République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Filière : Informatique

Option : STIC

Data indexing and query processing via a Voronoi diagram for the Internet of Things

Encadré Par :

Kouahla Zine Eddine

Rédiger par :

Ahmed Herga Manar

Membres de jury :

Dr Farou Brahim

Dr Boughareb Djalila

Juin 2022

Remerciement

La réalisation de ce mémoire a été grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

En tout premier lieu, je remercie le bon Dieu, tout puissant, de m'avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés.

Un grand merci à mes parents pour tout le soutien et l'amour qu'ils me portent depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours, puisse dieu, vous accorder santé, bonheur, et langue vie et faire en sorte que jamais je ne vous déçoive.

Merci à mes deux frères Monder, Mohcen et ma petite sœur que j'aime Mayar d'être à mes coté tout au long mon cursus éducatif.

Je voudrais adresser toute ma reconnaissance à mon encadreur monsieur **Kouahla Zin-Eddine** pour sa patience, sa disponibilité et surtout ses judicieux conseil qui ont contribué à alimenter ma réflexion.

Je désire aussi remercier les professeurs de département d'informatique Guelma qui m'ont fourni les outils nécessaires à la réussite de mes études universitaires.

Je voudrais exprimer ma reconnaissance envers mes amis et mes sœurs qui m'ont donné la vie : Malek, Amel, Rayan et Djihad qui m'ont apporté leur soutien moral et intellectuel tout au long de ma démarche.

Résumé

Ces dernières années, le développement de la technique et l'augmentation des données hétérogènes reçues par les capteurs d'iot sont causer l'incompétence des systèmes de traitement traditionnels pour traiter et stocker ces données volumineuses. Et cette problématique a ouvert la porte aux chercheurs à concevoir des nouveaux systèmes et modèles.

Il existe plusieurs structures d'indexation mais les plus connus qui ont prouvé leurs efficacités sont : l'indexation arborescente où se trouve l'arbre BCCF, BCCF* et l'indexation graphique où se trouve le diagramme de Voronoi. Malgré les avantages de ces structures, il reste le problème de performance de processus de recherche et qualité de l'index. Et tant que la combinaison fait la force cette étude viser à combinais les deux structures pour rendre le système capable de traiter et stocker les données massives de l'ido. Pour cela, nous proposons l'utilisation de graphe de Voronoi sur un échantillon comme prétraitement et complétait l'indexation par l'utilisation du principe de l'arbre BCCF, pour minimiser le cout de construction et améliorer la qualité de processus de découverte et de recherche et le plus important pour une vision globale des données massives de l'ido.

Mots clés : Indexation, Internet des Objets, Optimisation, Recherche Knn, Similarité.

Abstract

In recent years, the development of technology and the increase in heterogeneous data received by IOT sensors have caused the traditional processing systems to be incompetent to process and store this large data. And this issue has opened the door for researchers to design new systems and models.

There are several indexing structures but the best known ones that have proven their effectiveness are : tree indexing where the BCCF tree is located, BCCF* and graphical indexing where the Voronoi diagram is located. Despite the advantages of these structures, there remains the problem of search process performance and index quality. And as long as the combination makes the strength this study aims to combine the two structures to make the system capable of processing and storing the massive data of the IOT. Pour cela, nous proposons l'utilisation de graphe de Voronoi sur un échantillant comme prétraitement et complété l'indexation par l'utilisation du principe de l'arbre BCCF, to minimise the cost of construction and improve the quality of discovery and research processes and most important for a global view of massive IOT data.

Keywords : Indexing, Internet of Thigs, Optimization, Knn search, Similarity.

