République Algérienne Démocratique et Populaire Ministère de

l'Enseignement Supérieur et de la Recherche Scientifique

**UNIVERSITÉ DJILALI BOUNAÂMA - KHEMIS-MILIANA**

**FACULTÉ DES SCIENCES ET DE LA TECHNOLOGIE**

**DÉPARTEMENT DES MATHÉMATIQUES ET DE L'INFORMATIQUE**

*Projet de fin d'études présenté pour l'obtention de*

*Diplôme de* ***Master en Informatique***

*Option* : ***Génie Logiciel et systèmes distribués***

Thème :

# Utilisation d'une approche deeplearning pour l'optimisation d'un algorithme de graphes

***Réalisé par :***

- ***BAHET Lounis***
- ***BENHALIMA Mohamed***

***Encadré par :***

***Mr. HARBOUCHE Oussama***

*Année Universitaire : 2019/2020*

# Dédicace…

Je ne veux pas laisser cette occasion pour vous exprimer mon attachement et les souvenirs reconnaissants que je garde de vous, je serai profondément heureux de dédier ce modeste travail :

A ma chère Maman : Douce, affable, honorable, aimable : tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager et de prier pour moi.Tu as étais toujours là pour moi quand personne d'autre n'a était ,Je te dédie ce travail en témoignage de mon profond amour. Puisse Dieu, le tout puissant, te préserver et t'accorder santé, longue vie et bonheur.je t'aime maman.

A mon cher Papa : Merci pour tout l'amour que tu m'as donné, aucune dédicace ne serait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour toi.

A mes cher frères merci d'être toujours à mes côtés.

A tous mes amis ….

A tous mes cousins et mes cousines

Lounis&mohamed

# SOMMAIRE

## Chapitre 03 : Les algorithmes de recherche dans les graphes

# Chapitre4 :Le calcul du chemin le plus court à l'aide de techniques d'apprentissage en profondeur

## Résumé :

Le graphe est un concept puissant pour la représentation des relations entre des paires d'entités. Les données ayant une structure de graphes sous-jacente peuvent être trouvées dans de nombreuses disciplines, décrivant des composés chimiques, des surfaces des modèles tridimensionnels, des interactions sociales ou des bases de connaissance, pour n'en nommer que quelques-unes.

L'apprentissage en profondeur s'est avéré efficace dans un certain Récemment, d'importants efforts de recherche ont été consacrés à l'application de méthodes d'apprentissage en profondeur pour optimiser des algorithme de graphes

## Abstract :

The graph is a powerful concept for the representation of relationships between pairs of entities. Data with an underlying graph structure can be found in many disciplines, describing chemical compounds, surfaces of three-dimensional models, social interactions or knowledge bases, to name a few.

Deep learning has been shown to be effective in a recent period. Significant research effort has been devoted to the application of deep learning methods to optimize graph algorithms.

## الملخص:

يمثل الرسم البياني مفهومًا قويًا لتمثيل العلاقات بين أزواج الكيانات. يمكن العثور على البيانات ذات بنية الرسم البياني الأساسية في العديد من التخصصات ، والتي تصف المركبات الكيميائية ، وأسطح النماذج ثلاثية الأبعاد ، والتفاعلات الاجتماعية أو قواعد المعرفة ، على سبيل المثال لا الحصر.

لقد ثبت أن التعلم العميق فعال في بعض الأوقات الحديثة ، وقد تم تكريس جهود بحثية كبيرة لتطبيق أساليب التعلم العميق لتحسين خوارزميات الرسم البياني.

# Introduction generale :

Ces dernières années l'apprentissage "profond" ou "deep learning" fait beaucoup parler de lui. Et pour cause, ce sous ensemble de l'apprentissage machine ('machine learning") s'est imposé de manière impressionnante dans plusieurs champs de recherche: reconnaissance faciale, synthèse vocale, traduction automatique, et bien d'autres.

Il en résulte que les premiers modèles qui ont été mis en place dans ces domaines ont été construits à partir d'une certaine expertise métier (pour la reconnaissance vocale: la décomposition en phonèmes, pour la traduction machine: le passage par des règles grammaticales et syntaxiques). Des années de recherche ont été consacrées à l'exploitation et à la transformation de ces données non structurées de manière à en tirer du sens.

L'objectif de cette thèse est d'étudier l'approche de DL sur des algorithmes de graphes ce qui progrès des techniques d'analyse de graphes nous passons en revue de manière exhaustive les différents types de méthodes d'apprentissage en profondeur sur graphiques.

# Chapitre I

## Théorie des graphes

## 1. Introduction

Lorsqu'un humain analyse ou synthétise un cadre de conception en utilisant la représentation numérique administrant sa conduite, il crée un modèle scientifique du cadre du bâtiment, puis contrôle les conditions d'utilisation en les apprenants et leur lien avec la réalité physique. Dans la pratique de conception régulière, on utilise un modèle qui est connu pour être raisonnable pour le cadre actuel et le point de calcul. Les graphes sont considérés comme un superbe appareil d'affichage qui est utilisé pour montrer de nombreux types de relations entre toutes les circonstances physiques. De nombreux problèmes de vrai peuvent être évoqués par des graphes. Les graphes sont l'un des principaux objets de concentration en mathématiques discrètes. L'hypothèse du graphe est l'une des branches de l'arithmétique actuelle qui a connu récemment une amélioration des plus remarquables. Au départ, l'hypothèse de graphe n'était qu'une accumulation de problèmes récréatifs ou de tests comme les visites d'Euler ou les quatre ombrages d'un guide, sans association indubitable entre eux[1] [2].

La théorie des graphes est utilisée pour trouver des solutions aux problèmes du monde réel, en plus du fait que l'informatique de nouvelle génération dépend de la théorie des graphes. Les idées théoriques graphiques sont très utilisées par les applications informatiques. Surtout dans les domaines de recherche de l'informatique tels que l'exploration d'informations, la division d'images, le regroupement, la capture d'images, l'organisation et ainsi de suite. Par exemple, une structure d'information peut être planifiée comme un arbre qui, ainsi, sommets et arêtes utilisés. Ainsi, l'affichage des topologies du système devrait également être possible en utilisant des idées de graphiques. De même, l'idée la plus vitale de l'ombrage de graphe est utilisée dans la portion d'actif, la planification. De plus, les voies, les promenades et les circuits dans l'hypothèse du graphe sont utilisés dans le cadre d'applications colossales, disent le problème des représentants commerciaux, les idées de base de données, l'organisation des actifs. Cela incite à faire avancer de nouveaux calculs et de nouvelles hypothèses qui peuvent être utilisés dans le cadre d'applications colossales [3]

Dans ce chapitre on va parler sur la théorie des graphes histoire et définition, les type des graphes détaillé, sous graphe, Isomorphisme et Automorphisme de graphe, réseau, densité, arbres, et ainsi comment représenter les graphes, nous démontrons aussi l'application de la théorie des graphes dans différents domaines de l'informatique.

## 2. L'histoire de la théorie des graphes

L'histoire de la théorie des graphes (ou des "complexes cellulaires") débute peut-être avec les travaux d'Euler au 18ème siècle et trouve son origine dans l'étude de certains problèmes, tels que celui des ponts de Königsberg (les habitants de Königsberg se demandaient s'il était possible, en partant d'un quartier quelconque de la ville, de traverser tous les ponts sans passer

deux fois par le même et de revenir à leur point de départ), la marche du cavalier sur l'échiquier ou le problème du coloriage de cartes et du plus court trajet entre deux points.

La théorie des graphes s'est alors développée dans diverses disciplines telles que la chimie (isomères), la biologie, les sciences sociales (réseaux de transports), gestion de projets (C.P.M.), informatique (topologie des réseaux, complexité algorithmique, protocoles de transferts), la

physique quantique, etc. Depuis le début du 20ème siècle, elle constitue une branche à part entière des mathématiques, grâce aux travaux de König, Menger, Cayley puis de Berge et d'Erdös. Cette branche des mathématiques a connu un grand regain d'intérêt suite à l'émergence des réseaux sociaux Internet dont les connexions entre "amis" et "suiveurs" constituent des graphes dont l'analyse topologique et statistique peut nous apprendre de nombreuses choses.

De manière générale, un graphe permet de représenter la structure, les connexions d'un ensemble complexe en exprimant les relations entre ses éléments: réseau de communication, réseaux routiers, interaction de diverses espèces animales, circuits électriques, ...

Les graphes constituent donc une méthode de pensée qui permet de modéliser une grande variété de problèmes en les ramenant à l'étude de sommets et d'arcs.

Les derniers travaux en théorie des graphes sont souvent effectués par des informaticiens, du fait de l'importance que revêt l'aspect algorithmique dans leur domaine (voir le début du chapitre de Méthodes Numériques pour un petit exemple).

Effectivement, il s'agit essentiellement de modéliser des problèmes. Nous exprimons le problème en termes de graphes de sorte qu'il relève d'un problème de la théorie des graphes que nous savons le plus souvent résoudre car rentrant dans une catégorie de problèmes connus.

Les solutions de problèmes de graphes peuvent être faciles et efficaces (le temps nécessaire pour les traiter informatiquement étant raisonnable car ils dépendent polynomialement du nombre de sommets du graphe) ou difficiles (le temps de traitement étant alors exponentiel) auquel cas nous utilisons une heuristique, c'est-à-dire un processus de recherche d'une solution (pas forcément la meilleure).

Si la théorie des graphes connaît un assez grand engouement ces 30 dernières, peut-être est-ce dû au fait qu'elle ne nécessite pas dans ses concepts élémentaires de bagage mathématique considérable. Effectivement, il suffit d'avoir parcouru les chapitres de Probabilités, de Théorie Des Ensembles et d'Algèbre Linéaire ainsi que de Topologie présentés sur le site pour déjà se sentir à l'aise avec les différentes définitions.[4]

## 3-Théorie des graphes:

La théorie des graphes est une branche des mathématiques discrètes qui traite de la manière dont les objets sont connectés. Ainsi, la connectivité dans un système est une qualité fondamentale de la théorie des graphes. Le concept principal en théorie des graphes est celui d'un graphe. Pour un mathématicien, un graphe est l'application d'un ensemble sur lui-même, c'est-à-dire une collection d'éléments de cet ensemble et les relations binaires entre ces éléments. Les graphiques sont des objets unidimensionnels, mais ils peuvent être intégrés ou réalisés dans des espaces de dimensions supérieures [5].

## 4.Définitions préliminaires:

Les paragraphes suivants présentent un certain nombre de définitions largement utilisées, utilisées plus loin dans ce chapitre

## 5.Types de graphiques:

D'une manière générale**,** un graphe G (V, E) est défini comme étant un ensemble V de sommets (nœuds) qui sont interconnectés par un ensemble E d'arêtes (liens). Le nombre d'éléments N dans V s'appelle l'ordre de G et on note | V | = N et le nombre d'éléments M dans E est la taille de G et on note | E | = M

Simple et multigraphes: dans un graphe G, lorsque deux sommets quelconques de V sont reliés par plusieurs arêtes, le graphe est appelé multigraphie. Un graphique sans boucles et avec au plus une arête entre deux sommets est appelé un graphe simple. Les graphes utilisés dans FSM sont supposés être des graphes étiquetés simples.

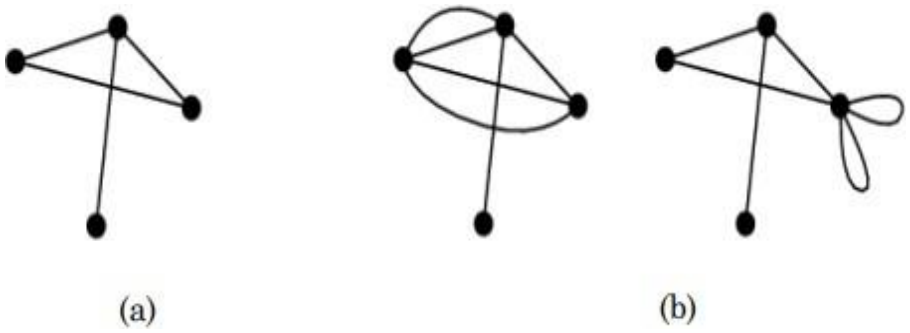



**Figure II-8:** (a) graphe simple, (b) nom graphe simple ayantarête multiple (gauche) nom graphe simple ayant des boucles (droite).

Dans un graphe G (V, E) pour une arête e = {u, v}, on dit:

- e connecte u et v
- u et v sont les points finaux de e.
- u et e sont des incidents (v et e sont des incidents).
- u et v sont adjacents ou voisins

Le degré deg (v) d'un sommet v est le nombre d'arêtes qui lui sont incidentes. Un sommet de degré 0 est appelé isolé (6).

Graphes réguliers: un graphe G dans lequel tous les sommets sont de degré égal est appelé un graphe régulier, un graphe régulier de degré k est également appelé k - régulier

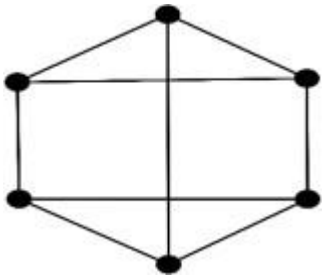
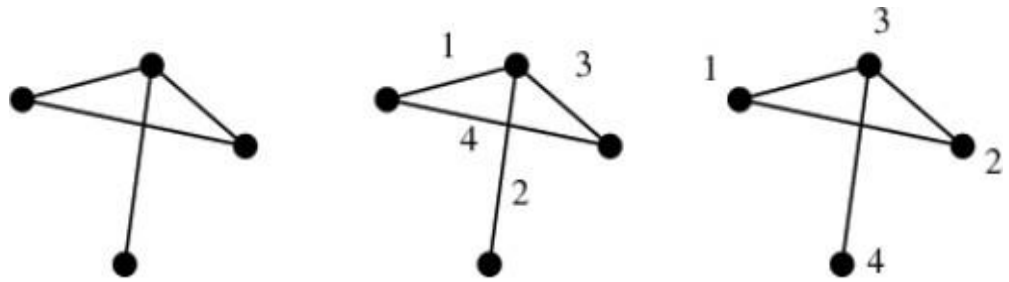


**Figure I.1:** Graphe 3-régulier k = 3.

Graphes étiquetés: un graphe étiqueté peut être représenté par G (V, E, LV, LE, $\varphi$), où V est un ensemble de sommets, E □V x V est un ensemble d'arêtes; LV et LE sont des ensembles d'étiquettes de sommet et d'arête. respectivement; et $\varphi$ est une fonction d'étiquette qui définit les correspondances V → LV et E → LE. Lorsque les étiquettes de bord sont membres d'un

ensemble ordonné (par exemple, les nombres réels), un graphe étiqueté peut être appelé graphe pondéré. [7]




*graphe non étiqueté* *graphe marqué par des arêtes* *graphe étiqueté par sommet*

**Figure I.2:** Graphes sans étiquette et étiquetés

**Graphes orientés et non orientés:** un graphe G n'est pas orienté si ∀ e ∈ E, e est une paire de sommets non ordonnée et est dirigé sinon

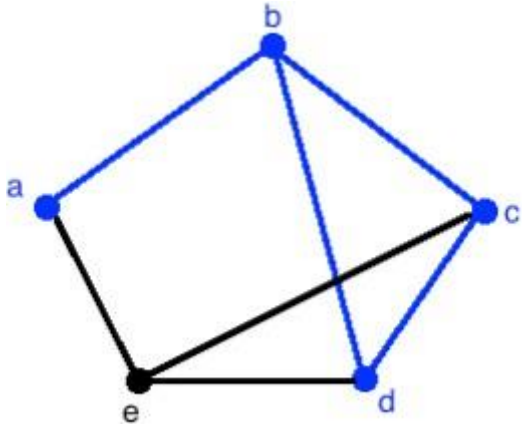

**Figure II-11:** Graphe non dirigé (à gauche), Graphe dirigé (à droite)

Promenades, sentiers, chemins et cycles: la marche d'un graphe G est une alternance séquence de sommets et d'arêtes v0, v1, v1,… , v i − 1, ei, vi commençant et finissant par des sommets,

dans lesquels chaque arête est incidente avec deux sommets qui le précèdent et le suivent immédiatement. Cette promenade rejoint les sommets v0 et vi et peut également être désignée par v1, v2,… , vi(les arêtes étant évidentes par le contexte). La longueur l d'une promenade

est le nombre d'occurrences d'arêtes dans celle-ci; v0 est appelé le sommet initial de la promenade, tandis que vi est appelé le sommet terminal de la promenade.

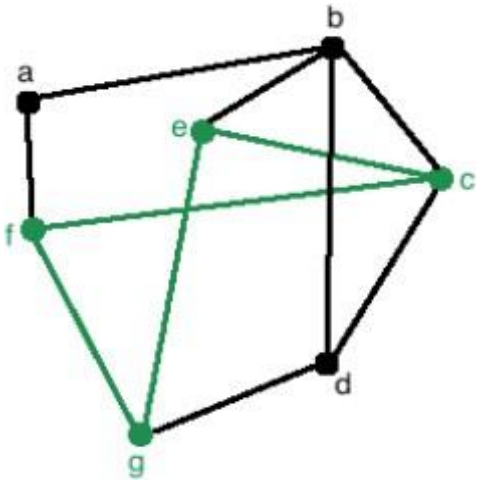
Par exemple, le graphique ci-dessous décrit une éventuelle promenade (en bleu). La marche est notée abcdb. Notez que les promenades peuvent avoir des bords répétés. Par exemple, si nous avions la marche abcdcbce, alors ce serait parfaitement bien même si certains bords sont répétés




Dans le graphique ci-dessus, la longueur de la promenade est abcdb 4 car elle traverse 4 bords.

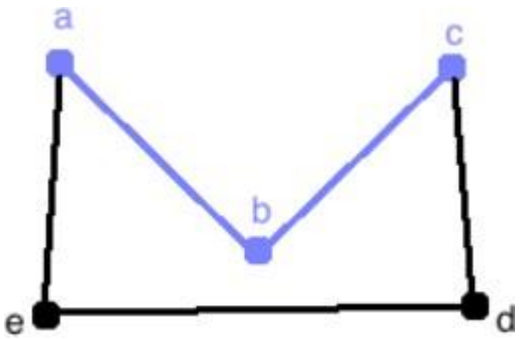
Une promenade fermée est une promenade vi-vi, c'est-à-dire une promenade qui commence et se termine au même sommet vi. Sinon, une promenade est dite ouverte.

Par exemple, le graphique suivant montre une promenade fermée en vert:



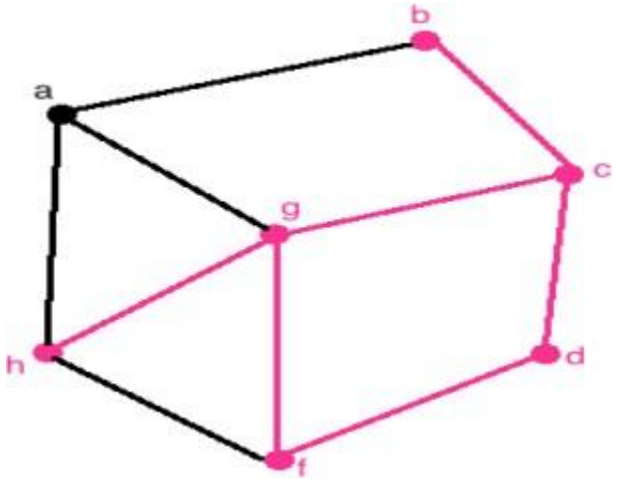
Notez que la promenade peut être définie par cegfc et que les sommets de début et de fin de la promenade sont c. C'est pourquoi cette promenade est fermée.

Une promenade est un sentier si tous les bords sont distincts. Jusqu'à présent, les deux précédents les exemples peuvent être considérés comme des traînées car il n'y a pas de bords répétés. Voici un autre exemple de sentier:




Notez que la promenade peut être définie comme abc. Il n'y a pas de bords répétés, donc cette promenade est aussi un sentier.

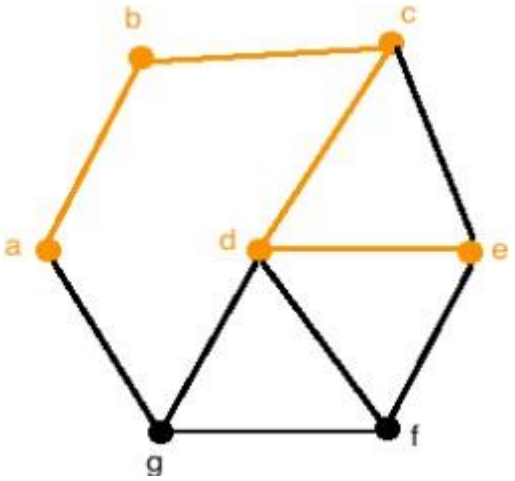
Regardons maintenant le graphique suivant:



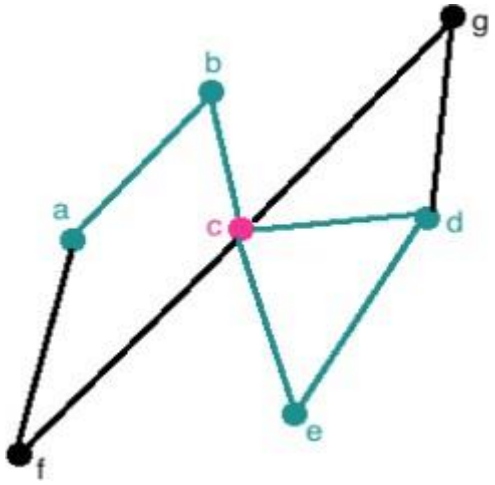

Notez que cette promenade doit répéter au moins un bord.

