pmatrix}$$

Chaine : On calcule d'abord le nombre d'éléments de la chaine, ensuite on encode les ids des nœuds selon leurs ordre dans la chaine :

$$\mathrm{L}(ch) = L_{\mathbb{N}}(|ch| \text{ - } 1) + \sum_{i=0}^{|ch|}(n-i)$$

Matrice d'erreur : la matrice d'erreur E est encoder sur deux parties E^+ et E^-. E^+ correspond a la partie de A que M modélise en rajoutant des liens non existants. contrairement à E^- qui représente les partie de A que M ne modélise pas. Notons que les quasi-clique et les quasi-noyau bipartie ne sont pas inclut dans la matrice d'erreur puisque ils sont encodés exactement donc on les ignorent. Le codage de E^+ et E^- est similaire a celui des quasi-clique, on a :

$$\mathrm{L}(E^+) = \log(|\mathrm{area}(|E^+|) + ||E^+||l_1 + ||E^+||'l_0$$
$$\mathrm{L}(E^-) = \log(|\mathrm{area}(|E^-|) + ||E^-||l_1 + ||E^-||'l_0$$

Algorithme de VoG : Pour la recherche du meilleur modèle $M \in \mathcal{M}$, VoG procède sur trois étapes :

1. Génération des sous-structures : Dans cette phase, Les méthodes de détection de communautés et de clustering sont utilisé pour décomposer le graphe en sous-graphes pas forcément disjoints. La méthode de décomposition utilisé dans VOG est SlashBurn.
2. Étiquetage des sous-graphes : L'algorithme cherche pour chaque sous-graphe généré dans l'étape précédente la structure $x \in \Omega$ qui le décrit le mieux, en tolérant un certain seuil d'erreur.
    a Étiquetage des structures parfaites : Tout d'abord, le sous-graphe est testé par rapport au structures complètes du vocabulaire (e.i. clique , étoile, chaine et noyau bipartie) pour une similarité sans erreur. Le test des cliques et des chaines est basé sur la distribution des degrés, Plus précisément, si tous les sommets d'un sous graphe d'ordre n ont un degré égale a n-1, il s'agit alors d'une clique. De même si tous les sommets ont un degré de 2 sauf deux sommets ayant le degré 1, le sous-graphe est une chaine. D'un autre coté, le sous-graphe est considéré comme noyau bipartie si les amplitudes de ses valeurs propres maximales et minimales sont égales, pour pouvoir identifier les sommets de chaque ensemble du noyau nous utilisons le parcours BFS avec la coloration des sommets. Quant a l'étoile, elle est considéré comme un cas particulier d'un noyau bipartie, il suffit donc que l'un des ensemble soit composé d'un seule sommet.
    b Étiquetage des structures approximative : Si le sous graphe ne correspond pas à une structures complète, on cherche la structure qui l'approxime mieux en terme de MDL. Pour ce faire, on le représente sous forme de chaque type $x \in \Omega$ et on choisit le type avec la description minimal.
    Remarque : Pour les structure parfaites (clique, étoile, chaine et noyau bipartie), en plus du cout d'encodage de la structure, on rajoute le cout de l'erreur local c'est a dire, $\mathrm{L}(x^*) + \mathrm{L}(E^+_{x^*}) + \mathrm{L}(E^-_{x^*})$ où $\mathrm{L}(x^*)$ est la description de la structure, $\mathrm{L}(E^+_{x^*})$ et $\mathrm{L}(E^-_{x^*})$ sont les arêtes incorrectement modélisés et les arêtes non modélisés respectivement dans area(x^*).

Après avoir représenter le sous graphe sous forme d'une structure, on l'ajoute a l'ensemble des structure candidates $\mathcal{C}$, en l'associant a son cout.