Table des matières

Remerciement 2

Résumé 3

Abstract 4

Introduction générale 8

Contexte et Problématique : 8

Objectif : 9

Organisation de mémoire : 9

Chapitre 01 : Internet d'objet 10

1.1 Introduction : 11

1.2 Internet des objets (IOT) : 11

1.2.1 Définition : 11

1.3 Comment fonctionne IOT : 12

1.4 Architecture des systèmes IOT : 13

1.5 Les avantages et les inconvénients de l'IOT : 14

1.6 Domaine d'application d'IOT : 14

1.7 Cloud computing : 15

1.7.1 Définition : 15

1.7.2 Caractéristique du Cloud Computing : 16

1.7.3 Types de Cloud Computing : 17

1.7.4 Les modèles de service Cloud : 18

1.8 Fog Computing : 19

1.8.1 Définition : 19

1.8.2 Architecture du Fog Computing : 19

1.8.3 Le Fog Computing vs le cloud computing : 20

1.9 Edge computing (informatique de périphérie) : 21

1.10 Conclusion : 22

Chapitre 02 : Technique d'indexation arborescente dans l'espace métrique 23

2.1 Introduction : 24

2.2 Indexation arborescente dans l'espace métrique : 24

2.2.1 Partitionnement de l'espace : 27

2.3 Arbre BCCF : 30

2.4 Conclusion : 30

Chapitre 03 : Technique d'indexation graphique dans l'espace métrique 31
3.1 Introduction :..... 32
3.2 Technique d'indexation graphique dans l'espace métrique :..... 32
3.2.1 K-graphiques voisins les plus proches :..... 32
3.2.2 Graphiques de voisinage relatif :..... 33
3.2.3 Graphiques du petit monde navigable :..... 34
3.3 Voronoi diagramme : 36
3.4 Conclusion :..... 36
Chapitre 04 : Indexation arborescente VS Indexation graphique 37
4.1 Introduction :..... 38
4.2 Indexation arborescente :..... 38
4.2.1 Avantage de l'arbre BCCF :..... 39
4.2.2 Inconvénient de l'arbre BCCF :..... 39
4.3 Indexation graphique :..... 40
4.3.1 Avantage d'indexation graphique :..... 44
4.3.2 Inconvénient d'indexation graphique :..... 44
4.4 Conclusion :..... 45
Chapitre 05 : Notre Proposition 46
5.1 Introduction :..... 47
5.2 L'architecture de système proposé :..... 47
5.3 Prétraitement : réduction des données et le hachage :..... 49
5.3.1 Réduction des données :..... 49
5.3.2 Table d'hachage :..... 50
5.4 Indexation : Combinaison du graphe de Voronoï et des arbres :..... 50
5.5 La recherche Knn :..... 53
5.6 Conclusion :..... 56
Chapitre 06 : Implémentation et résultat 57
6.1 Introduction :..... 58
6.2 Collections indexées :..... 58
6.3 Evaluation de l'algorithme de construction :..... 59
6.4 Evaluation de la recherche Knn :..... 62
6.5 Conclusion :..... 69
Conclusion générale : 70

Table des figures

FIGURE 1 : INTERNET DES OBJETS 12
FIGURE 2 : ARCHITECTURE EN COUCHES DE L'IOT 13
FIGURE 3 : LES AVANTAGES ET LES INCONVENIENTS DE L'IOT 14
FIGURE 4 : LES QUATRE TYPES DU CLOUD 17
FIGURE 5 : LES TROIS MODELES DES SERVICES CLOUD 18
FIGURE 6 : L'ARCHITECTURE A TROIS NIVEAUX DU FOG COMPUTING 19
FIGURE 7 : EXIGENCES D'INDEXATION DES DONNEES MASSIVES [19] 25
FIGURE 8 : PARTITIONNEMENT D'ESPACE AVEC L'ARBRE BCCF 30
FIGURE 9 : PROPRIETE DE PROXIMITE DE LA RNG. LES SOMMETS U ET V SONT RELIES S'IL N'Y A PAS DE SOMMET A L'INTERSECTION DES DEUX BOULES [7] 34
FIGURE 10 : DIAGRAMME DE VORONOI. 36
FIGURE 11 : ARCHITECTURE DE SYSTEME D'INDEXATION [102] 47
FIGURE 12 : PROPOSITION DE SCHEMA EXPLICATIF DE NOTRE SYSTEME. 52
FIGURE 13 : ARCHITECTURE D'ALGORITHME DE RECHERCHE 55
FIGURE 14 : TEMP ET COMPLEXITE DE LA CREATION DE VORONOI 59
FIGURE 15 : NOMBRE DES DISTANCES CALCULES 60
FIGURE 16 : NOMBRE DES COMPARAISONS EFFECTUES 61
FIGURE 17 : NOMBRE DE DISTANCE CALCULE POUR LA RECHERCHE KNN 64
FIGURE 18 : NOMBRE DE COMPARAISON EFFECTUE POUR LA RECHERCHE KNN 66
FIGURE 19 : TEMP DE LA RECHERCHE KNN (SECONDE) 68