Une promenade est un chemin si tous les sommets (et donc nécessairement toutes les arêtes) sont distincts, autrement dit, un chemin est défini comme un chemin ouvert sans sommets répétés. Notez que tous les chemins doivent donc être des chemins ouverts, puisqu'un chemin ne peut à la fois commencer et se terminer au même sommet. Par exemple, la promenade de couleur orange suivante est un chemin:

parce que la promenade abcde ne répète aucun bord.

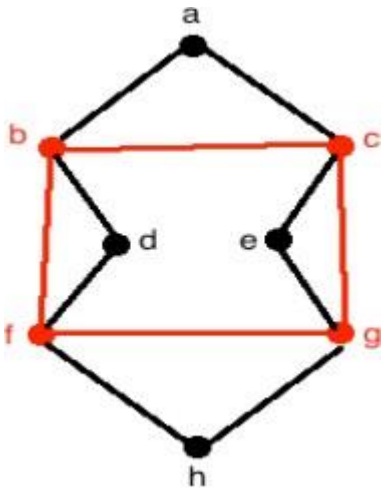



Regardons maintenant le graphique suivant avec la marche des sarcelles. Cette promenade n'est PAS un chemin puisqu'elle répète un sommet, à savoir le sommet rose c:




De plus, un cycle dans G est défini comme un tracé fermé dans lequel aucun autre sommet n'est répété en dehors du sommet de début / fin. Et de plus, un graphe acyclique G est défini comme un graphe sans cycle.

Vous trouverez ci-dessous un exemple de cycle. Remarquez comment aucune arête ne se répète dans la promenade ***bcgfb***, ce qui en fait un sentier, et que les sommets de début et de fin b sont identiques, ce qui les rend fermés.




**Graphes connectés et complets**

G est connecté, s'il contient un chemin pour chaque paire de sommets et est déconnecté sinon. G est complet (clique) si chaque paire de sommets est reliée par une arête. [8]

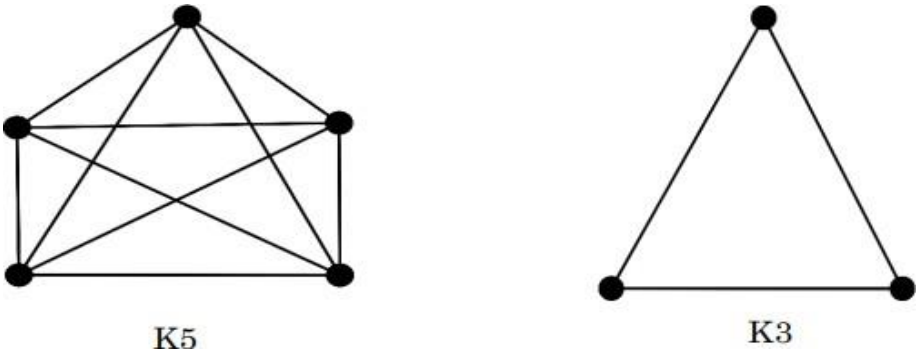



**Figure I.3:** Deux graphes complets et connectés avec 5 et 3 sommets respectivement, lorsqu'ils sont considérés ensemble, ils forment un graphe déconnecté.

### 6.Sous-graphe:

Étant donné deux graphes G1 (V1, L1, LV1, LE1, $_{\varphi}$1) et G2 (V2, E2, LV2, LE2, $_{\varphi}$2), G1 est un sous- graphe général appelé de G2, si G1 est satisfait: (I) V1 V2, et ∀v V1, $_{\varphi}$ 1 (v) = $_{\varphi}$ 2 (v), (ii) E1 E2 et (u, v) E1, $_{\varphi}$ 1 (u, v) = $_{\varphi}$ 2 (u, v) . La figure II-13 (a) est un exemple du sous-graphe général dans lequel un sommet v5 et des arêtes e4; e6; e7; e8; G1 est un sous-graphe induit de G2, si G1 satisfait davantage: Gu, v ∈ V1, (u, v) E1 (u, v) E2, simplement, un sous-graphe induit G1 d'un graphe G2 a un sous-ensemble des sommets de G2 et les mêmes arêtes entre paires de sommets que dans G2. en plus des conditions ci-dessus. G2 est aussi un supergraphe de G1 La figure II-13 (c) est un exemple du sous-graphe induit dans lequel un sommet v5 est omis. Dans ce cas, seuls les bords e8 et e9 sont également omis, et e4; e6; e7 sont conservés car ils existent entre v1; v3 et v4 dans l'original G. [7]

La troisième classe importante et générique de la sous-structure est un sous-graphe connecté où V1⊆ V2, E1⊆ E et tous les sommets de V1 sont mutuellement accessibles par certaines arêtes de E1. La figure II-13 (d) est un exemple où v6 est omis de (c).

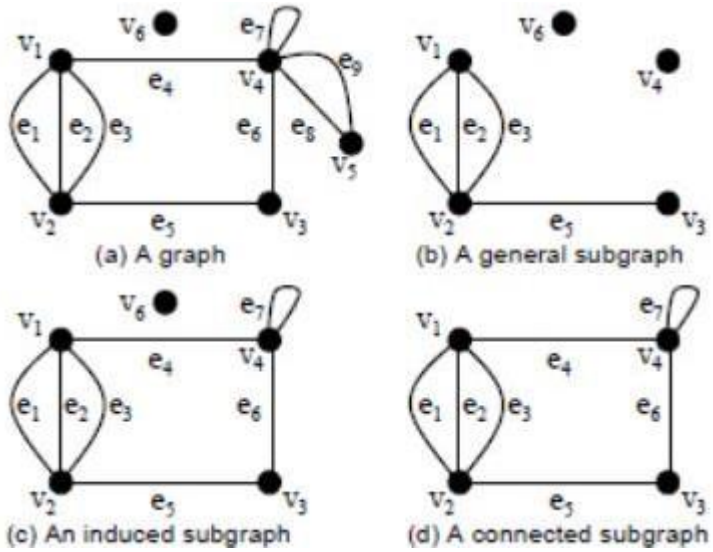



**Figure I.4:** (a) un graphe G, (b), (c), (d) représente respectivement un sous-graphe général, induit et connecté .

## 7.Isomorphisme de graphe

Un graphe G1 (V1, L1, LV1, LE1, $_{\varphi}$ 1) est isomorphe à un autre graphe G2 (V2, L2, LV2, LE2, $_{\varphi}$ 1) et est noté: G1 - G2 si et seulement si une bijection f: V1 → V2 existe tel que: (i) ∀u ∈V1, $_{\varphi}$ 1 (u) = $_{\varphi}$ 2 (f (u)), (ii) (u, v) E1 (f (u), f (v )) ∈ E2, (iii) (u, v) E1, $_{\varphi}$ 1 (u, v) = $_{\varphi}$ 2 (f (u), f (v)). La bijection f est un isomorphisme entre G1 et G2. Un graphe G1 est un sous-graphe isomorphe à un graphe G2, si et seulement s'il existe un sous-graphe g G2 tel que g1 soit isomorphe à g. Dans ce cas, on appelle g une incorporation de G1 dans G2. [7]

La figure II-15 et la figure II-16 présentent respectivement un isomorphisme de graphe et de sous-graphe:

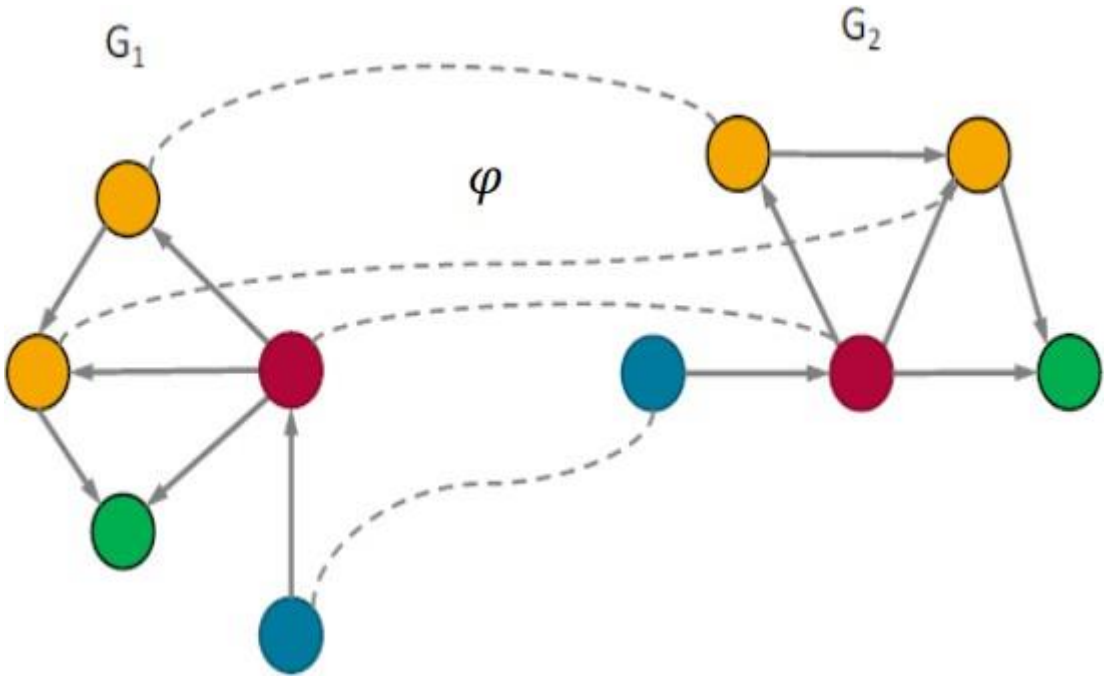



**Figure I.5:** Isomorphisme de graphe

Les deux graphes ci-dessus sont isomorphes, malgré leurs dessins différents

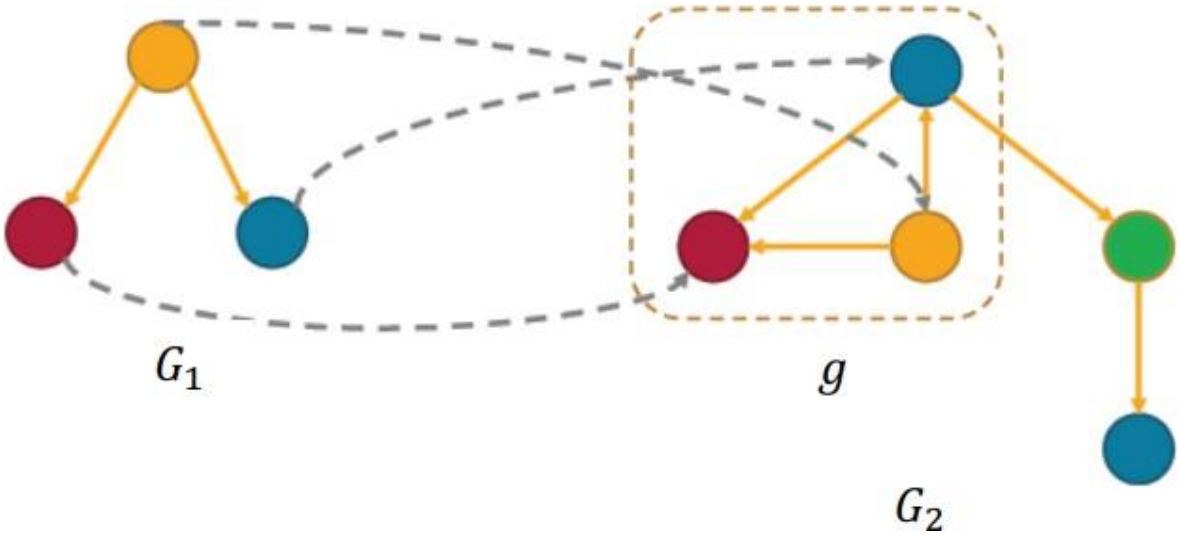



**Figure II-15:** Isomorphisme de sous-graphe.

Pensez à un isomorphisme de graphe comme à une structure préservant la bijection, et il découle de la définition que les graphes isomorphes sont identiques, mais dessinés différemment.

Généralement, le problème de la reconnaissance des graphes isomorphes est l'un des grands problèmes non résolus de la théorie des graphes. Constructionde tous N! Les mappages possibles d'un graphe à un autre (où N est le nombre de sommets), bien qu'évidemment peu pratiques, restent le seul moyen sûr de vérifier l'isomorphisme des graphes.

Un invariant d'un graphe G est une quantité associée à G qui a la même valeur pour tout graphe isomorphe à G. Par conséquent, les invariants de graphe sont des quantités indépendantes de l'étiquetage des sommets d'un graphe. Par conséquent, le nombre de sommets et le nombre d'arêtes sont des invariants. Un ensemble complet d'invariants détermine un graphe allant jusqu'à l'isomorphisme.

## 8.Automorphisme

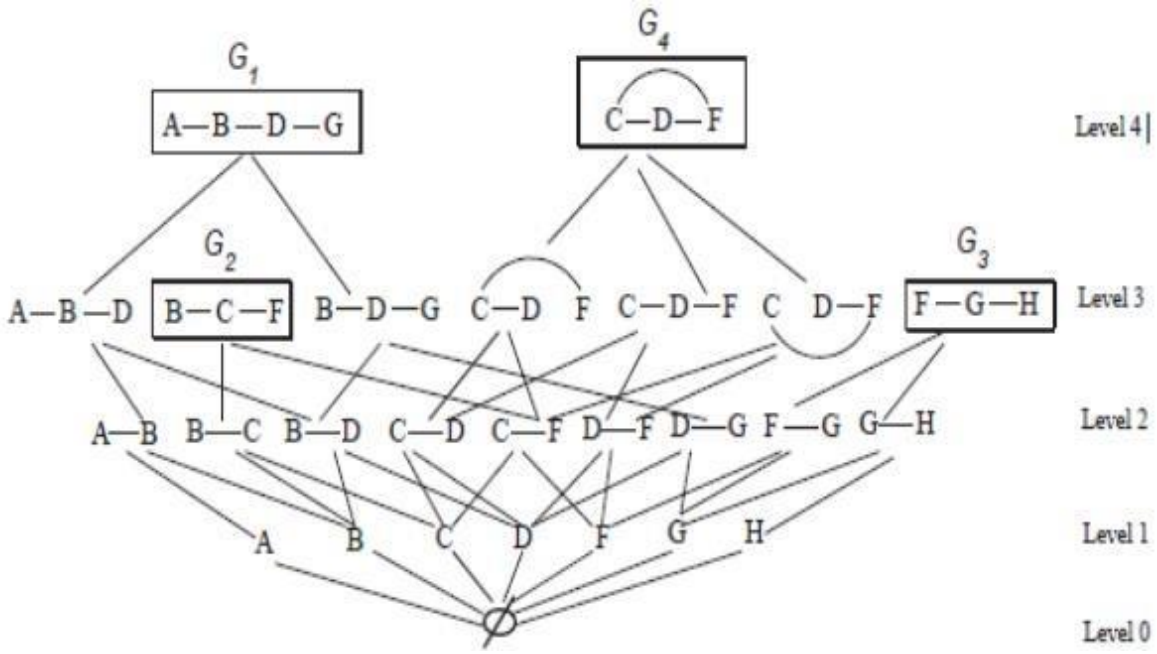
Un mappage isomorphe des sommets d'un graphe G sur eux-mêmes (ce qui préserve également la relation d'adjacence) est appelé un automorphisme d'un graphe G. Evidemment, chaque graphe possède un automorphisme trivial appelé automorphisme d'identité. Pour

certains graphiques, c'est le seul automorphisme; ce sont les graphes d'identité. L'ensemble de tous les automorphismes d'un graphe G forme un groupe appelé groupe d'automorphisme de G. [9]

**Réseau**

Dans une base de données G, un réseau est une forme structurelle utilisée pour modéliser l'espace de recherche afin de rechercher des sous-graphes fréquents, chaque sommet représentant un sous-graphe connecté du graphe en G. Le sommet le plus bas représente le sous-graphe vide et les sommets du niveau le plus élevé. les graphes de G. Un sommet p est un parent du sommet q du réseau, si q est un sous-graphe de p, et que q est différent de p de exactement un bord. Le sommet q est un enfant de p. Tous les sous-graphes de chaque graphe Gi ∈ G apparaissant dans la base de données sont présents dans le réseau et chaque sous-graphe n'y apparaît qu'une fois [7].

**Figure I.6:** Réseau (G)




**Exemple:** étant donné un ensemble de données graphes G = {G1, G2, G3, G4}, le réseau correspondant (G), est donné à la figure II-16. Sur la figure, le sommet le plus bas représente le sous-graphe vide et les sommets du niveau le plus élevé correspondent à G1, G2, G3 et G4. Les parents du sous-graphe B - D sont les sous-graphes A - B - D (joignant le bord A - B) et B

- D - G (joignant le bord D - G). De même, les sous-graphes B - C et C - F sont les enfants du sous-graphe B - C - F.

### 9.Densité

La densité d'un graphe G = (V; E) est calculée par

**densité (G) =** $2 \cdot \frac{|E|}{(|V| \cdot (|V| - 1))}$.

La densité du graphique mesure le rapport entre le nombre d'arêtes et le nombre maximal d'arêtes dans un graphique complet. Un graphique est dit dense si le rapport est proche de 1 et est considéré comme rare si le rapport est proche de 0. [10]

### 10. Arbres

Arbre libre: un arbre libre est défini comme un graphe non dirigé, connecté et acyclique. Arbre non ordonné étiqueté: Un arbre non ordonné étiqueté (un arbre non ordonné, en abrégé) est un graphe acyclique dirigé désigné par T (V, , E, vr), où V est un ensemble de sommets de T; est une fonction d'étiquetage telle que vi∈ V, (vi) → vi; E ⊆ V × V est un ensemble d'arêtes de T; et vr est un sommet distingué appelé racine de T. Pour vi ∈ V, il existe un chemin unique (vr, v1, v2,……., vi) de la racine vr à vi. Si un sommet vi se trouve sur le chemin allant de la racine au sommet vj, alors vi est un ancêtre de vj et vj est un descendant de vi. Pour chaque bord (vi, vj) E, vi est le parent de vj et vj est un enfant de vi. Les sommets qui partagent le même parent sont des frères et soeurs. La taille de T est définie comme étant le nombre de sommets dans T. Un sommet sans enfant est un sommet de feuille; sinon c'est un sommet intermédiaire. Le chemin le plus à droite de T est le chemin allant du sommet de la racine à la feuille la plus à droite. La profondeur (niveau) d'un sommet est la longueur du chemin allant de la racine à ce sommet.

Arbre ordonné étiqueté: Un arbre ordonné étiqueté (un arbre ordonné, en abrégé) est un arbre étiqueté non ordonné mais avec un ordre de gauche à droite imposé aux enfants de chaque sommet.

Les autres formes de sous-arbres incluent les suivantes: sous-arbre de bas en haut, sous-arbre induit et sous-arbre intégré.

### 11. Représentation des graphes

Il existe deux façons classiques de représenter un graphe en machine : par une matrice d'adjacence ou par un ensemble de listes d'adjacence.

**Représentation par matrice d'adjacence :**

Soit le graphe G = (S, A). On suppose que les sommets de S sont numérotés de 1 à n, avec

n =| S |. La représentation par matrice d'adjacence de G consiste en une matrice booléenne M de taille n × n telle que M[i][j] = 1 si (i, j) ∈ A, et M[i][j] = 0 sinon.

Si le graphe est valué (par exemple, si des distances sont associées aux arcs), on peut utiliser une matrice d'entiers, de telle sorte que M[i][j] soit égal à la valuation de l'arc (i, j) si (i, j) ∈

A. S'il n'existe pas d'arc entre 2 sommets i et j, on peut placer une valeur particulière (par exemple 0 ou −∞ ou null) dans M[i][j] .

Dans le cas de graphes non orientés, la matrice est symétrique par rapport à sa diagonale descendante. Dans ce cas, on peut ne mémoriser que la composante triangulaire supérieure de la matrice d'adjacence.

**Taille mémoire nécessaire :** la matrice d'adjacence d'un graphe ayant n sommets nécessite de l'ordre de O(n*n ) emplacements mémoire. Si le nombre d'arcs est très inférieur à n*n , cette représentation est donc loin d'être optimale.