3. Assemblage du modèle : Dans cette dernier étape, une sélection d'un ensemble de structures parmi ceux de $\mathcal{C}$ est réaliser, des heuristiques de sélections sont utilisés car le nombre de permutations est très grand ce qui implique des calcules exhaustifs. Les heuristiques permettent d'avoir des résultats approximatives et rapides, parmi les heuristiques utilisés dans VOG on trouve :
   - PLAIN : Cette heuristique retourne tout les structures candidates. e.i. $M = \mathcal{C}$.
   - TOP-K : Cette heuristique sélectionne les k meilleurs candidats en fonction de leurs gain en bits.
   - GREEDY'N FORGET(GNF) : Parcours structure par structure dans l'ensemble $\mathcal{C}$ ordonnés par leurs qualité (gain en bits), ajoute la structure au modèle tant que elle n'augmente pas le cout total de la représentation, sinon l'ignore.

Ci-dessous le pseudo algorithme de VOG :

Algorithm 1 Pseudo Algorithme VOG

1: **Entré :** Graphe G.
2: **Étape 1 :** Génération des sous-graphe en utilisant une méthode de décomposition.
3: **Étape 2 :** Étiquetage des sous graphe en choisissant la structure $x \in \Omega$ qui représente chaque sous- graphe avec le moindre coût.
4: **Étape 3 :** Assemblage des sous graphes en utilisant des heuristiques pour sélectionner un sous-ensemble non-redondant à partir des structure candidate de l'étape 2.
5: **Sortie :** Modèle M et son cout d'encodage.

2.6 Méthodes de compression basés les k2-trees :

2.7 Conclusion

Chapitre 3

chapitre 03 : etude empirique

Definition 3.0.1. Here is a new definition

Deuxième partie

Conclusion

Random citation (Seo et al., 2018) embeddeed in text.

Random citation (Brisaboa et al., 2009) embeddeed in text.

Bibliographie

(2012). *Quelques rappels sur la théorie des graphes*. IUT Lyon Informatique.

Brisaboa, N. R., Ladra, S., and Navarro, G. (2009). k 2-trees for compact web graph representation. In *International Symposium on String Processing and Information Retrieval*, pages 18–30. Springer.

Fages, J.-G. (2014). *Exploitation de structures de graphe en programmation par contraintes*. PhD thesis, Ecole des Mines de Nantes.

Guillaume, J.-L. and Latapy, M. (2002). The web graph : an overview. In *Actes d'ALGOTEL'02 (Quatrièmes Rencontres Francophones sur les aspects Algorithmiques des Télécommunications)*.

Hennecart, F., Bretto, A., and Faisant, A. (2012). Eléments de théorie des graphes.

Jean-Charles Régin, A. M. (2016). Théorie des graphes. Technical report.

Koutra, D., Kang, U., Vreeken, J., and Faloutsos, C. (2015). Summarizing and understanding large graphs. *Statistical Analysis and Data Mining : The ASA Data Science Journal*, 8(3) :183–202.

Lehman, E., Leighton, F. T., and Meyer, A. R. (2010). Mathematics for computer science. Technical report, Technical report, 2006. Lecture notes.

Lemmouchi, S. (2012). *Etude de la robustesse des graphes sociaux émergents*. PhD thesis, Université Claude Bernard-Lyon I.

Lopez, P. (2003). Cours de graphes.

Müller, D. (2012). *Introduction à la théorie des graphes.* Commission romande de mathématique (CRM).

Parlebas, P. (1972). Centralité et compacité d'un graphe. *Mathematiques et sciences humaines*, 39 :5–26.

Pellegrini, M., Haynor, D., and Johnson, J. M. (2004). Protein interaction networks. *Expert review of proteomics*, 1(2) :239–249.

Rigo, M. (2010). *Théorie des graphesorie des graphes.* Université de liège, Faculté des sciences Département de mathématiques.

Roux, P. (2014). *Théorie des graphes.*

SABLIK, M. (2018). Graphe et langage.

Seo, H., Park, K., Han, Y., Kim, H., Umair, M., Khan, K. U., and Lee, Y.-K. (2018). An effective graph summarization and compression technique for a large-scaled graph. *The Journal of Supercomputing*, pages 1–15.