Liste des tableaux

TABLEAU 1 : COMPARAISON ENTRE LE CLOUD ET LE FOG COMPUTING 20
TABLEAU 2 : FOG COMPUTING VS EDGE COMPUTING 21
TABLEAU 3 : TAXONOMIE DES TECHNIQUES D'INDEXATION ARBORESCENTES DANS LES ESPACES METRIQUE 29
TABLEAU 4 : TYPES DE GRAPHIQUES POUR LES RECHERCHES DE SIMILARITE. [7] 35
TABLEAU 5 : ALGORITHMES DE CONSTRUCTION POUR LES GRAPHIQUES DE PROXIMITE (N EST LA TAILLE DE L'ENSEMBLE DE DONNEES). [7] 41
TABLEAU 6 : ALGORITHMES DE RECHERCHE DE SIMILARITE POUR LES GRAPHIQUES DE PROXIMITE. [7] 42
TABLEAU 7 : COMPLEXITE DE TABLEAU D'HACHAGE [103] 50
TABLEAU 8 : CARACTERISTIQUES DES DATASETS UTILISEES 58

Introduction générale

Dans ce travail, nous nous sommes intéressés à l'indexation de données dans un espace métrique. Ce besoin est né de la nécessité d'indexer et de rechercher des données de plus en plus nombreuses et complexes en temps réel.

Introduisons donc le contexte général de ce travail, ainsi que les problèmes à résoudre dans ce mémoire.

Contexte et Problématique :

Internet of Things (IOT) est une révolution technologique qui va changer la manière dont les gens travaillent, pensent et vivent. Il s'agit d'un paradigme avancé qui nécessite un ensemble de technologies, connaissances et infrastructures. IOT est un sujet de recherche qui offre des opportunités illimitées et occupe une grande importance par les chercheurs au monde [109].

Ces dernières années, la quantité de données a augmenté de façon exponentielle. Selon les statistiques publiées par le « Statista Research Department » [110], d'ici à 2025, les prévisions suggèrent qu'il y aura plus de 75 milliards d'objets connectés à l'Internet en service. Cela représenterait une augmentation de presque trois fois par rapport à l'année 2019. Pour cette raison, la quantité de données collectées par ces objets devient incalculable. Cette expansion a créé le besoin de stocker, de gérer, de traiter et de sécuriser d'énormes volumes de données afin de tirer le meilleur parti de cette masse d'informations pour un public de plus en plus large.

Pour résoudre cette problématique, et comme une solution prometteuse un nouveau paradigme informatique appelé Cloud computing est né, en raison de sa capacité (théoriquement) illimitée de stockage et de traitement, de sa disponibilité, de son indépendance géographique et de la transparence des données et des services.

Malgré les avantages offerts par cette technique, elle nécessite un délai de latence plus long pour recevoir les données des capteurs en raison de la longue distance qui les sépare, ce qui rend difficile de développer des applications qui fonctionnent en temps réel telles que les systèmes de surveillance des patients, les systèmes de contrôle des véhicules, etc. Pour remédier à ce problème, un nouveau paradigme a émergé, ce paradigme appelé Fog (From cOre to edGe) computing qui est considéré comme une extension du cloud computing pour minimiser le temps de latence, et de calcul en temps réel.