**Opérations sur les matrices d'adjacence :** le test de l'existence d'un arc ou d'une arête avec une représentation par matrice d'adjacence est immédiat (il suffit de tester directement la case correspondante de la matrice). En revanche, le calcul du degré d'un sommet, ou l'accès à tous les successeurs d'un sommet, nécessitent le parcours de toute une ligne (ou toute une colonne) de la matrice, quel que soit le degré du sommet. D'une façon plus générale, le parcours de l'ensemble des arcs/arêtes nécessite la consultation de la totalité de la matrice, et prendra un temps de l'ordre de n*n . Si le nombre d'arcs est très inférieur à n*n , cette représentation est donc loin d'être optimale.[11]

**Representation par lists d'adjacence**

Soit le graphe G = (S, A). On suppose que les sommets de S sont numérotés de 1 à n, avec n

=| S |. La représentation par listes d'adjacence de G consiste en un tableau T de n listes, une pour chaque sommet de S. Pour chaque sommet si ∈ S, la liste d'adjacence T[si ] est une liste chainée de tous les sommets sj tels qu'il existe un arc ou une arête (si , sj ) ∈ A. Autrement dit, T[si ] contient la liste de tous les sommets successeurs de si . Les sommets de chaque liste d'adjacence sont généralement chainés selon un ordre arbitraire. Si le graphe est valué (par exemple, si les arêtes représentent des distances), on peut stocker dans les listes d'adjacence, en plus du numéro de sommet, la valuation de l'arête. Dans le cas de graphes non orientés, pour chaque arête (si , sj ), on aura sj qui appartiendra à la liste chainée de T[si ], et aussi si qui appartiendra à la liste chainée de T[sj ].

**Taille mémoire nécessaire :** si le graphe G est orienté, la somme des longueurs des listes d'adjacence est égale au nombre d'arcs de A, puisque l'existence d'un arc (si , sj ) se traduit par la présence de sj dans la liste d'adjacence de T[si ]. En revanche, si le graphe n'est pas orienté, la somme des longueurs de toutes les listes d'adjacence est égale à deux fois le nombre d'arêtes du graphe, puisque si (si , sj ) est une arête, alors si appartient à la liste d'adjacence de T[sj ], et vice versa. Par conséquent, la liste d'adjacence d'un graphe ayant n sommets et m arcs ou arêtes nécessite de l'ordre de O(n + m) emplacements mémoires.

**Opérations sur les listes d'adjacence :** le test de l'existence d'un arc ou d'une arête (si , sj ) avec une représentation par liste d'adjacence est moins direct que dans le cas d'une matrice d'adjacence (il n'existe pas de moyen plus rapide que de parcourir la liste d'adjacence de T[si

] jusqu'à trouver sj ). En revanche, le calcul du degré d'un sommet, ou l'accès à tous les successeurs d'un sommet, est plus efficace que dans le cas d'une matrice d'adjacence : il suffit de parcourir la liste d'adjacence associée au sommet. D'une façon plus générale, le parcours de l'ensemble des arcs/arêtes nécessite le parcours de toutes les listes d'adjacence, et prendra un temps de l'ordre de p, où p est le nombre d'arcs/arêtes (à comparer avec n*n dans le cas d'une représentation par matrice d'adjacence).[11]

En revanche, le calcul des prédécesseurs d'un sommet est mal aisé avec cette représentation, et nécessite le parcours de toutes les listes d'adjacences de T. Une solution dans le cas où l'on a besoin de connaitre les prédécesseurs d'un sommet est de maintenir, en plus de la liste d'adjacence des successeurs, la liste d'adjacence des prédécesseurs.

## 12. Application de théorie de graphe :

Les graphiques sont parmi les modèles les plus universels de structures caractéristiques et artificielles. Ils peuvent être utilisés pour montrer de nombreuses sortes de relations et de flux de processus dans des cadres physiques, naturels et sociaux. De nombreux problèmes d'intrigue fonctionnelle peuvent être évoqués par des graphes. À partir de la vue d'ensemble de l'étude de domaine de la théorie des graphes et de la poursuite de l'étude dans ce document d'enquête dans une autre section, nous démontrons l'application de la théorie des graphes dans différents domaines de l'informatique. Certaines des applications importantes sont expliquées ici:

### 1. Exploration de texte :

Les méthodes automatisées d'analyse de texte et d'exploration de texte ont reçu beaucoup d'attention en raison de l'augmentation remarquable des documents numériques. La technologie informatique a apporté un changement radical à notre vie quotidienne. De nos jours, en utilisant des méthodes numériques, nous pouvons stocker, gérer et récupérer des informations dans des documents texte automatiquement sans regarder les documents imprimés. L'analyse et l'exploration de texte automatisées deviennent de plus en plus importantes dans les applications informatiques. Les modèles de diagramme ont la capacité de capturer des données auxiliaires dans les écrits, mais ils ne prennent pas en compte les relations sémantiques entre les mots [12] [13].

### 2. Technologie de réseau :

Le domaine de la théorie des graphes joue un rôle indispensable dans différents domaines. L'une des régions essentielles dans l'hypothèse du graphique est la correspondance du système distant. Une partie de l'hypothèse du diagramme est identifiée par la disposition des problèmes dans les réseaux mobiles ad hoc (MANETS). Dans les systèmes ad hoc, les problèmes, par exemple, la disponibilité, la polyvalence, la direction,

La démonstration du système et la reproduction doivent être envisagées. Puisqu'un système peut être affiché sous forme de graphique, le modèle peut être utilisé pour résoudre ces problèmes. Les

graphiques peuvent être appelés arithmétiquement sous forme de réseaux. De même, les systèmes peuvent être mécanisés par des méthodes de calcul. Les problèmes, par exemple, l'épaisseur du moyeu, la polyvalence entre les moyeux, l'agencement de connexion

entre les moyeux et la direction du faisceau doivent être recréés. Pour reproduire ces idées, l'hypothèse de diagramme arbitraire est poursuivie. Les problèmes de disponibilité sont étudiés en utilisant des clés de graphique, (une clé géométrique ou un diagramme à clé k ou une clé k a d'abord été présenté sous forme de diagramme pondéré sur un arrangement de focales comme ses sommets et chaque combinaison de sommets a un moyen entre leur poids au plus k fois la séparation spatiale entre ces foyers, pour un k établi.) graphiques de voisinage, (Un diagramme de proximité est essentiellement un diagramme dans lequel deux sommets sont associés par une arête si et juste si les sommets remplissent des conditions géométriques spécifiques ), la scarification et l'hypothèse épouvantable du graphique.

Différents calculs sont également accessibles pour étudier le blocage dans MANET où ces systèmes sont affichés en vue de pensées hypothétiques de diagramme [14].

### 3. Système de recherche d'informations :

Un réseau est défini comme un système d'éléments qui interagissent ou se régulent mutuellement. Les réseaux peuvent être représentés mathématiquement sous forme de graphiques. En règle générale, le terme «graphique» fait référence à des représentations visuelles de la variation d'une variable par rapport à d'autres variables, ou au concept mathématique d'un ensemble de sommets reliés par des arêtes, ou à des structures de données basées sur ce concept mathématique; tandis que le terme «réseau» se réfère généralement à des systèmes de choses interconnectés (objets ou personnes inanimés), ou à des types spécialisés du concept mathématique des graphiques. Les approches théoriques des graphes pour la recherche d'informations peuvent être retracées aux premiers travaux de [15] sur la recherche d'informations sémantique, qui ont été suivies de plusieurs variantes de la recherche d'informations conceptuelle et de la recherche d'informations. De nombreuses variantes de formalismes graphiques ont depuis été utilisées dans les approches connexionnistes de la recherche d'informations. Plus récemment, les applications de la théorie des graphes ont été utilisées pour d'autres applications dans l'IR, par exemple les mesures d'évaluation IR [16] et le reclassement [17]

### 4. Représentation des connaissances

De nos jours, les entreprises sont établies dans un environnement caractérisé par des quantités croissantes de données, ce qui rend l'agrégation, la représentation et le raisonnement des connaissances très importants pour le traitement de ces données. Les techniques de créativité sont utiles pour acquérir des connaissances [18]. Le raisonnement humain et son

environnement étant incertains et imprécis, en particulier dans les contextes où la créativité est appliquée, la logique floue peut améliorer le processus d'agrégation et de représentation des connaissances. La théorie des graphes est un formalisme utile pour représenter les connaissances d'une manière compréhensible par ordinateur. Certains types de graphiques peuvent rendre compte de l'imprécision et de l'incertitude en introduisant un flou. Les applications basées sur des graphiques de connaissances doivent fonctionner de manière productive et riche en informations sémantiquement riches, mais bien organisées et obligées. Bien que les méthodes de démonstration sociale et les bases de données de graphiques soient des dispositifs utiles pour résoudre une partie

des problèmes particuliers, elles ne peuvent pas offrir un cadre spécialisé et calculé étendu pour l'ensemble de l'entreprise. Beaucoup se tournent plutôt vers les références du Web sémantique, avec le langage incontournable de l'ontologie Web (OWL), comme une «limace d'argent» revendiquée pour le défi d'administration des graphiques sémantiques. Néanmoins, aussi efficace que la pile du Web sémantique finisse par être en ce qui concerne la diffusion d'informations connectées sur le Web, son incitation en tant que réponse de représentation de diagramme d'apprentissage pour rester solitaire, les applications particulières à l'espace sont plus subtiles [19].

### 5. Intelligent artificiel et recherche :

Les ingénieurs logiciels utilisent le plus souvent des procédures de contrefaçon de conscience créées pour les cadres dynamiques. L'hypothèse du graphique est une représentation utile au motif que d'un point de vue, les composants du diagramme peuvent être caractérisés afin d'avoir une correspondance équilibrée avec les composants de nombreuses sortes de cadres de construction. En génie logiciel, les méthodes de recherche sont des systèmes qui recherchent des réponses à un problème dans un espace de chasse. Les arrangements ou «états objectifs» pouvaient de temps en temps être une question, un objectif, un sous-objectif ou un chemin vers la chose regardée. L'espace de poursuite peut être parlé par un "graphique" qui dépend de l'hypothèse du diagramme. La structure "graphique" ne doit pas être confondue avec les diagrammes que les utilisateurs peuvent avoir tirés de leurs cours d'arithmétique cohérents.

Une structure d'information "arborescente" pour PC est une sorte de diagramme inhabituel. Un arbre a un concentrateur racine sur le point le plus élevé de la structure et il a tout au plus un chemin vers chaque concentrateur. Chaque hub peut être associé à un niveau inférieur de voisins appelés hubs de jeunes (successeurs). Les concentrateurs qui n'ont pas d'enfants sont appelés concentrateurs foliaires. En utilisant le boîtier de clé automatique, la maison du propriétaire est caractérisée comme l'espace de chasse. Il a le moyeu "home" comme base d'un arbre. Le hub domestique dispose de trois hubs tyke (les hubs de la pièce) et d'autres hubs pour enfants, par exemple, des zones de travail et un tiroir dans les chambres. En conséquence, l'hypothèse du graphique est extrêmement utile pour rechercher dans un faux cadre vif [20] [21].

## 13. Conclusion :

Notre objectif principal dans ce premier chapitre est de parler sur la théorie des graphes en général qui nous permettent de manipuler plus facilement des objets et leurs relations avec une représentation graphique naturelle.

L'ensemble des techniques et outils mathématiques mis au point en Théorie des Graphes permettent de démontrer facilement des propriétés, d'en déduire des méthodes de résolution, des algorithmes, ...

Dans le prochain chapitre, on va parler sur l’intelligence artificielle et Apprentissage machine et apprentissage en profondeur.

**Points clés - Chapitre 01 :**

- Nous avons introduit le chapitre en définissant l'exploration de graphes.
- Nous avons donné quelques notions préliminaires sur les graphes et leur théorie.
- Où nous avons parlé dans le dernier chapitre de l'importance Théorie des graphes dans le développement et les utilisations dans le domaine de la science et de la technologie

# Chapitre II

# L‘intelligence artificielle (IA) Et Apprentissage machine et Apprentissage en profondeur

## 1. Introduction :

Depuis son point de départ dans les années 1950, notamment avec les travaux du mathématicien Alan TURING, l'intelligence artificielle (IA), discipline mathématique et technique destinée à reproduire l'intelligence humaine, s'est développée par cycles successifs parallèlement à la croissance de la puissance de calcul informatique disponible. Dans les années 1980, le concept d'apprentissage automatique («Machine Learning») se développe permettant à une machine de déduire une «règle à suivre » uniquement à partir de l'analyse de données. Cette période voit apparaître la majorité des algorithmes « apprenants » utilisés aujourd'hui (réseau de neurones, apprentissage par renforcement, machines à vecteurs de support, etc.). Ces avancées se concrétisent notamment par le succès de l'ordinateur Deep Blue face au grand maître d'échecs, Gary KASPAROV en 1992.À partir des années 2000, un nouveau cycle d'innovations en IA se met en place avec le formidable développement d'Internet et des très grandes infrastructures de calcul, offrant un accès à un volume de données encore jamais atteint dans l'histoire humaine. Avec cette capacité nouvelle, le développement des techniques d'apprentissage profond («Deep Learning») permet aux machines de commencer à surpasser les performances des meilleurs experts humains dans des domaines comme la reconnaissance visuelle, l'analyse documentaire ou la traduction.

Dans ce chapitre on va parler de quelques notions sur l'intelligence artificielle, apprentissage automatique (machine learning), apprentissage profond (deep learning) avec les dernières actualités dans ces domaines et les liens entre eux.


**Figure II.1 : La relation entre IA, ML et DL**

## 2. Intelligence artificielle :

L'intelligence artificielle (IA) correspond à un ensemble de technologies qui permet de simuler l'intelligence et d'accomplir automatiquement des tâches de perception, compréhension et prise de décision. Ces techniques font particulièrement appel à l'utilisation de l'informatique, de

l'électronique, des mathématiques (notamment statistiques), des neurosciences et des sciences cognitives. [1]

### 2.1.Les sous-domaines de l'IA :

#### 2.1.1.Représentation des connaissances et Raisonnement Automatique :

Comme son nom le suggère, cette branche de l'IA traite le problème de la représentation des connaissances (qui peuvent ˆêtre incomplètes, incertaines, ou incohérentes) et de la mise en œuvre du raisonnement.

#### 2.1.2.Résolution de problèmes généraux :

L'objectif est de créer des algorithmes généraux pour résoudre des problèmes concrets.

#### 2.1. 3.Traitement du langage naturel :

Ce sous-domaine vise `à la compréhension, la traduction, ou la production du langage (écrit ou parlé).

#### II.2.1.4.Vision artificielle :

Le but de cette discipline est de permettre aux ordinateurs de comprendre les images et la vidéo (par exemple, de reconnaitre des visages ou des chiffres).

#### II.2.1.5.Robotique :

Cette discipline vise  a réaliser des agents physiques qui peuvent agir dans le monde. [26]

### II.3. L'apprentissage automatique (Machine Learning):

L'apprentissage automatique est devenu l'un des sujets les plus importants au sein des organisations de développement qui cherchent des moyens novateurs de tirer parti des ressources de données pour aider l'entreprise à acquérir un nouveau niveau de compréhension. Pourquoi ajouter l'apprentissage automatique dans le mix ? Avec les modèles d'apprentissage automatique appropriés, les organisations ont la capacité de prédire continuellement les changements dans l'entreprise de sorte qu'ils sont le mieux en mesure de prédire ce qui est à venir. Comme les données sont constamment ajoutées, les modèles d'apprentissage automatique garantissent que la solution est constamment mise à jour. La valeur est simple : si vous utilisez les sources de données les plus appropriées et en constante évolution dans le contexte de l'apprentissage automatique, vous avez la possibilité de prédire l'avenir.

L'apprentissage automatique est une forme d'intelligence artificielle qui permet à un système d'apprendre des données plutôt que par une programmation explicite. Cependant, l'apprentissage automatique n'est pas un processus simple.

L'apprentissage automatique utilise divers algorithmes qui tirent des leçons itératives des données pour améliorer, décrire les données et prévoir les résultats. Comme les algorithmes ingèrent des données de formation, il est alors possible de produire des modèles plus précis basés sur ces données. Un modèle d'apprentissage automatique est la sortie générée lorsque vous entraînez votre algorithme d'apprentissage automatique avec des données. Après la formation, lorsque vous

fournissez un modèle avec une entrée, vous recevrez une sortie. Par exemple, un algorithme prédictif créera un modèle prédictif. Ensuite, lorsque vous fournirez des données au modèle prédictif, vous recevrez une prévision fondée sur les données qui ont formé le modèle. L'apprentissage automatique est maintenant essentiel pour créer des modèles analytiques. [2]

### II.3.1. Pourquoi L'apprentissage automatique ?

Supposons que vous souhaitiez écrire le programme de filtrage sans utiliser de méthodes d'apprentissage automatique. Dans ce cas, vous devrez suivre les étapes suivantes:

- Au début, vous examineriez à quoi ressemblent les spams. Vous pouvez les sélectionner pour les mots ou les expressions qu'ils utilisent, comme «carte de débit», «libre», etc., ainsi que pour les modèles utilisés dans le nom de l'expéditeur ou dans le corps du message.
- Deuxièmement, vous écririez un algorithme pour détecter les modèles que vous avez vus, puis le logiciel marquerait les e-mails en tant que spam si un certain nombre de ces modèles sont détectés.
- Enfin, vous testeriez le programme, puis renfaîtiez les deux premières étapes jusqu'à ce que les résultats soient suffisamment bons.

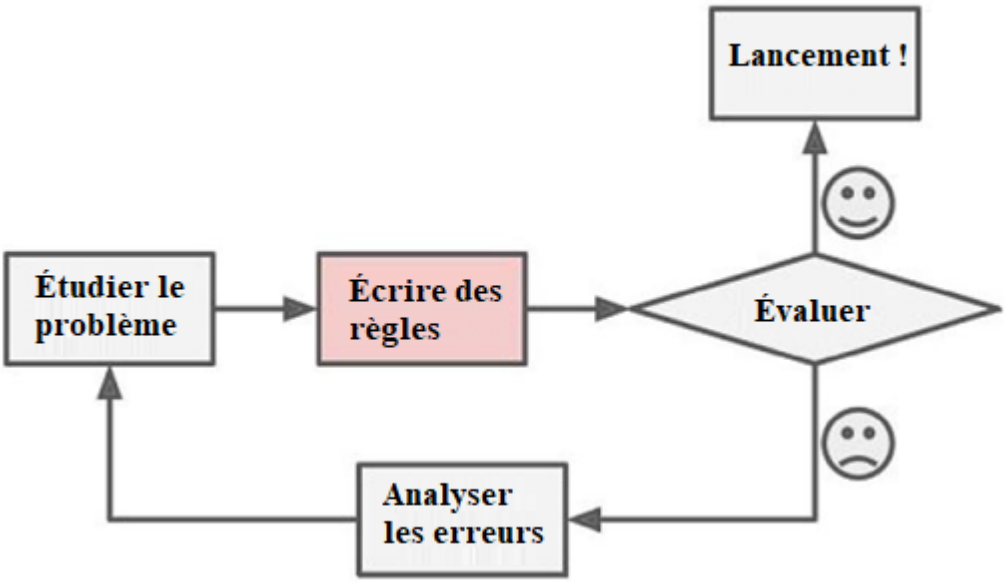



**Figure II.2 : L'approche traditionnelle.**

Comme le programme n'est pas un logiciel, il contient une très longue liste de règles difficiles à maintenir. Mais si vous avez développé le même logiciel en utilisant ML, vous pourrez le maintenir correctement. [3]

### II.3.2. Types de systèmes de L'apprentissage automatique :

Il existe différents types de systèmes d'apprentissage automatique. Nous pouvons les diviser en catégories, selon que

- Ils ont été formés avec des humains ou non humains :
  - Supervisé ;
  - Non supervisé ;
  - Semi-supervisé ;
  - Apprentissage par renforcement.
- S'ils peuvent apprendre progressivement.
- S'ils travaillent simplement en comparant de nouveaux points de données pour trouver des points de données, ou peuvent détecter de nouveaux modèles dans les données, ils créeront ensuite un modèle. [3]

### A. L'apprentissage supervisé :

Dans l'apprentissage supervisé (ou l'apprentissage avec un expert), les données sont des échantillons de motifs entrée-sortie.

Dans ce cas, une description concise des données est la fonction permettant de générer la sortie d'une entrée donnée. Ce problème est appelé apprentissage supervisé car les objets considérés sont associés déjà avec valeurs attendues (par ex classes et valeurs réelles).

Dans le problème d'apprentissage supervisé, un échantillon de couples entrée-sortie donné est appelé « échantillon d'entrainement »(ou ensemble d'entrainement), la tâche est de trouver une fonction déterministe qui mappe n'importe quelle entrée à une sortie qui est capable de prédire les futures observations entrée-sortie, et diminuer le possible les erreurs. Pour n'importe quelle valeur attendue d'un objet présenté dans l'échantillon d'entrainement elle retourne la valeur la plus apparue dans l'ensemble d'entrainement avec cet objet. [2]

On distingue l'apprentissage en classification ou régression selon le type de sorties.

**● Classification :**

Si l'espace de sorties n'a pas de structure sauf en cas d'égalité (ou pas) de deux éléments de sortie, ceci est appelé « le problème de classification d'apprentissage » (ou simplement classification). Chaque élément de l'espace de sortie est appelé « classe » (en anglais class).

L'algorithme qui résout le problème de classification est appelé « classificateur ». Dans les problèmes de classification la tâche est d'affecter de nouvelles entrées à un nombre de catégories ou classes discrètes. [2]

**Exemple de problème de classification :**

**Objectif :** Prédire qui gagne plus de 50k$ à partir de données de recensement. [4]

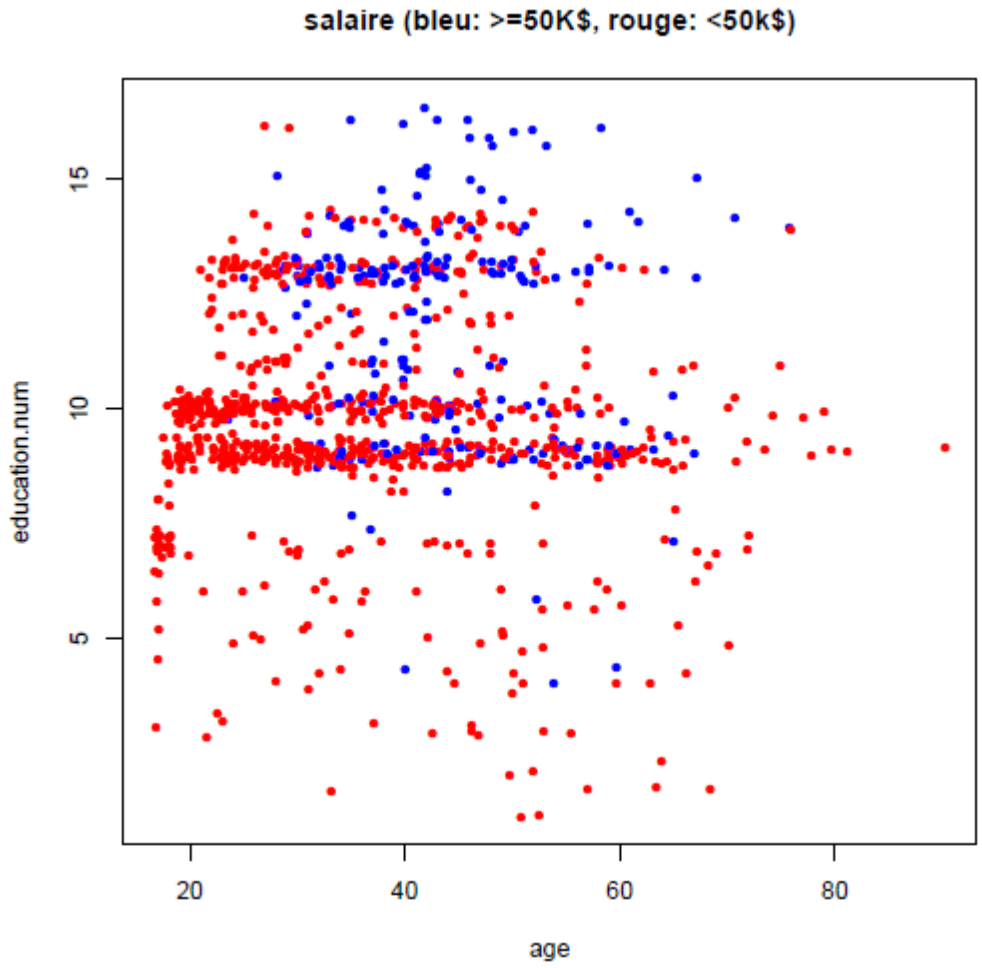



**Figure II.3 : Exemple de problème de classification.**

● **Régression :**

Si l'espace de sortie est formé par les valeurs des variables continues, pour un instant donné l'échange de stock s'indice sur quelques futures temps, la tache d'apprentissage est connue comme un problème de régression ou « fonction d'apprentissage ». Les exemples typiques de régression sont de prédire la valeur de partages dans le marché d'échange de stock et d'estimer la valeur d'une mesure physique dans une section d'un plat thermoélectrique. [2]

**Exemple de problème de régression :**

Prédire l'âge d'un ormeau (abalone) à partir de sa taille, son poids, etc. [4]

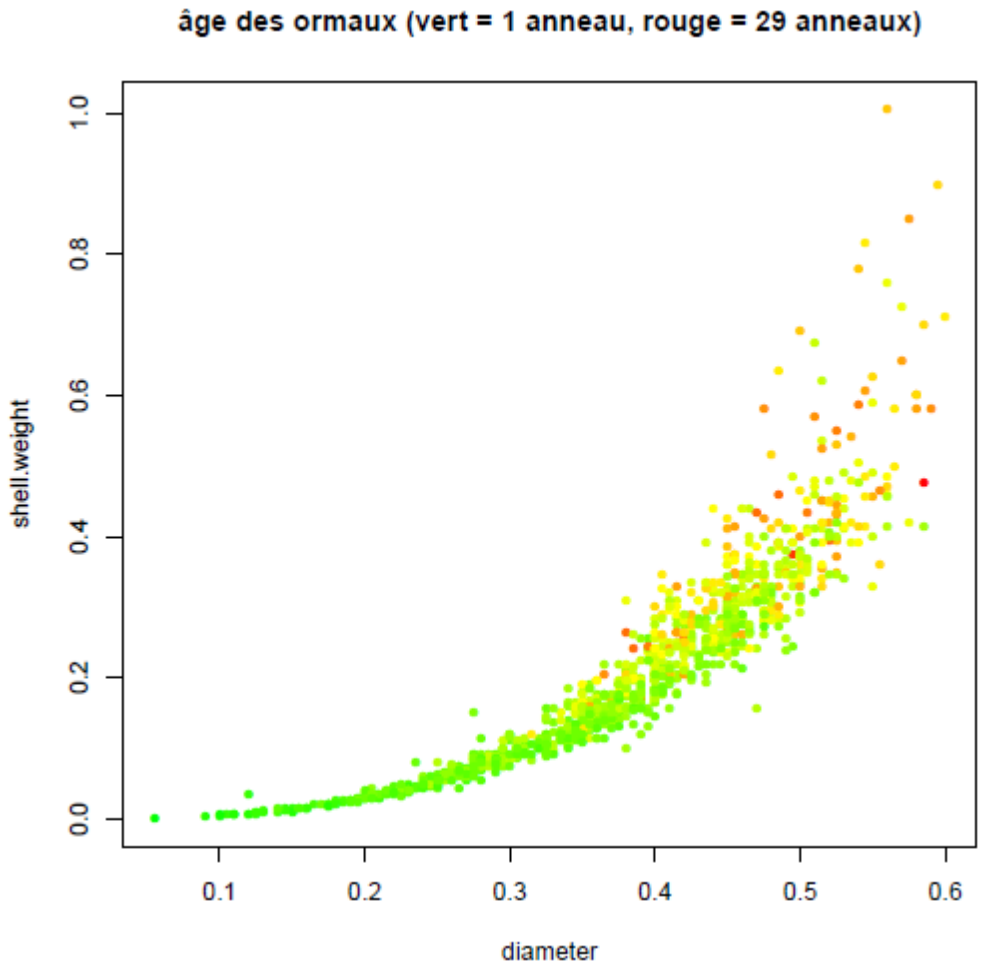



**Figure II.4 : Exemple de problème de régression.**

**B. L'apprentissage non supervisé :**

Dans l'apprentissage non supervisé il n'y a pas de notion de sortie désirée, on dispose seulement d'un nombre fini de données d'apprentissage, constituées ―d'entrées‖, sans qu'aucun label n'y soit rattaché. [5]

- **Estimation de densité :**

Dans un problème d'estimation de densité, on cherche à modéliser convenablement la distribution des données. L'estimateur obtenu **f^(x)** doit pouvoir donner un bon estimé de la densité de probabilité à un point de test x issu de la même distribution (inconnue) que les données d'apprentissage. [5]

Étant donné un ensemble de $N$ observations vectorielles $\mathbf{D}N=\{x1,\ldots,xN\}$ décrites par $d$ variables, donc $\mathbf{D}N \subset \mathbf{R}d$ , l'objectif général de l'estimation de densité est de trouver la fonction de densité de probabilité $f$ qui a généré l'échantillon aléatoire $\mathbf{D}N$ .

La figure suivante montre un exemple dans lequel les observations (représentées par les points bleus dans le plan horizontal) sont issues de $\mathbb{R}2$ et la densité qui a généré ces points est la surface courbe. Le sommet (en rouge) sur cette surface correspond à la région du plan horizontal où les observations (les points bleus) sont les plus denses. [6]

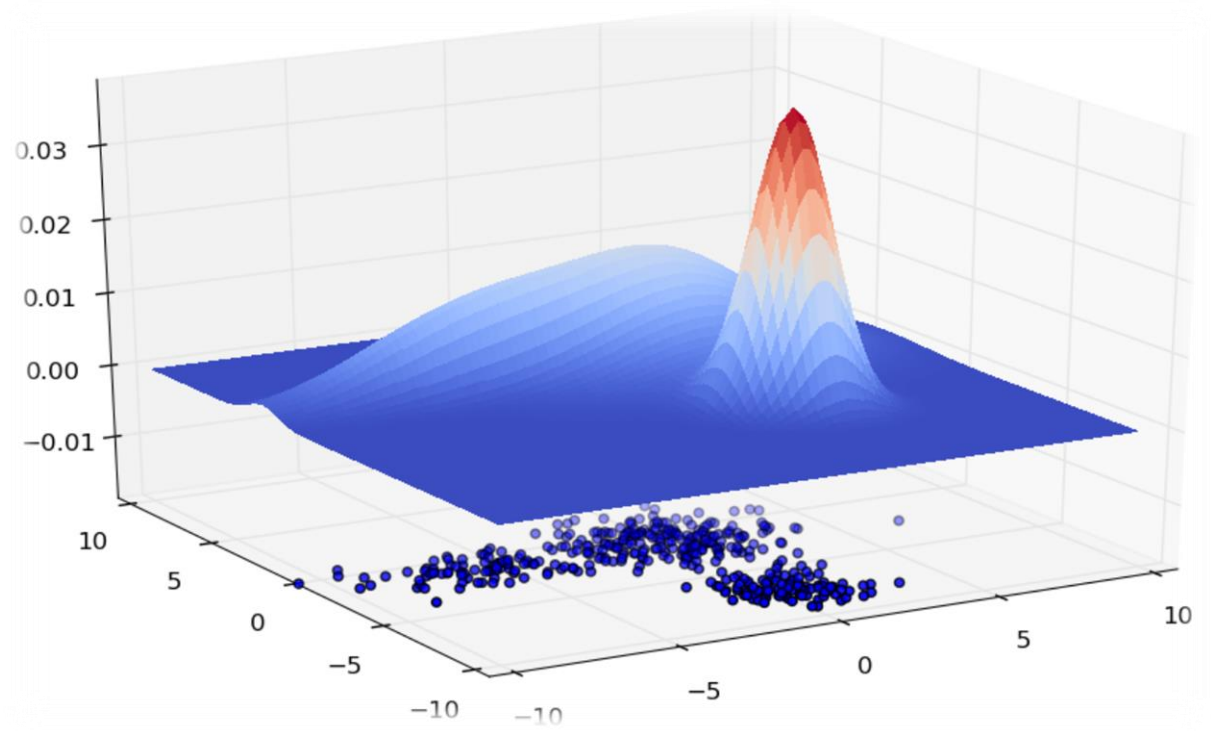



**Figure II.5 : Exemple de données de $\mathbb{R}2$ et de densité estimée.**

- **Partitionnement (Clustering) :**

Le problème du partitionnement est le pendant non-supervisé de la classification. Un algorithme de partitionnement tente de partitionner l'espace d'entrée en un certain nombre de classes en se basant sur un ensemble d'apprentissage fini, ne contenant aucune information de classe explicite.

Les critères utilisés pour décider si deux points devraient appartenir à la même classe ou à des classes différents sont spécifiques à chaque algorithme, mais sont très souvent liés à une mesure de distance entre points. [5]

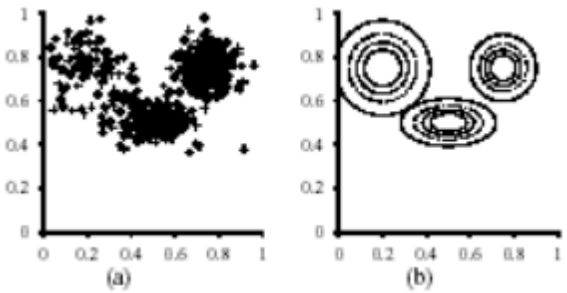



**Figure I.6: Clustering – Expectation maximization.**

- **Réduction de dimensionnalité :**

Le but d'un algorithme de réduction de dimensionnalité est de parvenir à résumer l'information présente dans les coordonnées d'un point en haute dimension ( $x \epsilon Rn, n\ grand$) , par un nombre plus réduit de caractéristiques. Le but espéré est de préserver l'information importante, de la mettre en évidence en la dissociant du bruit, et possiblement de révéler une structure sous-jacente qui ne serait pas immédiatement apparente dans les données d'origine en haute dimension. L'exemple le plus classique d'algorithme de réduction de dimensionnalité est l'Analyse en Composantes Principales (ACP). [7]

**Raisons pour réduire la dimensionnalité :**

- Malédiction de la dimensionnalité :

  - Ajout d'une dimension augmente exponentiellement l'espace mathématique.
  - 100 points équidistants de 0,01 en une dimension⇒1020 points nécessaires en 10 dimensions pour conserver la même densité.
  - Grande dimensionnalité : complexité élevée en calculs et en mémoire.

- Epargner des coûts de mesures.

- Plus un modèle est simple, moins il y a de variance.

- Plus facile d'expliquer avec moins de variables : extraction de connaissances.

- Visualiser les données : analyse de résultats. [7]

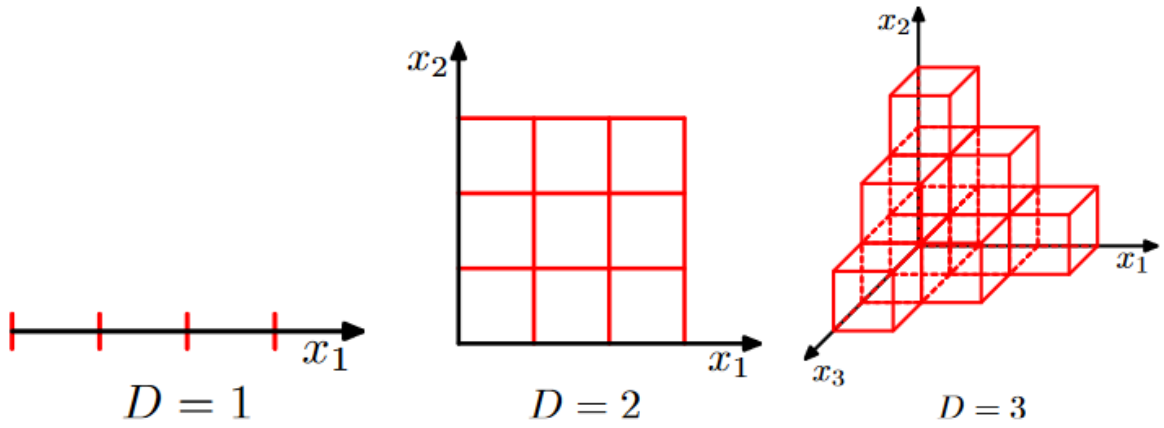



**Figure II.7 : Malédiction de la dimensionnalité.**

- **L'apprentissage par renforcement :**

Nous ne faisons ici que mentionner très succinctement le cadre général de l'apprentissage par renforcement, ce domaine étant hors du champ de notre sujet.

La particularité et la difficulté du cadre de l'apprentissage par renforcement est que les décisions prises par l'algorithme influent sur l'environnement et les observations futures.

L'exemple typique est celui d'un robot autonome qui évolue et effectue des actions dans un environnement totalement inconnu initialement. Il doit constamment apprendre de ses erreurs et succès passés, et décider de la meilleure politique à appliquer pour choisir sa prochaine action. [8]

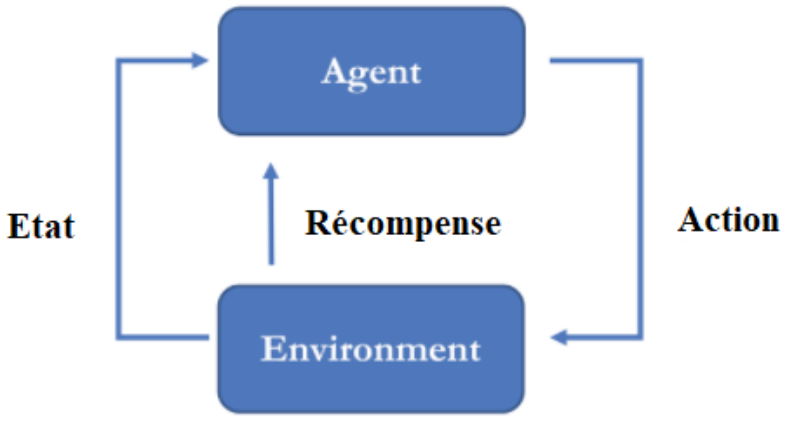



**Figure I.8 : Le cycle d'apprentissage par renforcement**

- **L'apprentissage par transfert :**

L'apprentissage par transfert fournit un cadre théorique et méthodologique intéressant permettant d'adapter un modèle appris à partir d'un domaine et d'une tâche source vers un nouveau domaine et/ou une nouvelle tâche cible. Le domaine fait référence à l'espace de représentation des données ainsi qu'à la répartition de ces données dans cet espace ; la tâche désigne à la fois l'espace des labels et la fonction de prédiction qui associe un label à un exemple de test. Par exemple, pour une tâche de classification d'expressions faciales, les exemples labélisés peuvent être des images de visage auxquelles on associe une catégorie d'émotion (joie, colère, tristesse...). Si ces images ont été capturées dans des conditions spécifiques(en intérieur avec une caméra haute résolution par exemple), le système appris sur le domaine source ne pourra certainement pas être appliqué directement sur une image de visage prise en extérieur par une webcam. Pour autant il serait intéressant de transférer le domaine source vers le domaine cible car il est couteux et parfois impossible d'annoter les données cibles. [9]

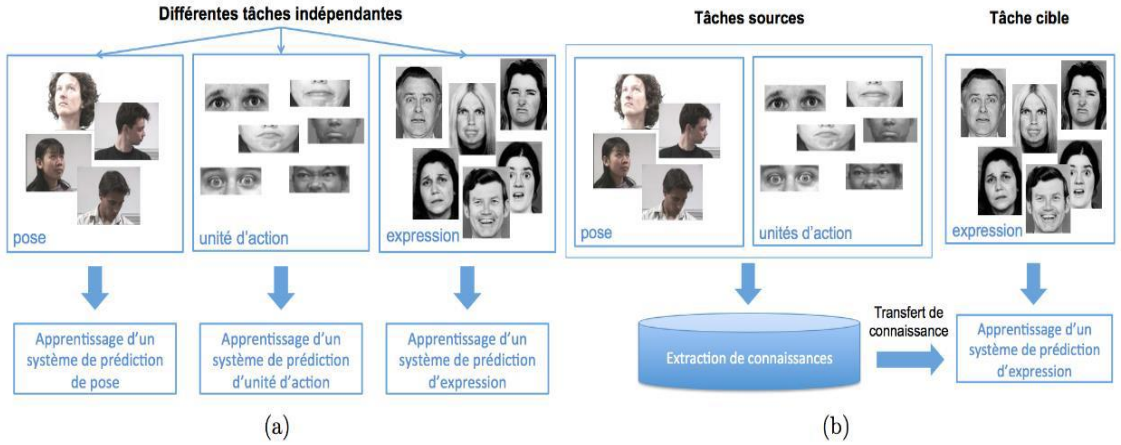



**Figure II.9: Principe général de (a) l'apprentissage automatique traditionnel et (b) l'apprentissage par transfert.**

Les domaines d'application du Transfer Learning sont nombreux. Principalement, les méthodes de transfert de connaissance sont très souvent utilisées pour la reconnaissance d'image ainsi que le traitement automatique du langage. Ces deux domaines d'apprentissage sont très complexes et chronophages. C'est pour cela que le Transfer Learning apporte un souffle nouveau pour tenter d'optimiser ces traitements en exploitant au maximum des modèles déjà entraînés. Nous allons voir ici plusieurs méthodes de Transfer Learning. [10]

## II.4. L'apprentissage profond (Deep Learning):

Le Deep Learning ou apprentissage profond est un type d'intelligence artificielle dérivé d'apprentissage automatique (Machine Learning) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées. [11]

### II.4.1. Fonctionnement du l'apprentissage profond :

Le Deep Learning s'appuie sur un réseau de neurones artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines voire de centaines de « couches » de neurones, chacune recevant

et interprétant les informations de la couche précédente. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit. [12]

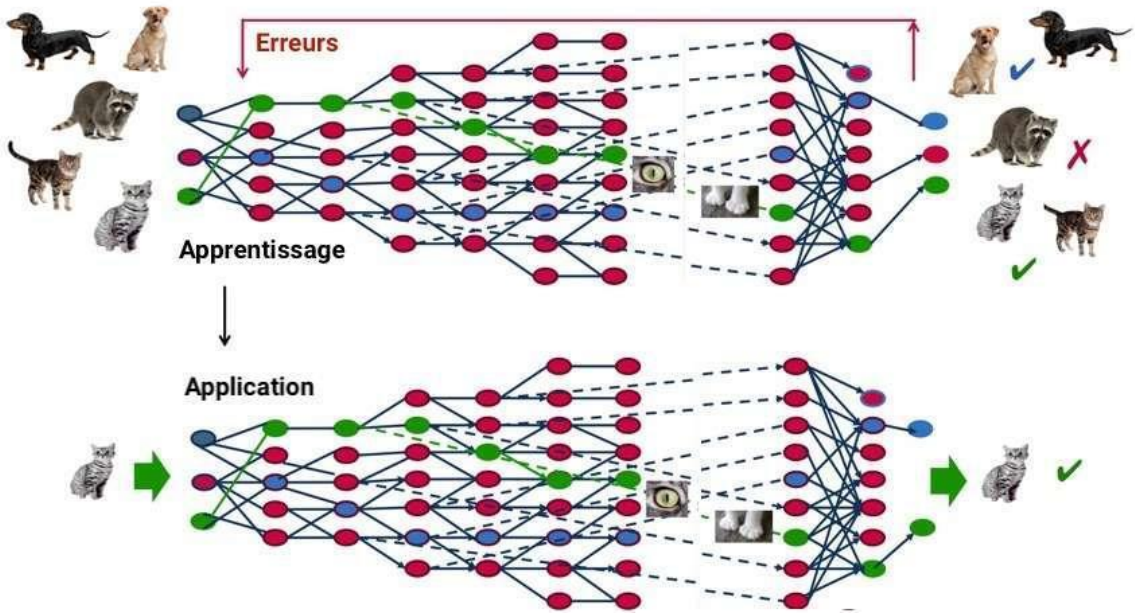



**Figure II.10 : Schéma de fonctionnement de Deep Learning .**

À travers un processus d'auto-apprentissage, le Deep Learning est capable d'identifier un chat sur une photo. À chaque couche du réseau neuronal correspond un aspect particulier de l'image.