Malgré que le cloud computing offre plus de capacité de stockage et de traitement, il n'est pas suffisant pour fournir des services à des millions d'utilisateurs [111] parce que l'amélioration de la qualité du service ne dépend pas seulement de l'augmentation de l'espace de stockage et de la puissance de calcul, mais aussi de la structuration et de l'organisation des données. Ces dernières sont connues dans la littérature sous le nom d'Indexation. Par conséquent Plusieurs techniques d'indexation ont été proposées pour une indexation et une recherche de similarité efficace dans les réseaux de l'IdO dans l'espace métrique telle que L'arbre BCCF (arbre binaire basé sur les conteneurs du Cloud-Fog computing) et des autres versions de l'arbre BCCF sous le nom BCCF* et les graphes tel que le graphe de Voronoi qui a prouvé leur performance.

Objectif :

Dans ce projet, nous visons à proposer une nouvelle structure d'indexation de données massives de l'IdO basé sur la combinaison des deux structures (graphe et arbre) et basée sur le paradigme informatique Cloud-Fog-Edge computing pour diminuer le coût de la construction de l'index et réduit en tant que possible l'espace de recherche pour améliorer la qualité du processus de découverte et de récupération de données de l'IdO et de minimiser le nombre de temp de recherche.

Organisation de mémoire :

Ce mémoire est organisé en six chapitres :

Chapitre 01 : Internet d'objet

Ce chapitre représente un état de l'art sur IdO, Cloud, Fog et Edge computing.

Chapitre 02 : Technique d'indexation arborescente dans l'espace métrique

Ce chapitre représente un état de l'art sur la technique d'indexation arborescente

Chapitre 03 : Technique d'indexation graphique dans l'espace métrique

Ce chapitre représente un état de l'art sur la technique d'indexation graphique

Chapitre 04 : Technique d'indexation arborescente vs technique d'indexation graphique

Ce chapitre représente une comparaison entre technique d'indexation arborescente et l'indexation graphique

Chapitre 05 : Notre proposition

Dans ce chapitre, nous allons présenter notre contribution, à savoir la conception d'une nouvelle technique d'indexation qui combine les deux structures. La méthode proposée est basée sur le prétraitement des données au niveau des couches Edge et Fog computing.

Chapitre 06 : implémentation et résultat

Dans ce dernier, nous avons validé notre proposition en faisant des expérimentations sur différentes collections de données réelles.

Nous concluons ce mémoire par une conclusion générale et quelques perspectives sur le domaine.

Chapitre 01 : Internet d'objet

1.1 Introduction :

L'internet des objets désigne la nouvelle tendance qui consiste à utiliser de petits appareils connectés en permanence pour envoyer des données à une application dorsale basée sur le cloud. Cela ouvre un nouvel ensemble de possibilités et de produits que les entreprises développent et vendent sur les marchés industriels et grand public.

Dans ce chapitre, nous présentons un aperçu de l'état de l'art qui met en évidence certains détails techniques de notre cadre. Ce chapitre se compose de trois sections. Dans la première section, nous présentons tout d'abord la technologie *IOT*. La deuxième section présente un aperçu général du cloud computing. Enfin, la dernière section du chapitre présente la définition et la description des paradigmes *Fog computing et Edge computing*.

1.2 Internet des objets (IOT) :

1.2.1 Définition :

Il est difficile de donner une définition exacte de l'internet des objets, c'est pourquoi nous avons rassemblé quelques définitions qui existent déjà dans la littérature.