À chaque étape, les « mauvaises » réponses sont éliminées et renvoyées vers les niveaux en amont pour ajuster le modèle mathématique. Au fur et à mesure, le programme réorganise les informations en blocs plus complexes. Lorsque ce modèle est par la suite appliqué à d'autres cas, il est normalement capable de reconnaître un chat sans que personne ne lui ait jamais indiqué qu'il n'a jamais appris le concept de chat. Les données de départ sont essentielles : plus le système accumule d'expériences différentes, plus il sera performant. [12]

### II.4.2. Applications de l'apprentissage profond :

Le Deep Learning est utilisé dans de nombreux domaines :

- Reconnaissance d'image, [13]
- Traduction automatique, [14]
- Voiture autonome, [15]
- Diagnostic médical, [16]
- Recommandations personnalisées, [17]
- Modération automatique des réseaux sociaux, [18]
- Prédiction financière et trading automatisé, [19]
- Identification de pièces défectueuses, [20]

- Détection de malwares ou de fraudes, [21]
- Chatbots (agents conversationnels), [22]
- Exploration spatiale, [23]
- Robots intelligents. [24]

L'apprentissage profond peut, par exemple, aider à :

- Mieux reconnaître des objets hautement déformables ;
- Analyser les émotions révélées par un visage photographié ou filmé ;
- Analyser les mouvements et position des doigts d'une main, ce qui peut être utile pour traduire les langues signées ;
- Améliorer le positionnement automatique d'une caméra, etc. ;
- Poser, dans certains cas, un diagnostic médical (ex. : reconnaissance automatique d'un cancer en imagerie médicale), ou de prospective ou de prédiction (ex. : prédiction des propriétés d'un sol filmé par un robot). [25]
- Reproduire une œuvre artistique à partir d'une photo à l'ordinateur. [25]

**II.5 .conclusion :**

a travers ce deuxieme chapitre, nous avons parlé sur intelligence artificielle et ses domaines d'application, ainsi l'apprentissage automatique en general et en particullier l'apprentissage en profonseur ou on a detaillé commet sa fonctionne et le domaine d'application.

Dans le chapitre prochain on va entamer directement les algorithme de recherche et ainsi comment les adapter dans le deeplearnig.

**Points clés :**

- Nous avons présenté dans ce chapitre intelligence artificiel
- On a présenté les types d'apprentissage automatique (supervisé, non supervisé)
- Nous avons également parlé de l'apprentissage en profondeur et de ses domaines d'application

# Chapitre 03 : Les algorithmes de recherche dans les graphes

## 1.Introduction :

Les problèmes de cheminement dans les graphes (en particulier la recherche d'un plus court chemin) comptent parmi les plus classiques de la théorie des graphes et les plus importants dans leurs applications. Le problème du plus court chemin (pcch) peut être posé de la façon suivante : étant donné un graphe G = (X, U), on associe `à chaque arc u = (i, j) un nombre réel, noté l(u) ou lij, appelé longueur de l'arc. Le problème du pcch entre deux sommets i0 et j0 du graphe consiste à déterminer, parmi tous les chemins allant de i0 `a j0 celui, noté µ ∗ dont la longueur totale : l (µ ∗) = X u∈µ∗ l(u) soit minimal.

Condition nécessaire Le problème du pcch à une solution si et seulement s'il n'existe pas dans le graphe de circuit de longueur strictement négative pouvant être atteint à partir de i0. Si cette condition nécessaire est vérifiée, il existe toujours un chemin de longueur minimale qui soit élémentaire. En effet, lorsque tous les circuits du graphe pouvant être atteint à partir de i0 ont une longueur strictement positive, tout pcch est nécessairement élémentaire. Lorsque l'on cherche un pcch entre deux sommets, on doit déterminer d'autres pcch entre i0 et d'autres sommets du graphe.

Aussi, les algorithmes existant se divisent en deux catégories : ceux dans lesquels on recherche un pcch d'un sommet spécifie i0 à tous les autres sommets du graphe ; ceux qui procèdent directement de la recherche de tous les pcch entre i et j pour tous les couples (i, j) du graphe. Il existe un grand nombre d'algorithmes permettant de d´déterminer un pcch d'un sommet particulier i0 à tous les autres sommets du graphe. Les plus efficaces d'entre eux réalisent un marquage des sommets c'est-à-dire qu'à chaque sommet i est associe une marque λ(i) représentant à la fin de l'algorithme la longueur du pcch allant de i0 `a i.

# 2.Plus courts chemins :

## 2.1d'origine fixée dans un graphe avec longueurs non négatives :

### 2.1.1Algorithme de Dijkstra :

Soit G = (X, U) un graphe dont les arcs sont munis de longueurs réelles positives ou nulles. On cherche les pcch de i0 à tous les autres sommets du graphe. L'algorithme de Moore-Dijkstra procède en n − 1 itérations. A l'initialisation, λ (i0) ← 0 et λ(i) ←

∞ pour tout i 6= i0. A une itération quelconque de l'algorithme, l'ensemble des sommets est partagé en deux sous-ensembles S et X\S

. Le sous-ensemble S contient l'ensemble des sommets d´définitivement marqués c'est-à-dire les sommets i pour lesquels la marque λ(i) représente effectivement la longueur d'un pcch allant de i0 à i (à l'initialisation S ← {i0}). X\S contient les sommets i ayant une marque provisoire vérifiant :

$\lambda(i) = \min_{k \in S \cap \Gamma^{-}(i)} \{\lambda(k) + lki\}$ L'algorithme est basé sur le lemme suivant

Lemme : Si i est un sommet de X\S de marque provisoire λ(i) minimale :

$\lambda(i) = \min_{j \in X \setminus S} \lambda(j)$

Alors λ(i) est la longueur d'un pcch chemin allant de i0 à i.

Ainsi à chaque itération, on sélectionne le sommet i de plus petite marque provisoire, on l'inclut dans l'ensemble S des sommets définitivement marqués et on remet à jour les marques de ces successeurs non définitivement marqués de la façon suivante :

$\forall j \in \Gamma + (i) \setminus S, \lambda(j) = \min \{\lambda(j); \lambda(i) + lij\}$

Lorsque tous les sommets du graphe sont dans l'ensemble S, les marques représentent les longueurs des pcch allant de i0 à tous les autres sommets du graphe.

Comme `à chaque itération, on attribue une marque d´définitive à un nouveau sommet, on d´détermine n − 1 pcch entre i0 et les autres sommets du graphe en au plus n − 1 itérations.

λ(i0) ← 0 ;

λ(i) ← +∞ ∀i ∈ X\{i0} ;

p(i) ← i ∀i ∈ X ; S ← {i0} ;

i ← i0 ;

Tant que (S 6= X) faire

Pour tout j ∈ Γ +(i)\S

faire Si (λ(j) > λ(i) + lij )

Alors λ(j) ← λ(i) + lij ; p(j) ← i;

Fin Si

Fin Pour

Sélectionner i ∈ X\S tel que λ (i) = minj∈X\Sλ (j) S ← S ∪

{i}; Fait

Preuve par récurrence de l'exactitude de l'algorithme :

Au rang 1 : λ (i0) = 0 = λ ∗ (i0)

Au rang 2 : Soit i ∈ Γ + (i0) tel que λ(i) = minj∈X\S {λ(j)}

λ(i) = li0i = λ ∗ (i) car tout autre chemin allant de i0 à i comportera au moins deux arcs et sera donc de valeur minimale : v = minj∈Γ+(i0)\{i} {λ(j) + q}.

Et, comme q ≥ 0, v ≥ λ(i). On suppose que la propriété est vraie au rang k − 1.

Au rang k : Soit i tel que $\lambda(i) = \min_{j \in X \setminus S}\{\lambda(j)\}$ Raisonnons par l'absurde et supposons qu'il existe un chemin de $\mu(i0, i) = \{i0, \ldots, s, r, \ldots, t, i\}$ de longueur $v(\mu(i0, i))$ tel que $v(\mu(i0, i)) < \lambda(i)$. Soit s le premier sommet appartenant à S rencontre en remontant le chemin μ de i vers i0. Comme la propriété est supposée vraie au rang k − 1, on a

$\lambda(s) = \lambda * (s)$ et :

$\lambda * (s) + lsr \leq v(\mu(i0, s)) + lsr \leq v(\mu(i0, s)) + lsr + \ldots + lti = v(\mu(i0, i))$

Car $\forall (i, j) \in U, lij \geq 0$. Par hypothèse, on a: $v(\mu(i0, i)) < \lambda(i)$. Or, $\lambda(r) \leq \lambda * (s) + lsr$

$< \lambda(i)$, c'est donc le sommet non marquer qui aurait dû être sélectionné à l'itération k et non le sommet i. On aboutit donc à une contradiction.

Complexité : A chaque itération, on sélectionne le sommet j de plus petite marque en O(n) opérations dans le pire cas, et on remet à jour les marques des successeurs de j en O (d +(j)) opérations. En tout, il y a n itérations pour marquer tous les sommets du graphe. La complexité totale est donc en O (n 2) + O(m) ≈ O (n 2).

Remarquons que lorsque le graphe est peu dense, le terme O (n 2) l'emporte sur O(m) et l'opération la plus coûteuse consiste à recherche le sommet de plus petite marque. Pour diminuer la complexité de cette recherche, on peut utiliser une structure de données appelée tas.

### 2.1.2Algorithme de Johnson :

Les sommets i de G et leur marque λ(i) sont stockées dans un maximier ce qui permet de trouver le sommet de marque initiale en O(1) opération. Dans l'algorithme de Johnson, on fait appel aux fonctions suivantes :

– insere (i, T) : insère le sommet i dans le tas T

– min(T) : retourne et supprime le sommet racine (de plus petite valuation) de T et recompose la structure

– diminue (valeur, i, T) : modifie la valuation du sommet i dans T (elle passe à valeur) et recompose la structure.

Le tas est constitué des sommets ayant des marques temporaires finies.

T ← Ø ;

λ(i0) ← 0 ; λ(i) ← +∞ ∀i ∈ X\{i0} ;

p(i) ← i ∀i ∈ X ; insere(i, T) ;

Tant que (S 6= X) faire

i ← min(T) ; S ← S ∪ {i} ;

Pour tout j ∈ Γ +(i)\S faire v = λ(i) + lij ;

Si (λ(j) > v) Alors

Si (λ(j) = +∞) Alors λ(j) ← v ;

p(j) ← i;

insere(j, T) ;

Sinon

λ(j) ← v ;

p(j) ← i; diminue(v, j, T) ;

Fin Si Fin Si

Fin Pour

Fait

A chaque itération, on sélectionne le sommet i de plus petite marque et on recompose le tas :

La complexité est en O(1) + O (log2n). Puis, on remet à jour les marques de tous les successeurs du sommet i sélectionné en O (d +(i) × log2n). Comme il y a n itérations, on a une complexité totale de O (nlog2n) + O (mlog2n) ≈ O (mlog2n).

## 2.2. d'origine fixée dans un graphe sans circuit avec longueurs quelconques :

### 2.2.1.Algorithme de Bellman :

Soit G = (X, U) un graphe sans circuit dont les arcs sont munis de longueurs réelles quelconques. On cherche les pcch allant de i0 à tous les autres sommets du graphe. Condition d'optimalité : Un ensemble de valeurs $\lambda * (i)$ pour i = 1, . . . , n, avec $\lambda * (i0) = 0$, représente les longueurs des pcch allant de i0 aux autres sommets du graphe si et seulement si :

$\forall (i, j) \in U : \lambda * (j) \leq \lambda * (i) + lij$

Lorsque le graphe ne présente pas de circuit, il faut déterminer une numérotation des sommets allant de 1 à n qui constitue un ordre topologique (le sommet de départ i0 ayant le numéro 1) et marquer les sommets dans cet ordre.

$\lambda(1) \leftarrow 0$ ; p(1) ← 1 ;

Pour j de 2 à n faire

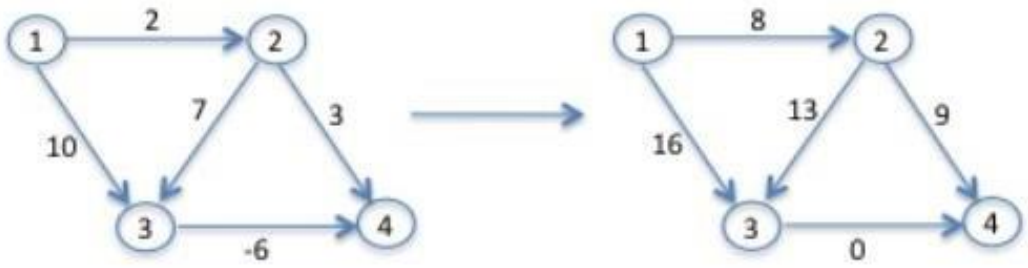
$\lambda(j) = \min_{i \in \Gamma^-(j)} \lambda(i) + lij$ p(j) =

$\text{argmin}_{i \in \Gamma^-(j)} \lambda(i) + lij$

## 2.3. d'origine fixée dans un graphe avec longueurs quelconques :

Soit G = (X, U) un graphe dont les arcs sont munis de longueurs réelles quelconques. On cherche les pcch de i0 à tous les autres sommets du graphe.

Remarque préliminaire : Soit un graphe G présentant sur certains arcs une valuation négative et, envisageons de rendre les valuations toutes positives ou nulles en ajoutant à chaque valeur la valeur absolue maximale des valeurs n´négatives.




Ceci change la nature du problème puisque, comme on le voit dans l'exemple ci- dessus, le plus court chemin allant de 1 à 4 dans le graphe initial (de valeur 3) n'est plus optimal dans le graphe transformé.

### 2.3.1.Algorithme de Ford (1956) :

Dans l'algorithme de Ford, les marques λ(i) des sommets sont modifiées itérativement de façon à converger vers la condition d'optimalité présentée en section 3

L'algorithme consiste alors, à partir d'un ensemble de valeurs provisoires, à parcourir séquentiellement la liste des arcs du graphe de façon à vérifier que la condition d'optimalité est satisfaite. Si cette condition est satisfaite pour tous les arcs (i, j), le problème est résolu ; sinon, il existe un arc (i, j) tel que : $\lambda(j) > \lambda(i) + l_{ij}$ et , dans ce cas, on remet à jour la marque de j de la façon suivante : $\lambda(j) \leftarrow \lambda(i)+l_{ij}$ (ce qui signifie que l'on améliore la marque provisoire de j en empruntant le chemin de longueur λ(i) entre i0 et i suivi de l'arc (i, j)).

λ (i0) ← 0 ;

λ(j) ← +∞ ∀i ∈ X/{i0} ;

Tant que (il existe (i, j) ∈U tel que λ(j) > λ(i) + lij ) faire λ(j) ← λ(i)

+ lij ; pj ← i

Fait

### 2.3.2.Algorithme de Ford-Bellman :

Cet algorithme est une variante ”optimisée” de l’algorithme de Ford. Pour parcourir la liste des arcs du graphe, on regarde pour chaque sommet i l’ensemble de ses prédécesseurs. Et, à une itération k donnée, on ne va pas s’intéresser à tous les sommets du graphe mais seulement à ceux dont la marque a été modifiée au cours de l’itération précédente. L’algorithme calcule donc à chaque itération k un ensemble de marques, notées λ k j pour tout j ∈ X. On note M = n j ∈ X | λ k j < λk−1 j o, l’ensemble des sommets dont les marques ont ´été modifiées à l’itération k, et seuls les sommets appartenant à Γ+(M) peuvent voir leurs marques modifiées au cours de l’itération k + 1. En fait les marques λ k j ont une interprétation très précise : c’est la valeur du meilleur chemin de i0 à j ne contenant pas plus de k arcs. Ainsi, en l’absence de circuit absorbant dans le graphe, l’algorithme termine nécessairement à l’issue de l’itération n car tout chemin élémentaire a une longueur maximale égale à

n − 1.

Si une ou plusieurs marques sont modifiées à l’itération n, cela signifie que le graphe présente un circuit de valeur négative (car il existe un pcch de longueur > n − 1 qui emprunte donc nécessairement un circuit de valeur < 0 !). L’algorithme de Ford- Bellman peut donc s’écrire comme suit :

```
k ← 0 ;

λ 0 (i0) ← 0

λ 0 (i) ← +∞ ∀i ∈ X/{i0} ;
p(i) ← i0 ∀i ∈ X/{i0} ;

M ← {i0} ;

Tant que (k ≤ n − 1 et M 6= Ø) faire
k ← k + 1 ; M0 ← Ø ;
```

Si (λ k (j) < λk−1 (j)) Alors M0

← M0 ∪ {j} ;

p(j) ← i ∗ avec i ∗ ∈ Γ −(j) ∩ M tel que λ k (j) = λ k−1 (i ∗) + li ∗j ; Fin Si

Fin Pour

M ← M0 ;

Fait

**Complexité :** A chaque itération, on remet à jour, dans le pire cas, les marques de tous les sommets en regardant l'ensemble des arêtes : O(m) opérations. Le nombre maximal d'itérations étant n, la complexité totale est en O (nm).

## 2.4.Plus courts chemins entre toutes les paires de sommets :

### 2.4.1.Algorithme de Floyd :

Soit la matrice A = {aij} de taille n × n avec :

aij = {0 si i = j

lij si (i, j) ∈ U

+∞ sinon}

L'algorithme de Floyd permet de calculer les pcch entre tous les couples de sommets de la façon suivante. A la première itération, on cherche le pcch entre chaque couple (i, j) passant éventuellement par le sommet 1 ; à l'itération l (avec l > 1), on cherche le pcch entre chaque couple (i, j) passant par des sommets d'indice inférieur ou égal à l. Une description formelle de l'algorithme est donnée ci-dessous.

```
Pour l de 1 à n faire
   Pour i de 1 à n faire
      Pour j de 1 à n faire aij = min {aij, ail + alj} ; Fin
      Pour
   Fin Pour
Fin Pour
```

La complexité totale est en O (n 3).

## 3.Algorithmes de parcours d'un graphe :

un parcours de graphe est un algorithme consistant à explorer les sommets d'un graphe de proche en proche à partir d'un sommet initial. Un cas particulier important est le parcours d'arbre.

Le mot parcours est également utilisé dans un sens différent, comme synonyme de chemin (un parcours fermé étant un circuit).

Il y a deux stratégies de parcours différentes : partant d'un sommet, le graphe est parcouru :

◦ en largeur

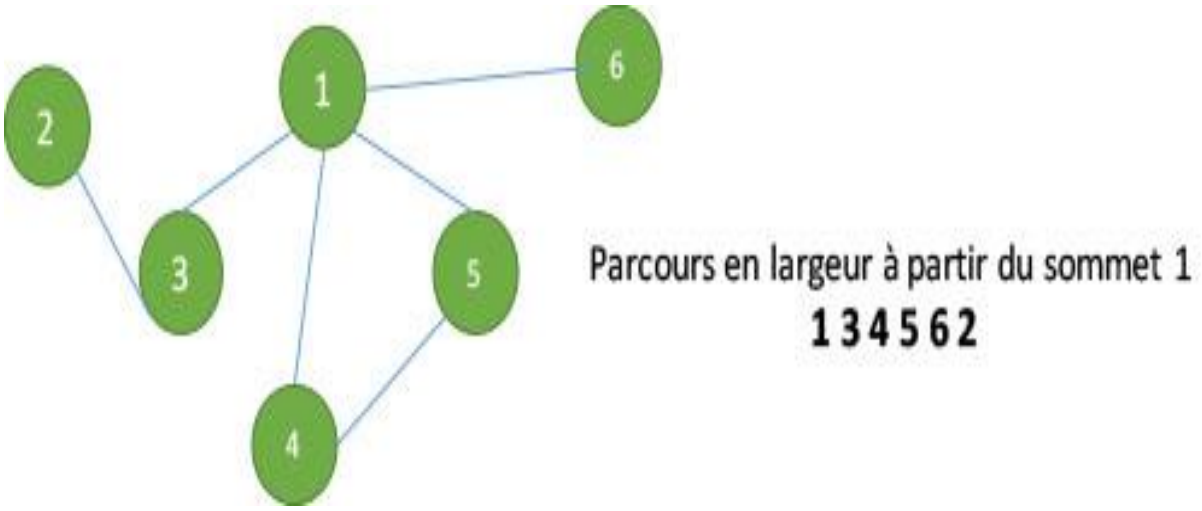
◦ en profondeur

### 3.1.Algorithme de parcours en largeur (ou BFS : Breadth First Search):

**Principe :**

Le parcours en largeur d'un graphe est similaire au parcours en largeur d'un arbre. À la différence des arbres, les graphes peuvent contenir des cycles, ce qui nous permet de revenir au même nœud. Pour éviter de traiter un nœud plusieurs fois, nous utilisons un tableau booléen visited.

Dans un parcours en largeur, tous les noeuds à une profondeur i doivent avoir été visités avant que le premier noeud à la profondeur i+1 ne soit visité. Un tel parcours nécessite l'utilisation d'une file d'attente pour se souvenir des branches qui restent à visiter.[5]




Étapes de l'algorithme :

Mettre le nœud de départ dans la file.

Retirer le nœud du début de la file pour l'examiner.

Mettre tous les voisins non examinés dans la file (à la fin).

Si la file n'est pas vide reprendre à l'étape 2.

**Algorithme :**

```
BFS(graphe G, sommet s):
{
  f = CreerFile();
  Marquer(s);
  Enfiler(f, s);
  TANT-QUE NON FileVide(f) FAIRE
     x = Défiler(f);
     Afficher(x)
     TANT-QUE ExisteFils(x) FAIRE
```

```
        z = FilsSuivant(x);
        SI NonMarqué(z) ALORS
           Marquer(z);
           Enfiler(f, z);
        FIN-SI
     FIN-TANT-QUE
  FIN-TANT-QUE
}
```

### 3.2.Algorithme de parcours en profondeur (ou DFS : Depth First Search) :

Le DFS est la méthode la plus simple pour parcourir un graphe. Elle fonctionne sur tout type de graphe, cyclique ou non,orienté ou non, etc.

En langage naturel ça donne : je pars d'un endroit que je ne connais pas, je me dirige vers d'autres endroits que je ne connais pas, et quand je suis bloqué je fais demi-tour jusqu'à retrouver un chemin que je n'ai pas encore parcouru.

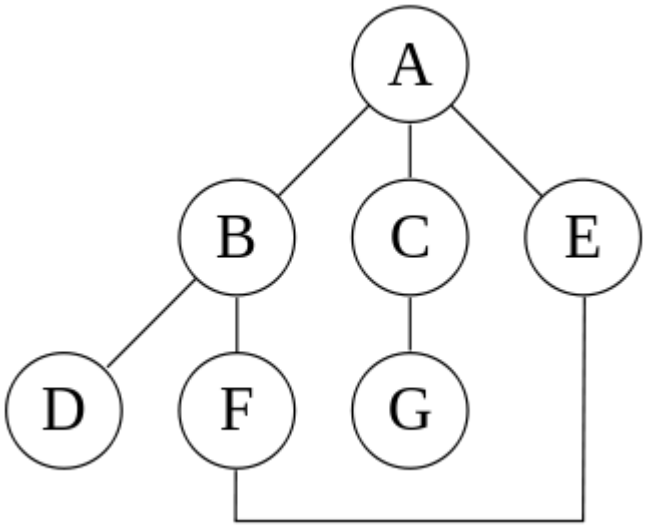
**Principe :**

On part du sommet r Lorsque l'on arrive à un sommet x, on ne l'explore pas, on visite un de ses voisins non encore visité, on cherche donc à prolonger le chemin de r à x.[6]

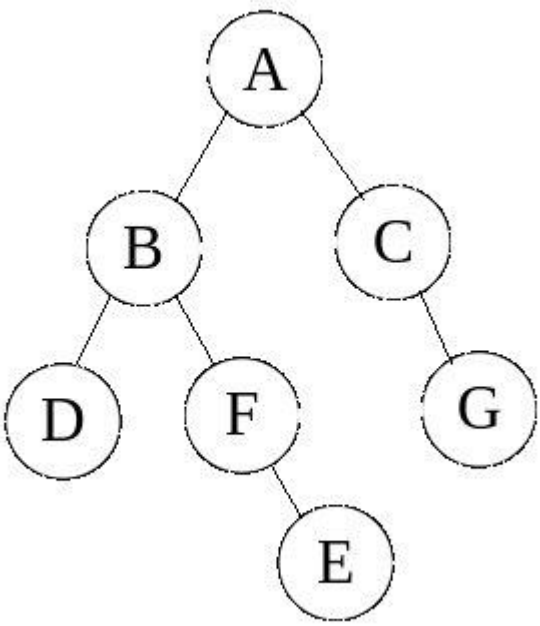
La gestion des sommets est réalisable au moyen d'une structure de données appelée pile.

l'algorithme descend en profondeur dans le graphe, avant de faire demi-tour. Sur le graphe ci-dessous l'ordre d'exploration pourra donc être :

A - B - D - F - E - C – G . [7]



Voyons le problème sous un autre angle, et dessinons les arêtes empruntées par le DFS :




**algorithme :**

→ Procédure PROFONDEUR (G : graphe, r : sommet)
pour i = 1 à n faire
Marque[i] ← faux
fin pour
h ← 1 (hauteur de pile)
Pile[h] ← r
tantque h > 0 faire
x ← Pile[h]
si PS[x] 6=/= 0 alors
y ← LS[PS[x]]
si Marque[y]=faux alors
Marque[y]=vrai
h ← h+1
Pile[h] ← y
finsi
Mettre à jour PS[x]
sinon
h ← h-1
finsi
fin tantque

## 4.Algorithmes d'arbres couvrants de poids minimum (Minimum spanning tree) :

Le problème est de connecter par un réseau (de câbles) un ensemble de points; on connaît les distances entre tous les couples de points entre lesquels on peut installer un câble, et on veut minimiser la longueur totale du câble qui doit être utilisé. On modélise le problème par un graphe non orienté G=(S,A), muni d'une fonction de coût c :A→N. Une solution du problème est un sous-graphe G'=(S,A') connexe sans circuit, c'est-à-dire un arbre, de coût minimum, ce coût étant $C(G')= \sum_{a\epsilon A'} c(a)$ ; une telle solution est appelée *un arbre couvrant de coût minimum* (voir figure 4) . On suppose que le problème a une solution, c'est-à-dire que le graphe G est connexe. Il existe plusieurs algorithmes pour résoudre

ce problème. Ce sont des algorithmes gloutons qui construisent la solution de façon incrémentale en ajoutant une arête à chaque étape.[8]

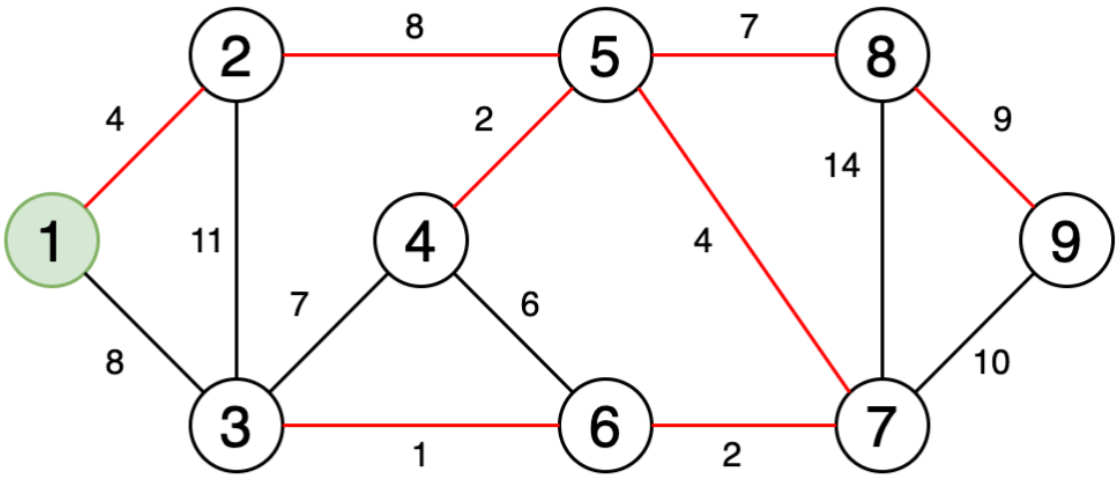



**Figure III-1: Graphe montrant un arbre couvrant minimum**

### 4.1-Algorithme Kruskal :

L'algorithme de Kruskal, est une implémentation directe de la propriété précédente. On construit un arbre, initialement vide, en choisissant à chaque étape l'arête de poids minimum reliant deux sommets non déjà reliés. Au départ, tous les sommets sont dans des composantes différentes. Si l'arête libre de poids minimum relie deux sommets dans des composantes distinctes, on la choisit et on fusionne les deux composantes auxquelles appartiennent ses extrémités. Sinon, on l'écarte et on examine l'arête de poids immédiatement supérieur.[9]

**Exemple de l'algorithme de Kruskal :**

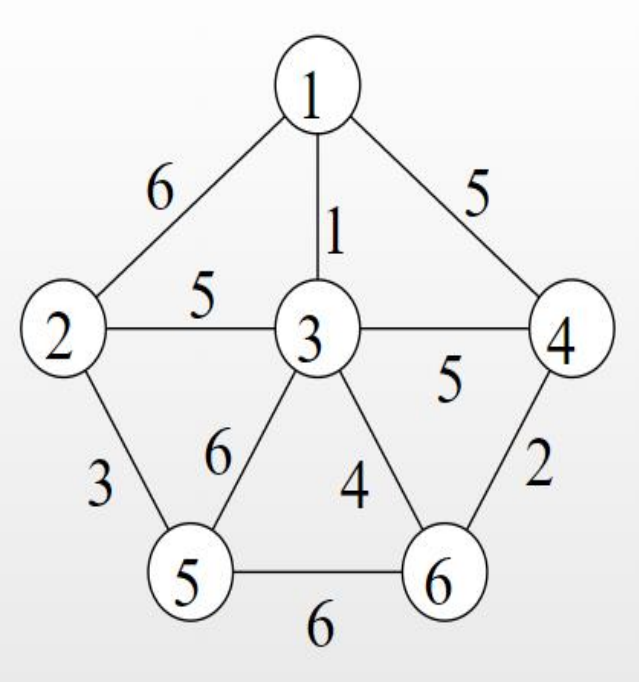



**Figure III.2-Graphe de départ**

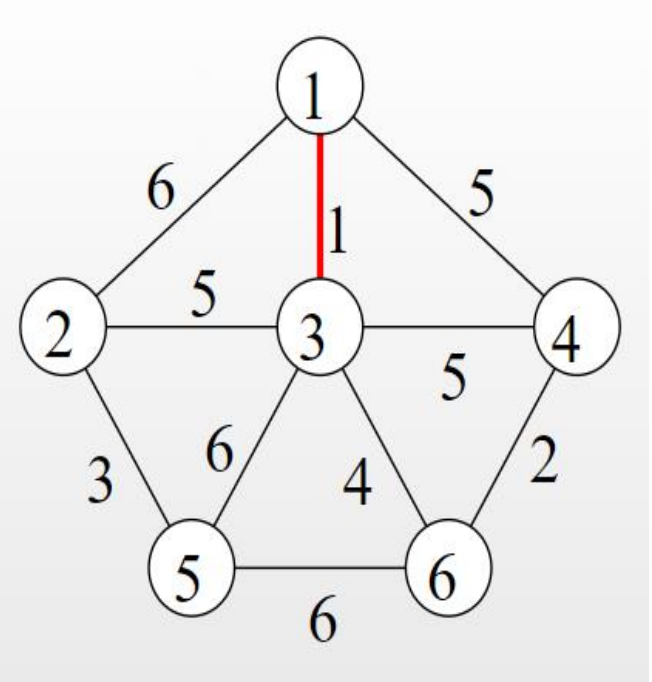



**Figure III.3- Itération 1**

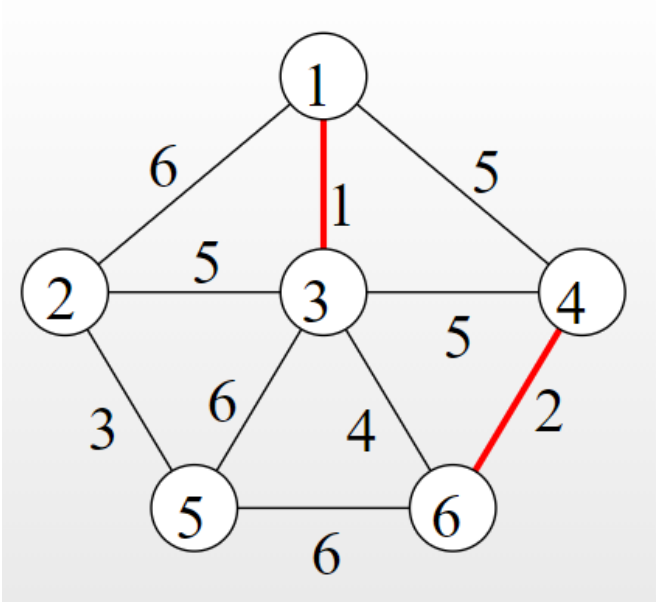



**figure III.3- Itération 2**

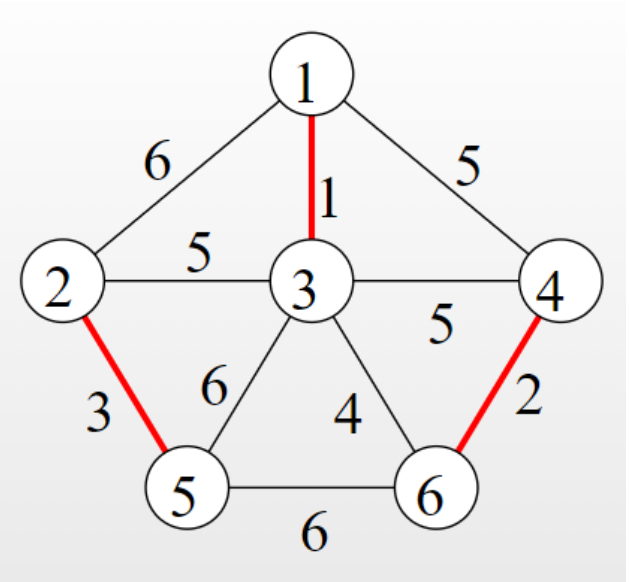



**Figure III.3- Itération 3**

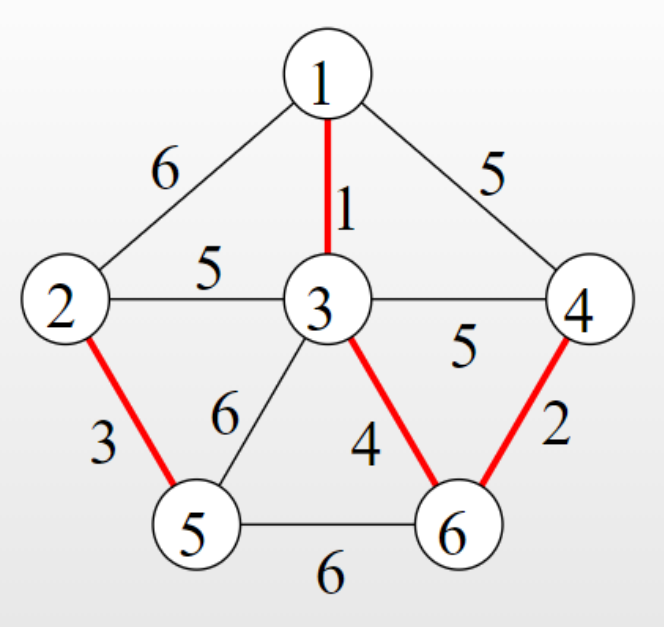



**Figure III.3- Itération 4**

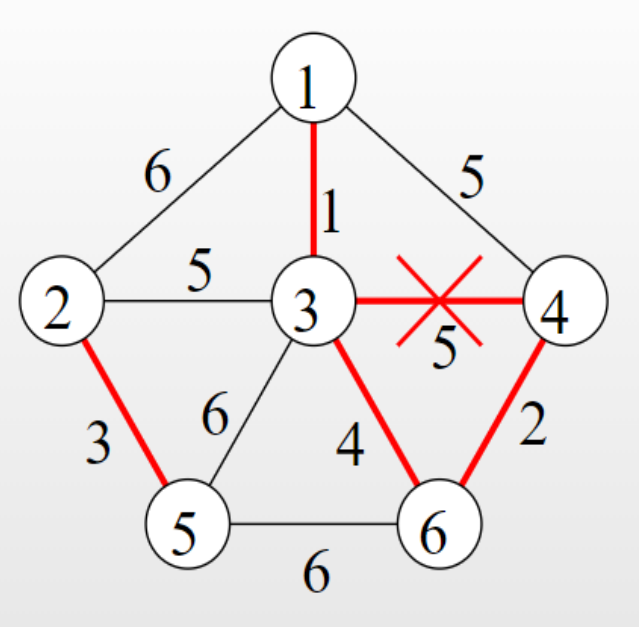



**Figure III.7- Itération 5 détection d'un cycle**

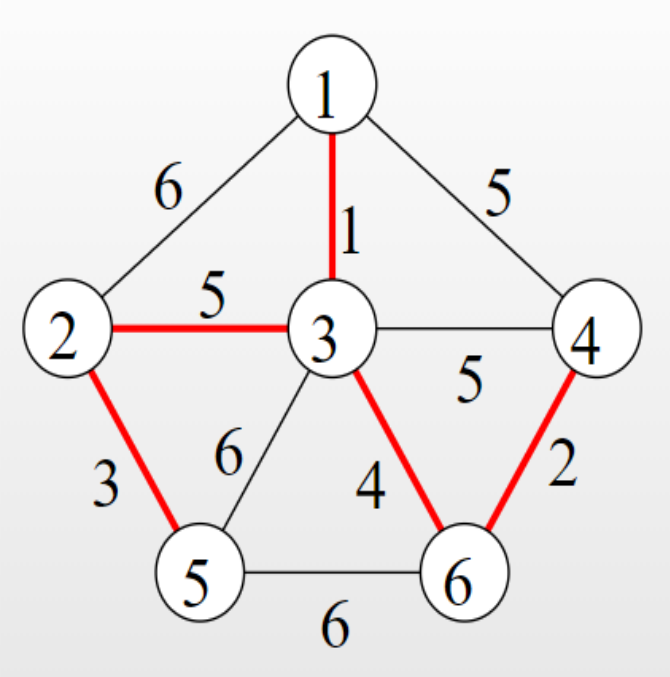



**Figure III.8- Itération 6**

**algorithme :**

```
kruskal(ARM) :-
  findall(
    Poids-(P1,P2)
  ,
    arc(P1,P2,Poids)
  ,
    Arcs
  ),
  keysort(Arcs, ArcsTries),
  krusk(ArcsTries, [], ARM).
krusk([], _, []) :-
  !.
krusk([_-(P1,P2)|Q], Visite, ARM) :-
  memberchk(P1, Visite),
  memberchk(P2, Visite),
  !,
  krusk(Q, Visite, ARM).
krusk([_-(P1,P2)|Q], Visite, [P1-P2|ARM]) :-
  union([P1,P2], Visite, Visite2),
  krusk(Q, Visite2, ARM).
```

### 4.2-Algorithme de Prim :

**L'algorithme de Prim applique ce principe :**

l'ensemble T est initialisé avec un sommet ; puis, à chaque étape, on ajoute à cet ensemble le sommet de S − T qui y est relié par l'arête de poids minimum ; de cette façon, on ne provoque pas de cycle et, comme le graphe est connexe, tous les sommets sont atteints. À chaque étape de l'itération, on cherche si le sommet qu'on vient de rajouter est relié par une arête de poids plus faible aux sommets non encore ajoutés. L'algorithme ressemble à celui de Dijkstra. La différence est dans la fonction d'actualisation. P[i, j] est le poids de l'arête de ij et ∞ si i et j ne sont pas reliés. On initialise S avec le sommet 1 et on note D[i] le poids

minimum d'une arête reliant S au sommet i. On choisit le sommet t qui a le D[t] minimum, puis on réévalue les D[i] des successeurs de t. 2.4 L'algorithme de Prim.[9]

complexité : Sa complexité est de O(mn) si le graphe est codé en liste ou matrice d'adjacence. Elle peut être abaissée en O(m + n log n) en le codant `a l'aide de tes de Fibonacci.[10]

Exemple de l'algorithme de prim :

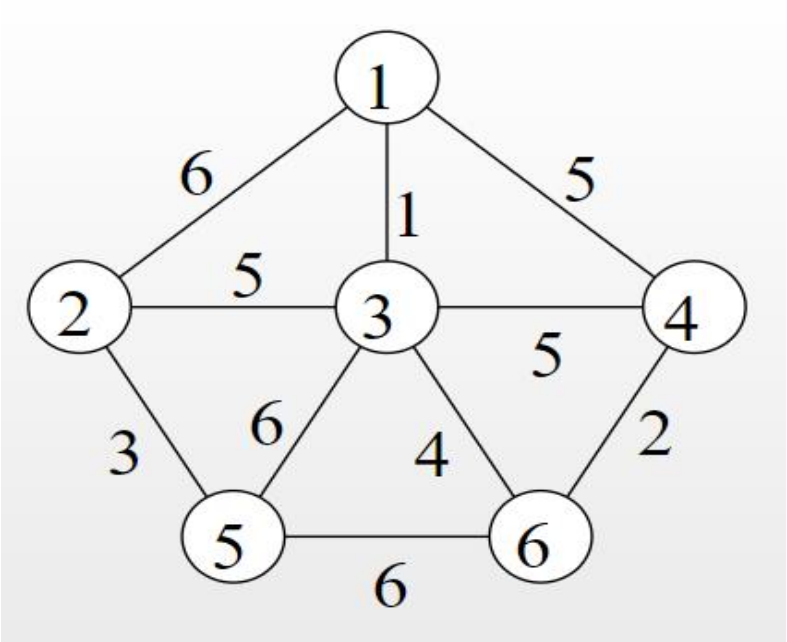



**Figure III.9- Graphe de départ**

**.**On part d'un arbre initial réduit à un seul sommet du graphe

. À chaque itération, on agrandit l'arbre en lui ajoutant le sommet libre accessible de plus petit poids possible

. On stoppe quand l'arbre est recouvrant.