Définition 01 : *L'idée de base de ce concept est que la présence omniprésente autour de nous d'une variété de choses ou d'objets - tels que des étiquettes d'identification par radiofréquence (RFID), des capteurs, des actionneurs, des téléphones portables, etc. - qui, grâce à des systèmes d'adressage uniques, sont capables d'interagir les uns avec les autres et de collaborer avec leurs voisins pour atteindre des objectifs communs. - qui, grâce à des schémas d'adressage uniques, sont capables d'interagir les uns avec les autres et de collaborer avec leurs voisins pour atteindre des objectifs communs [5].*

Définition 02 : *L'internet des objets (IoT) fait partie de l'internet du futur et peut être défini comme une infrastructure de réseau mondiale dynamique avec une capacité de configuration automatique basée sur des protocoles de communication standard et interopérables où les objets ont des identités, des attributs physiques et des personnalités virtuelles et utilisent des interfaces intelligentes, et sont intégrés de manière transparente dans le réseau d'information. Ces objets peuvent interagir et communiquer entre eux et avec l'environnement en échangeant des données et des informations sensibles à l'environnement tout en réagissant de manière autonome aux événements et en les influençant en exécutant des processus qui déclenchent des actions et créent des services avec ou sans intervention humaine. [6].*

Définition 03 : *Selon l'Union internationale des télécommunications (UIT), l'IdO est l'infrastructure mondiale de la société de l'information, qui fournit des services avancés en interconnectant des objets (physiques ou virtuels) à l'aide de la technologie Internet. [7]*

De notre point de vue et de notre compréhension, nous pouvons définir l'IoT comme un réseau d'objets interconnectés (capteurs et actionneurs) qui peuvent communiquer entre eux et avec l'environnement grâce à des protocoles de communication spécialisés pour l'échange de données générées, ces données sont stockées dans le Cloud, traitées dans le Fog et exploitées par des techniques spécialisées de Big Data.

Figure 1 : Internet des objets

1.3 Comment fonctionne IOT :

Il y a plusieurs étapes pour le fonctionnement de l'IOT [8] :

- **Premièrement (l'unicité) :** chaque objet connecté avec l'internet des objets il est nécessaire d'avoir une identité unique, grâce à l'évolution des adresses IP (Internet Protocol) et leurs générations, car grâce à elle nous pouvons fournir des milliers d'adresses IP différentes, et nous devons avoir un identifiant unique qui peut être attribué à chaque objet physique sur la terre.

- **Deuxièmement (la communication) :** Chaque objet peut communiquer avec d'autres objets. Il existe un certain nombre de technologies sans fil modernes qui permettent la communication entre objets, par exemple : wifi, Bluetooth, Near-field communication (NFC), ZigBee, Z-Wave, etc.

- **Troisièmement (capteur) :** Tout objet doit être équipé de capteurs afin que nous puissions obtenir des informations à son sujet. Les capteurs peuvent être des capteurs de température, d'humidité, de lumière, de mouvement, à ultrasons, etc. Les nouveaux capteurs sont de plus en plus petits, moins chers et plus durables.

- **Quatrièmement (microprocesseur) :** Chaque objet a besoin d'un microcontrôleur (microprocesseur) pour gérer les capteurs et la communication et pour effectuer des tâches. Il existe de nombreux microcontrôleurs qui peuvent être utilisés dans l'IOT en fonction des besoins.

- **Dernièrement (cloud) :** Pour stocker les serveurs, analyser et afficher les données, il est recommandé d'utiliser les services du cloud. Plusieurs grandes entreprises s'y intéressent déjà, comme "Watson", "IBM", "Google Cloud Platform", "Azure", "Microsoft", "Oracle Cloud", etc.

1.4 Architecture des systèmes IOT :

L'architecture IoT [9] est généralement classée en 4 couches, comme le montre la figure 02.

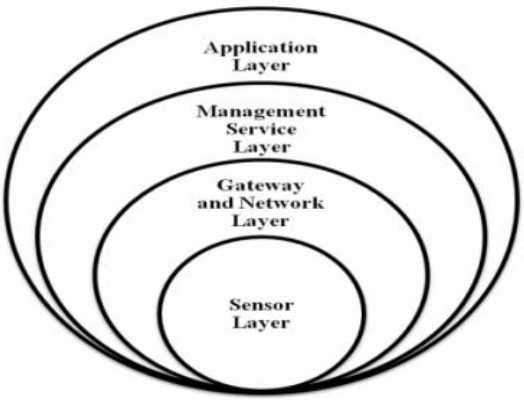


Figure 2 : Architecture en couches de l'IoT

- **Couche capteur :** Il s'agit de la couche la plus basse de l'architecture IOT, qui consiste en réseaux de capteurs, systèmes embarqués ou d'autres capteurs logiciels qui sont différentes formes de capteurs déployés sur le terrain. Chacun de ces capteurs possède identification et stockage d'informations (étiquettes RFID par exemple), collecte d'informations (réseaux de capteurs par exemple), etc. [10]
- **Passerelle et couche réseau :** Cette couche est responsable du transfert les informations collectés par les capteurs vers la couche suivante. Il devrait évolutif, flexible et standards protocole universel pour le transfert de données à partir d'appareils hétérogènes (différents types de nœuds capteurs). Cette couche devrait avoir un niveau élevé de performance et de réseau robuste.
- **Couche de service et de gestion :** cette couche agit comme une interface entre la passerelle (couche réseau) et couche application. Il est responsable de la gestion des appareils, gestion de l'information et de la capture de grande quantité de données brutes, extraire les informations pertinentes des données stockées ainsi que des données en temps réel. Ainsi que la sécurité et la confidentialité des données doit être garantie.
- **Couche application :** Il s'agit de la couche la plus élevée, qui fournit une interface à l'utilisateur pour accéder à diverses applications à différents utilisateurs. Les applications peuvent être utilisées dans plusieurs domaines.

1.5 Les avantages et les inconvénients de l'IOT :

Les avantages et les inconvénients de l'IOT [9] sont montrés dans le diagramme suivant :

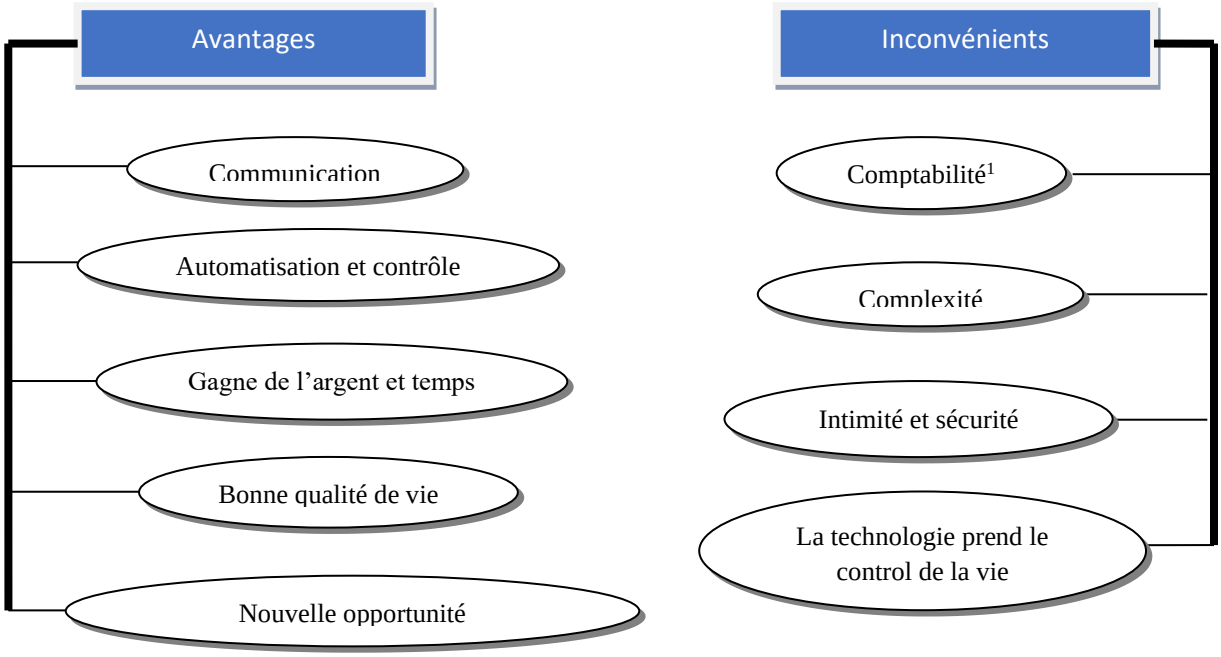


Figure 3 : les avantages et les inconvénients de l'IOT

1.6 Domaine d'application d'IOT :

L'IoT peut trouver ses applications dans presque tous les aspects de notre quotidien. Vous trouverez ci-dessous quelques exemples [11] :