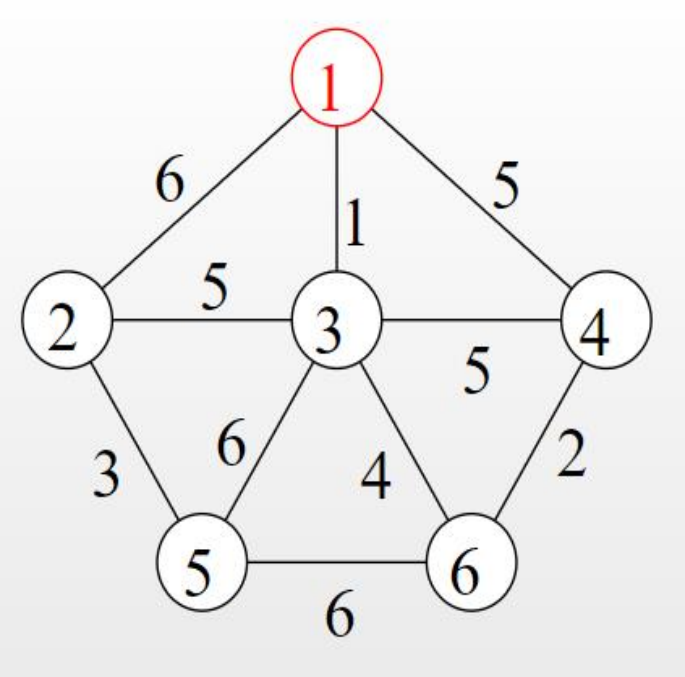



**Figure III.- 10 On part d'un sommet quelconque**

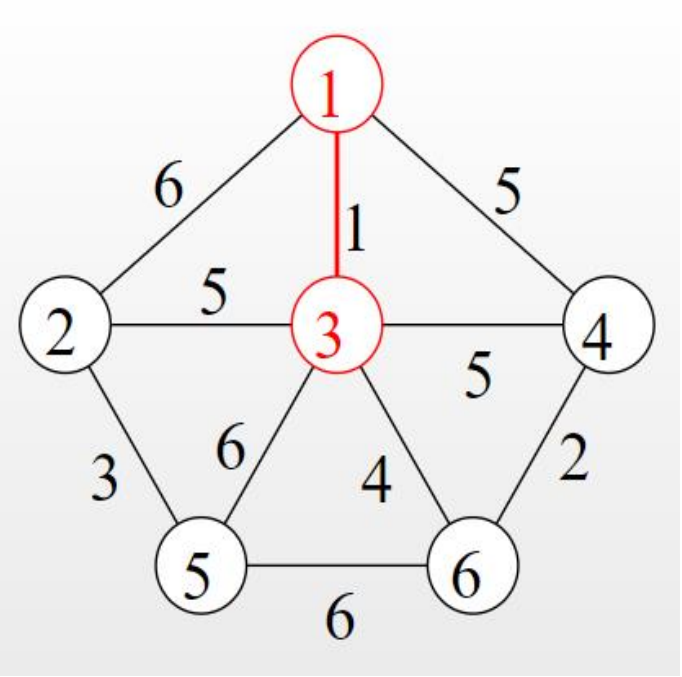



**Figure III.-11 Itération 1**

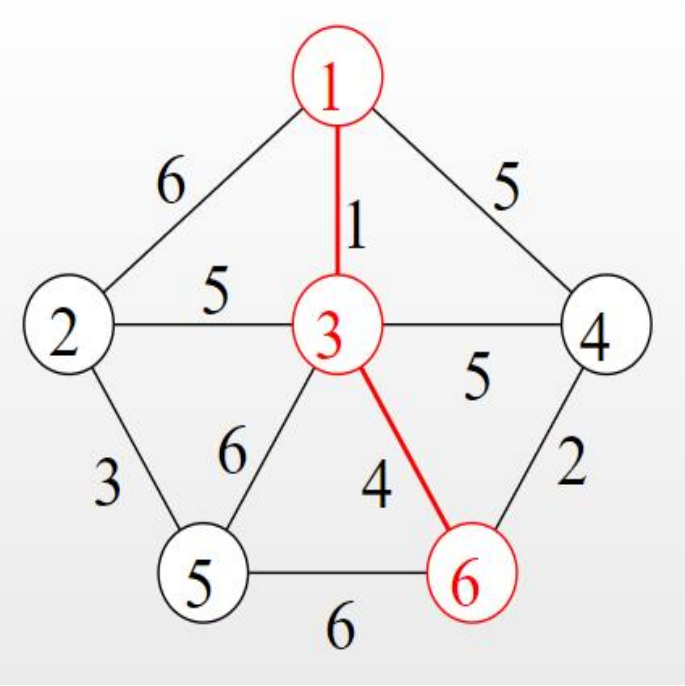



**Figure III.-12 Itération 2**

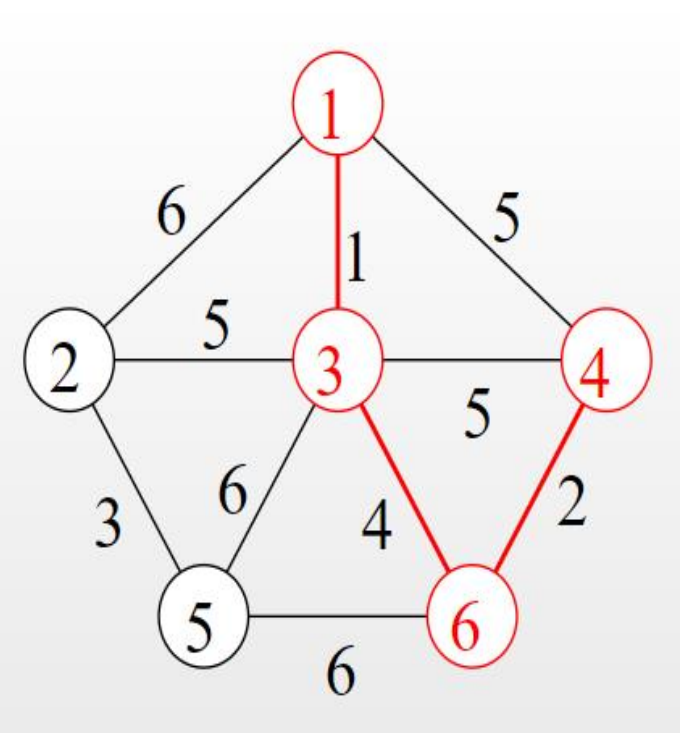



**Figure III.-13 Itération 3**

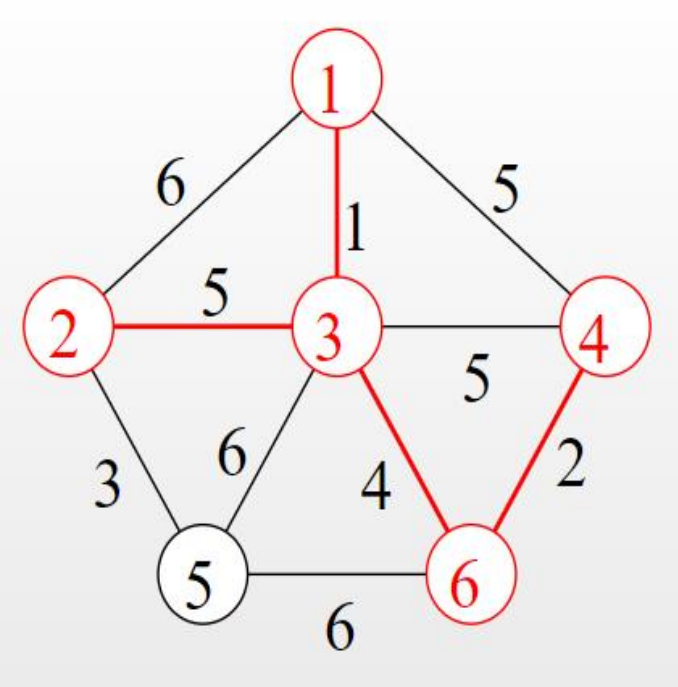



**Figure III.-14 Itération 4**

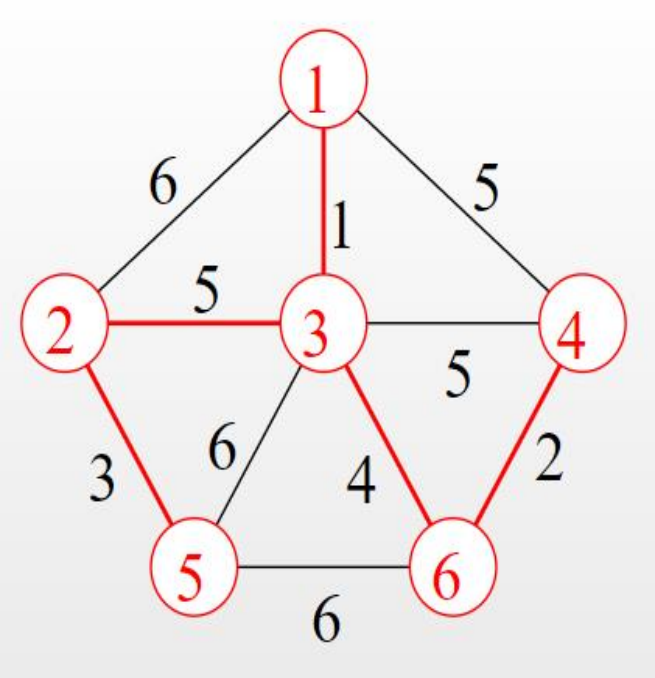



**Figure III.-15 Itération 5**

### 4.3.Différences entre les algorithmes de Prim et Kruskal :

- Prim construit l'arbre de recouvrement minimal (ARM) en commençant par un point quelconque du graphe, alors que Kruskal construit l'ARM à partir des arêtes
- Dans le cas d'un graphe disjoint, Prim retournera l'ARM dans lequel appartient le point de départ, alors que Kruskal renverra tous les arbres de recouvrement minimal (constituant ainsi une forêt d'arbres)

Le choix d'un de ces 2 algorithmes découle le plus souvent du choix de la représentation du graphe. La plupart des représentations permettent, pour un point donné, d'accéder à la liste des points voisins mais ne permettent pas de dresser rapidement la liste des arêtes du graphes (c'est le cas, par exemple, avec une liste d'adjacences), pourtant nécessaire à Kruskal. Dans ce cas, on privilégie donc Prim.

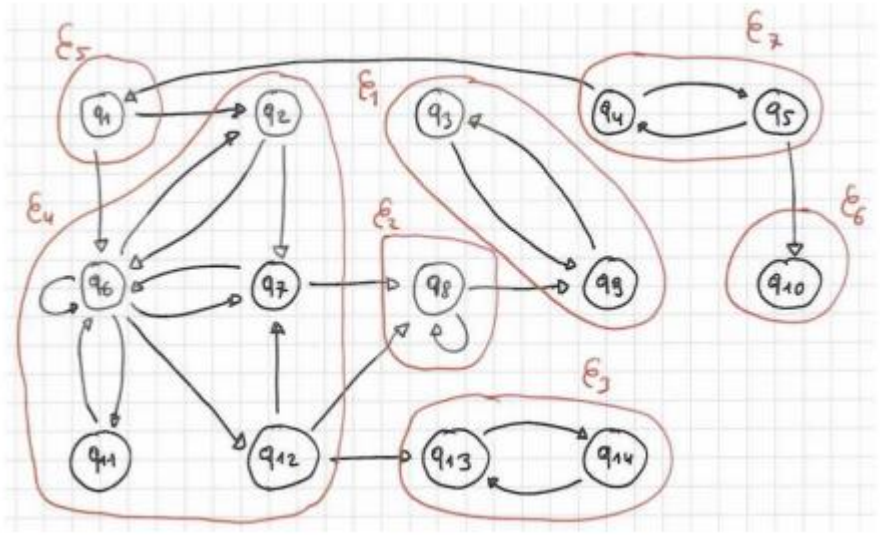
Nous avons utilisé une représentation nous permettant d'avoir directement accès aux arêtes du graphe. Nous privilégierons donc l'algorithme de Kruskal pour ses nombreux avantages (algorithme simple à comprendre, simple à mettre en place, code concis, plus rapide car effectuant moins de tris, algorithme applicable sans modification au cas d'un graphe disjoint).[11]

## 5.Composants fortement connectés :

G = (S,A) : un graphe orienté.

Une composante fortement connexe (CFC) C de G est un sous-ensemble maximal de sommets de G tel que : si u, v ∈ C, alors u $\rightarrow*$ G v et v $\rightarrow*$ G u.[12]

Exemple de l'algorithme de CFC de graphe G:




LES ALGORITHMES UTILISÉS POUR LE CALCUL DES COMPOSANTS FORTEMENT CONNECTÉS :

**5.1. Algorithme Tarjan** :

L'algorithme Tarjan pour les composants connectés fortement Il est l'un des algorithmes les plus efficaces pour la recherche de structures hautement associées dans un graphique. Opera fait la complexité linéaire au nombre d'arcs et de sommets O (| V | + | E |).

**5.2.Algorithme de Kosaraju** :

L'algorithme Kosaraju-Sharir consiste à effectuer une profondeur première recherche le graphique et une visite ultérieure dans les profondeurs de graphique transposée, dans lequel les chaînes sont inversées, le choix des nœuds afin de visiter obtenus lors de la première visite en profondeur. De cette façon, il est possible d'identifier les composants fortement connectés du graphe dans le temps Θ (V + E). Il est donc plus facile à mettre en œuvre mais algorithme moins efficace Tarjan et Gabow qui portent une profondeur première recherche le graphique.[13]

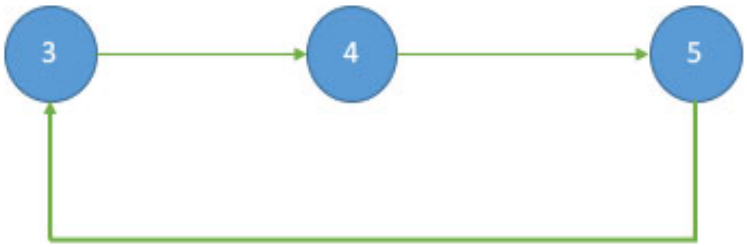
**6.Détection de cycle :**

La détection de cycles dans les graphes peut être intéressante pour plusieurs problèmes, que ce soit pour détecter des graphes caractérises par sous-structure interdite, pour des études de complexité (certains problèmes étant ´ dans P tandis que d'autre sont NP-complet), pour améliorer des algorithmes ´ dans les réseaux,...[15]

Que ce soit dans le cas oriente ou non-orient ´ e, il est relativement ais ´ e de ´ chercher un cycle simple ou élémentaire. Il est en effet bien connu qu'un par- ´ cours du graphe, que ce soit en largeur ou en profondeur, permet si l'on a colorie le graphe au passage de détecter la présence d'un cycle. ´ Cet algorithme est optimal et a la base de plusieurs autres algorithmes : Re-cherche d'un arbre couvrant, garbage collector dans les langages de programmation tels Java ou Ocaml,... Le parcours en profondeur comme en largeur explore une fois et une seule chaque sommet et chaque arete. Ainsi, sa complexité est en ´ O(n + **m). Dans le cas d'un graphe oriente, le** parcours du graphe peut ne pas détecter ´ les cycles. Il existe alors un autre algorithme en O(n + m)[16]

**Exemple de l'algorithme de detection de cycle :**

Dans le graphe ci-dessus il existe un cycle entre les sommets 3, 4 et 5



# Chapitre 04

# Le calcul du chemin le plus court à l'aide de techniques d'apprentissage en profondeur

## 1.INTRODUCTION

Le calcul des distances de chemin les plus courtes entre les nœuds est au cœur de nombreux algorithmes et applications graphiques.

Les méthodes exactes traditionnelles telles que la recherche en largeur en premier (BFS) ne s'appliquent pas aux réseaux massifs contemporains, en évolution rapide. Par conséquent, il est nécessaire de trouver des méthodes d'approximation pour permettre un traitement de graphe évolutif avec une accélération significative. Dans cet article, nous utilisons des plongements vectoriels appris par des techniques d'apprentissage en profondeur pour approximer les distances des chemins les plus courts dans de grands graphiques. Nous montrons qu'un réseau de neurones feedforward alimenté par des plongements peut approximer des distances avec une erreur de distorsion relativement faible. La méthode suggérée est évaluée sur le Réseaux sociaux Facebook, BlogCatalog, Youtube et Flickr.

Trouver les distances de chemin les plus courtes entre les nœuds dans un graphe est une primitive importante dans diverses applications. Par exemple, le nombre de liens entre deux URL indique une similitude de page dans un graphique du Web [1]. Dans une ontologie Web sémantique, les distances de chemin les plus courtes entre les entités sont utilisées pour classer leurs relations [2]. Le nombre de sauts d'une personne à une autre indique le niveau de confiance dans un réseau de confiance [3]. Dans les réseaux sociaux, la distance de chemin la plus courte est utilisée pour calculer la centralité de proximité [4].

Les méthodes traditionnelles de calcul de la distance des nœuds ne sont pas adaptées à la taille du graphique. Pour un graphe avec n nœuds et m arêtes, les implémentations ef fi caces de Dijkstra calculent les chemins les plus courts pour un nœud vers d'autres en un temps O (n log n + m). Une légère généralisation de Dijkstra, connue sous le nom d'algorithme A $*$, utilise des techniques heuristiques pour calculer les distances les plus courtes. Dans la pratique, A $*$ fonctionne au moins aussi rapidement que l'algorithme de Dijkstra, cependant, la complexité d'exécution est toujours O (n log n + m).

Tolérable pour les petits graphes, mais sur un graphe de grand million de nœuds, le calcul peut prendre jusqu'à une minute pour une seule distance de nœud [5]. Étant donné le coût élevé de stockage des distances précalculées, les chercheurs n'ont que le choix d'échantillonner des sous-graphiques ou de rechercher des résultats approximatifs. Pour de nombreuses applications pratiques, trouver des distances approximatives entre les nœuds peut être suffisant.

Dans cet article, nous proposons une nouvelle méthode pour approximer les mesures de distance de chemin le plus court entre deux nœuds en utilisant des plongements vectoriels générés par des techniques d'apprentissage en profondeur.

Les plongements Nodevec et Poincaré ont été récemment étudiés pour les tâches d'analyse de graphe de base telles que la prédiction de liens [6], [7] et la classification des nœuds [6], [8]. Cela est dû au fait qu'ils accélèrent le temps de calcul et donnent des résultats précis par rapport aux techniques traditionnelles de factorisation matricielle.

Nous exploitons donc ces deux plongements pour approcher les distances de chemin les plus courtes. À un niveau élevé, nous apprenons d'abord les plongements en utilisant node2vec [9] et Poincaré [10] pour chaque nœud du graphe. Nous suivons ensuite l'approche «basée sur les points de repère» proposée dans [11], où nous choisissons un petit nombre de nœuds comme points de repère. Nous calculons les distances réelles des chemins les plus courts entre chaque point de repère et tous les nœuds restants. Nous utilisons ces paires comme ensemble d'entraînement. Enfin, nous formons un réseau de neurones à réaction avec des vecteurs d'incorporation pour approximer la distance entre les nœuds correspondants. Les réseaux à réaction directe sont bien connus pour leur bonne capacité de représentation [12] et devraient être suffisants pour approcher les fonctions de distance de chemin le plus court donnant des plongements de graphes.

En raison de la propriété de parcimonie des réseaux (sociaux) du monde réel, le nombre d'arêtes est proportionnel au nombre de nœuds. Par conséquent, la génération de la vérité terrain à l'aide de l'approche basée sur les points de repère nécessite une complexité temporelle linéaire qui donne un temps d'exécution linéaire pour l'ensemble de la méthode proposée. Plus de détails sont développés dans la section III. Dans l'ensemble, nous étudions les techniques avancées d'incorporation de graphes pour apporter les contributions suivantes:

• Nous motivons et suggérons théoriquement l'utilisation des plongements node2vec et Poincaré pour l'approximation de la distance des nœuds dans les grands graphes (section III).

• Nous montrons que les réseaux de neurones peuvent prédire les distances de chemin les plus courtes de manière efficace, en particulier pour les chemins plus courts (section IV).

• Nous démontrons que différentes techniques d'inclusion ainsi que différents réglages de paramètres ont une influence significative sur la qualité de l'approximation (section IV).

• Nous comparons nos approximations aux travaux les plus importants de l'état de l'art et montrons que notre approche surpasse les méthodes actuelles en termes de précision de prédiction (section IV).

Afin de démontrer les performances de notre méthode dans la pratique, nous menons des expériences avec quatre ensembles de données du monde réel. Les expériences indiquent que notre prédicteur fonctionne mieux pour les chemins plus courts. Cependant, les plongements se comportent mal pour se rapprocher des chemins les plus longs. La raison réside à la fois dans la technique d'incorporation et dans la stratégie d'échantillonnage pour former les paires. Nous fournissons plus de détails dans la section IV.

Le reste du chapitre est structuré comme suit. La section II donne un aperçu des techniques récentes d'approximation de la distance de chemin le plus court et d'incorporation de graphiques. Notre méthode proposée est détaillée dans la section III. La section IV résume les résultats de l'évaluation expérimentale. Enfin, section V, remarques finales.

## 2. CONTEXTE ET TRAVAUX CONNEXES

### 2.1. Chemin le plus court

La tâche consistant à calculer la distance de chemin la plus courte entre un nœud unique et tous les autres nœuds est connue sous le nom de chemins les plus courts à source unique (SSSP). Des méthodes exactes telles que Dijkstra calculent SSSP pour des graphes pondérés avec n nœuds et m arêtes dans le temps O (m + n log n). Pour les graphiques clairsemés non pondérés, les chemins les plus courts peuvent être calculés à l'aide de la première recherche en largeur (BFS) dans le temps O (m + n). Cependant, les méthodes exactes sont extrêmement lentes pour effectuer des requêtes sur les très grands réseaux en ligne actuels. Pour de nombreuses applications pratiques, trouver des distances approximatives entre les nœuds peut être suffisant. Parmi les méthodes approximatives, une famille d'algorithmes évolutifs pour ce problème sont les approches dites basées sur les points de repère [11]. Dans cette famille de techniques, un ensemble fi xe de nœuds de repère est sélectionné et les chemins les plus courts réels sont précalculés de chaque repère à tous les autres nœuds. La connaissance des distances aux points de repère, ainsi que l'inégalité du triangle, permet généralement de calculer la distance approximative entre deux nœuds quelconques en temps O (l), où l est le nombre de points de repère. [11]. Bien que les algorithmes basés sur des

points de repère n'offrent pas de solides garanties théoriques sur la qualité d'approximation [13], ils se sont avérés bien performants dans la pratique, évoluant vers des graphiques avec des millions d'arêtes avec une précision acceptable et des temps de réponse inférieurs à une seconde par requête [14] .

Inspirés par des méthodes basées sur des points de repère, les auteurs de [11] et[15], a suggéré l'idée de systèmes de coordonnées de graphe, qui incorpore des nœuds de graphe dans des points sur un espace de coordonnées. Les coordonnées résultantes peuvent être utilisées pour estimer rapidement les requêtes de distance des nœuds sur le graphique d'origine. Cependant, il présente plusieurs limites dans la pratique. Premièrement, leur processus initial d'incorporation de graphes est centralisé et coûteux en calcul, ce qui présente un goulot d'étranglement de performance significatif pour les graphes plus grands.

Deuxièmement, leurs résultats produisent des taux d'erreur relativement élevés, ce qui limite les types d'applications qu'il peut servir. Enfin, il est incapable de produire des chemins réels reliant des paires de nœuds, ce qui est souvent nécessaire pour un certain nombre d'applications graphiques.

### 2.2Incorporation de graphes

Outre les méthodes précitées, incorporant les nœuds avec l'objectif explicite de préserver la longueur de chemin la plus courte, diverses méthodes pour incorporer le graphe en général (ou réduction de dimension respectivement) ont été proposées (cf. Goyal et Ferrara [16] pour une enquête). Parmi les méthodes classiques, on trouve l'analyse en composantes principales (ACP) [17], l'analyse discriminante linéaire (LDA) [18], l'ISOMAP [19], la mise à l'échelle multidimensionnelle (MDS) [20], la LLE [21] et la carte propre laplacienne [22]. ] (cf. Yan et al. [23] pour une enquête). Cependant, la plupart de ces méthodes reposent généralement sur la résolution de la décomposition propre et la complexité est au moins quadratique dans le nombre de nœuds, ce qui les rend inefficaces pour gérer des réseaux à grande échelle. Récemment, des approches basées sur les réseaux de neurones ont été proposées, inspirées des idées de Word2Vec [24], qui est construit autour de l'hypothèse distributionnelle, affirmant que les mots dans des contextes similaires ont tendance à avoir une signification similaire [25]. DeepWalk [26] échantillonne les marches aléatoires du graphe et les traite comme des équivalents de phrases. Autrement dit, étant donné la représentation d'un nœud dans l'espace d'incorporation, DeepWalk se rapproche de la probabilité conditionnelle des nœuds dans le voisinage. Ainsi, les nœuds partageant un voisinage similaire ont tendance à avoir une représentation similaire dans l'espace d'incorporation. Semblable à Word2Vec factorisant implicitement une matrice de cooccurrences

de mots [27], [28], Deepwalk a été montré pour factoriser une matrice de probabilités de transition de nœuds [29]. Node2vec [6] étend Deepwalk en introduisant des paramètres pour contrôler le comportement de marche aléatoire. Aux choix de paramètres les plus extrêmes, node2vec utilise un premier échantillonnage en largeur ou en profondeur en premier, explorant le voisinage proche ou les nœuds éloignés dans le réseau. LINE [30] optimise explicitement les plongements pour capturer la proximité du premier et du second ordre, en entraînant des plongements séparés pour eux, qui sont finalement concaténés. La proximité du premier ordre est donnée par des connexions explicites entre les nœuds, tandis que la proximité du second ordre est donnée en comparant les quartiers des nœuds. Nickel et Kiela [7] intègrent le graphe dans un espace hyperbolique, ou plus précisément dans une boule de Poincaré à n dimensions, capturant la hiérarchie et la similitude. Plusieurs approches ont été proposées pour mieux modéliser respectivement les relations interurbaines ou la proximité d'ordre supérieur. Au lieu d'approcher la matrice de proximité d'ordre k, comme le fait DeepWalk, GraRep [31] la calcule avec précision, au prix d'une complexité accrue. Yang et coll. [32] atténuent ce problème en utilisant des informations provenant de matrices de proximité d'ordre inférieur. Les auteurs de HOPE [33] ont expérimenté différentes mesures de similarité, telles que l'indice de Katz, le Page Rank enraciné et Adamic-Adar. HARP [34] et Walklets [35] abordent la capture de proximité d'ordre supérieur en adaptant la stratégie de marche aléatoire. Tandis que HARP grossit le graphique et apprend les représentations via des graphiques réduits hiérarchiquement, les Walklets sautent les étapes des marches aléatoires. Des architectures profondes ont été proposées, visant à capturer la non-linéarité dans les graphes. SDNE [36] et DNGR [37] utilisent des auto-encodeurs, GCN [38] dé fi nit un opérateur de convolution sur le graphe.

A notre connaissance, l'approximation des distances de chemin les plus courtes sur la base d'enfoncements qui n'ont pas été spécifiquement adaptés à cette fin n'a pas encore été étudiée. III. APPROCHE

A. Approximation de la distance

Soit G = (V, E) un graphe non pondéré non orienté avec n nœuds et m arêtes. Les techniques d'incorporation de graphes créent une valeur réelle, ce qu'on appelle l'inclusion vectorielle $\varphi(v) \in R^d$ pour chaque nœud $v \in V$. Étant donné une paire de nœuds $u, v \in V$ avec la distance de chemin la plus courte réelle d u, v, le but est d'approximer la distance comme ˆ d en utilisant un réseau de neurones à réaction directe. Formellement, nous définissons

ˆD comme fonction ˆd: $\varphi(u) \times \varphi(v)$ 7 → R +

qui mappe une paire de plongements vectoriels à une distance de chemin le plus court à valeur réelle d u, v dans G.

Pour entraîner le réseau de neurones, nous devons extraire les paires d'apprentissage du graphe entier G. Nous choisissons d'abord un petit nombre de l nœuds comme points de repère, où l n. Nous calculons ensuite les distances les plus courtes réelles entre chaque point de repère et tous les nœuds restants à l'aide de BFS. Il donne l (n - l) paires d'entraînement. Étant donné une paire d'apprentissage <φ (v), φ (u)>, nous créons une représentation conjointe comme entrée du réseau de neurones en appliquant une opération binaire, à savoir soustraction, concaténation, moyenne et multiplication ponctuelle.

## Références bibliographiques :

### Chapitre1 :

[1] Shai, Offer, and Kenneth Preiss, "Graph theory representations of engineering systems and their embedded knowledge." Artificial Intelligence in Engineering 13.3 (1999): 273-285.

[2] Formanowicz, Piotr, and Krzysztof Tanaś, "A survey of graph coloring-its types, methods and applications." Foundations of Computing and Decision Sciences 37.3 (2012): 223-238.

[3] Adam Schenker, Mark Last, horst Banke, Abraham andel, “Clustering of Web documents using a graph model”, Springer werlog, September 2007.

[4] : Théorie des graphes, O. Cogis + C. Robert, Éditions Vuibert, ISBN 2711753212

[5] Kavitha D, Rao BVM, Babu VK. A Survey on Assorted Approaches to Graph Data Mining. Int J Comput Appl. 2011;14(1):43–6

[6] Trlnajstic N. CHEMICAL GRAPH THEORY. Randie M, Klein DJ, Mezey P 0., Trinajstic N, editors. 323 p

[7] Aldrich H, Bates T, Kuhn T. Open HPI Mooc Courses. 2001. p. 1–5.

[8] Kashima H, Tsuda K, Inokuchi A. Marginalized kernels between labeled graphs. In: Proceedings of the 20th international conference on machine learning (ICML-03). 2003. p. 321–8.

[9] Gutman I, Polansky OE. Mathematical concepts in organic chemistry. Springer Science & Business Media; 2012. 28 p.

[10] Aridhi S. Distributed Frequent Subgraph Mining in the Cloud. Engineering Doctoral School of Clermont Ferrand; 2013

[11] https://perso.liris.cnrs.fr/christine.solnon polyGraphes.pdf

[12] Do, Thi Ngoc Quynh, and Amedeo Napoli, A graph model for text analysis and text mining. Diss. Master Thesis, Université de Lorraine, 2012.

[13] M. A. Rajan, M. Girish Chandra and Lokanatha C. Reddy, “Concepts of Graph Theory Relevant to Ad-hoc Networks”, Int. Journal of Computers, Communications & Control, Volume III, pp. 465-469, 2008.

[14] Minsky, M. L. (1969), Semantic information processing. Cambridge: The MIT Press

[15] Mizzaro, S., & Robertson, S. (2007). Hits hits trec: exploring ir evaluation results with network analysis. In Kraaij et al. (2007), pp. 479–486

[16] Zhang, B., Li, H., Liu, Y., Ji, L., Xi, W., Fan, W., et al. (2005), Improving web search results using affinity graph. In Baeza-Yates et al. (2005), pp. 504–511

[17] Kaltenrieder P, Portmann E, Binggeli N, Myrach T., A conceptual model to combine creativity techniques with fuzzy cognitive maps for enhanced knowledge management,

Springer; 2015

[18] Kotoulas S, Zeng Y, Huang Z., The 2012 international workshop on web-scale knowledge representation, retrieval, and reasoning, CIKM‟12; 2012 Oct 29-Nov 2; Maui, HI, USA. c2012.

[19] Vasif V. Nabiyev and Ünal Çakiroğlu, “Application of Graph Theory in an Intelligent

Tutoring System for Solving Mathematical Word Problems”, Eurasia Journal of Mathematics,

Science & Technology Education, 2016, 12(4), 687-701

[20] “Artificial Intelligence: Foundations of Computational Agents”, available online at: http://artint.info/html/ArtInt_49.html

[21] Zhao, Jun, Osman Yağan, and Virgil Gligor. "Random intersection graphs and their applications in security, wireless communication, and social networks." arXiv preprint arXiv: 1504.03161 (2015).

[22] Van Deursen, Arie, Ali Mesbah, and Alex Nederlof, "Crawl-based analysis of web applications: Prospects and challenges." Science of Computer Programming 97 (2015): 173- 180.

[23] Ahmat, Kamal. "Graph Theory and Optimization Problems for Very Large Networks." arXiv preprint arXiv: 0907.3099 (2009).

[24] Ganguly, Soumyajit, and Vikram Pudi, "Paper2vec: Combining Graph and Text Information for Scientific Paper Representation", European Conference on Information Retrieval, Springer, Cham, 2017.

[25] Deri, Luca, et al., "Graph theoretical models of DNS traffic", Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International, IEEE, 2013.

[26] Song, Chunyao, et al. "Event pattern matching over graph streams", Proceedings of the VLDB Endowment 8.4 (2014): 413-424.

**Chapitre2:**

[1] تأليف االن بونيه ترجمة د .علي صبري فرغلي , الذكاء االصطناعي واقعه و مستقبله , سلسلة كتب ثقافية شهرية : يصدرها المجلس الوطني للثقافة و الفنون و آالداب ,الكويت , إبريل 1993 .

[2] : Judith Hurwitz and Daniel Kirsch , Machine Learning for dummies, chapter 1 Understanding machine learning, page 4, IBM Limited Edition, 2018.

[3] : RUSSELL, Rudolph. Machine learning: Step-by-step guide to implement machine learning algorithms with python. [sn], 2018.

[4] : Aurélien Garivier , Apprentissage supervisé , 13 juin 2013 .

[5] : https://studylibfr.com/doc/2870064/table-des-matières---département-dinformatique-et-de-rec… , Consulté le 29/04/2019.

[6] : Michel Crucianu, Meziane Yacoub, Cours - Estimation de densité — Cours Cnam RCP208, Mis à jour le janv. 22, 2019. Créé avec Sphinx .

[7] : Christian Gagné, Réduction de la dimensionnalité, Publié le 5 octobre 2016.

[8]: https://acatr.wordpress.com/2017/12/04/atelier-du-13-decembre-lapprentissagepar-renforcement/ , Publié le 04/12/2019, Consulté le 19/05/2019.

[9] : JANSSOONE, Thomas. Analyse de signaux sociaux multimodaux: application à la synthèse d'attitudes sociales chez un agent conversationnel animé. 2018. Thèse de doctorat.

[10] : https://meritis.fr/ia/methodes-de-transfer-learning/ , Angelina Fausto, 19 septembre 2018 , Consulté le 08/04/2019 .

[11] : Jojo Moolayil, Learn Keras for Deep Neural Networks A Fast-Track Approach to Modern Deep Learning with Python, 2019.

[12] :https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deeplearning-17262/ , Consulté le 12/04/2019.

[13] : Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., ... & Hodjat, B. (2019). Evolving deep neural networks. In Artificial Intelligence in the Age of Neural Networks and Brain Computing (pp. 293-312). Academic Press.

[14] : Zhang, J., & Zong, C. (2019). Deep Learning for Natural Language Processing. In Deep Learning: Fundamentals, Theory and Applications (pp. 111-138). Springer, Cham.

[15] : Fink, M., Liu, Y., Engstle, A., & Schneider, S. A. (2019). Deep Learning-Based Multi-scale Multi-object Detection and Classification for Autonomous Driving. In Fahrerassistenz systeme 2018 (pp. 233-242). Springer Vieweg, Wiesbaden.

[16] : Zhang, J., Xie, Y., Wu, Q., & Xia, Y. (2019). Medical image classification using synergic deep learning. Medical image analysis, 54, 10-19.

[17] : Xiao, T., Liang, S., & Meng, Z. (2019, May). Hierarchical Neural Variational Model for Personalized Sequential Recommendation. In The World Wide Web Conference (pp. 3377-3383). ACM.

[18] : Chancellor, S., Birnbaum, M. L., Caine, E. D., Silenzio, V., & De Choudhury, M. (2019, January). A Taxonomy of Ethical Tensions in Inferring Mental Health States from Social Media. In Proceedings of the 2nd ACM Conference on Fairness, Accountability, and Transparency (Atlanta GA.

[19] : Ye, Z., Li, L., Fu, Q., Ren, J., & Hu, F. (2019). SR-GAN: Semantic Rectifying Generative Adversarial Network for Zero-shot Learning. arXiv preprint arXiv:1904.06996.

[20] : Baumgartl, H., Tomas, J., Buettner, R., & Merkel, M. (2019, July). A novel Deep-Learning Approach for Automated Non-Destructive Testing in Quality Assurance based on Convolutional Neural Networks. In Proceedings of the 13th International Conference on Advanced Computational Engineering and Experimenting.

[21] : Li, H., Zhou, S., Yuan, W., Li, J., & Leung, H. (2019). Adversarial-Example Attacks Toward Android Malware Detection System. IEEE Systems Journal.

[22] : Tao, C., Wu, W., Xu, C., Hu, W., Zhao, D., & Yan, R. (2019, January). MultiRepresentation Fusion Network for Multi-Turn Response Selection in RetrievalBased Chatbots. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (pp. 267-275). ACM.

[23] : Deng, S., Jia, S., & Chen, J. (2019). Exploring spatial–temporal relations via deep convolutional neural networks for traffic flow prediction with incomplete data. Applied Soft Computing, 78, 712-721.

[24] : Degrave, J., Hermans, M., & Dambre, J. (2019). A differentiable physics engine for deep learning in robotics. Frontiers in neurorobotics, 13.

[25] : https://medium.com/jedha/les-applications-du-deep-learning-d674483d44af , Consulté le 23/04/2019.

[26] https://www.electronique-mixte.fr/wp-content/uploads/2018/08/Cours-Intelligence-artificielle-15.pdf

**Chapitre3:**

R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows : Theory, Algorithms, and Applications , Prentice Hall, 1993.

C. Berge, Graphes et Hypergraphes, Dunod, 1970.

M. Gondran, M. Minoux, Graphes et Algorithmes, Eyrolles, 1995.

Lucas Létocart : Algorithmique de graphes, Institut Galilée, Université Paris 13.

[5]https://developpement-informatique.com/article/179/algorithmes-de-parcours-dun-graphe

[6] https://lipn.univ-paris13.fr/~borne/Docs/Graphes_chap3_Parcours.pdf

[7] https://zestedesavoir.com/tutoriels/681/a-la-decouverte-des-algorithmes-de-graphe/727_bases-de-la-theorie-des-graphes/3353_parcourir-un-graphe/

[8] http://cermics.enpc.fr/polys/oap/node75.html!;

[9] Microsoft PowerPoint - chap2.ppt (univ-mlv.fr)

[10] Demonstrations.pdf (parisdescartes.fr)

[11]Algorithmes sur les graphes en Prolog (developpez.com)

[12] Composantes fortement connexes - Algorithmique – L3 (irif.fr)

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction aux algorithmes, Deuxième édition. MIT Press et McGraw-Hill, 2001. ISBN 0-262-03293-7. Chapitre 22.5, pp.552-557.

Camil Demetrescu, Irene Finocchi, Joseph F. Italien, Algorithmes et structures de données, McGraw-Hill, 2004, ISBN 9788838661617. Chapitre 11.5, pp.289-293

[15] adrien-panhaleux.pdf (ens-lyon.fr)

[16] Cours de Vincent Bouchitte r ´ edig ´ e par Brice Goglin. ´ Graphes et algorithmique des graphes. Ecole normale sup ´ erieure d'informatique, 1998.