- **Conception de maisons intelligentes :** l'IoT peut aider à la conception des maisons intelligentes, par exemple : la gestion de la consommation d'énergie, interaction avec les appareils, détection des urgences, sécurité à domicile, etc.
- **Conception de système de transport intelligent :** Le système de transport intelligent fournira des moyens de transport de contrôle et de gestion efficace utilisant la technologie avancée des capteurs, informations et réseau. Le transport intelligent a nombreuses fonctionnalités intéressantes telles que l'électronique non-stop péage autoroutier, commande et ordonnancement mobiles d'urgence, application de la loi sur les transports, surveillance des infractions aux règles des véhicules, réduction de la pollution de l'environnement, système antivol, éviter les embouteillages, signaler les incidents de circulation, smart beaconing, minimisant les retards d'arrivée, etc.

1 Comptabilité : Comme les appareils de différents fabricants seront interconnectés dans l'IoT, actuellement, il n'y a pas d'internationale norme de compatibilité pour le marquage et la surveillance équipement.

- **Conception de villes intelligentes :** l'IoT peut aider à concevoir villes intelligentes, par exemple, surveillance de la qualité de l'air, découverte des situations d'urgence itinéraires, éclairage efficace de la ville, arrosage des jardins, etc.
- **Applications industrielles :** l'IoT peut trouver des applications dans l'industrie, par exemple, la gestion d'une flotte de voitures pour une organisation. L'IoT permet de suivre leurs performances environnementales et traiter les données pour déterminer et choisir celle qui a besoin de maintenance.
- **Prévision des catastrophes naturelles :** la combinaison de capteurs et leur coordination et simulation autonomes aidé à prévoir l'occurrence de glissements de terrain ou d'autres catastrophes et de prendre les mesures appropriées à l'avance.
- **Applications médicales :** L'IoT peut aussi trouver des applications dans le secteur médical pour sauver des vies ou améliorer la qualité de vie, par exemple, surveiller les paramètres de santé, surveiller les activités, soutien à la vie autonome, suivi de la prise de médicaments, etc.

1.7 Cloud computing :

Le terme "cloud computing" a fait couler beaucoup d'encre ces dernières années, cependant il n'existe pas encore de définition standard de ce terme.

Pour le grand public, le Cloud se matérialise notamment par les services de stockage et de partage de données numériques type Box, Dropbox, Microsoft OneDrive, ou Apple iCloud sur lesquels l'utilisateur peuvent stocker des contenu personnel (photos, vidéos, musique, document, etc.) et y accéder n'importe où dans le monde depuis n'importe quel terminal connecté.

Pour les informaticiens le Cloud est une infrastructure dans laquelle la puissance de calcul et de stockage sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison Internet sécurisé. C'est pourquoi des travaux récents ont été consacrés à la normalisation et à la standardisation de la définition de ce paradigme. Par exemple, Vaquero et al. [12] ont examiné plus de 20 définitions différentes provenant de diverses sources pour déterminer une définition standard. Dans ce travail, nous adopterons la définition du cloud computing de l'institut national des normes et de la technologie (NIST) [13], car elle couvre tous les aspects.

1.7.1 Définition :

« *Cloud computing est un modèle permettant un accès réseau omniprésent, pratique et à la demande à un ensemble partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement approvisionnées et libérées avec un minimum d'effort de gestion ou d'interaction avec le fournisseur de services. Ce modèle de Cloud computing se compose de cinq caractéristiques essentielles, de trois modèles de services et de quatre modèles de déploiement* » [13]

1.7.2 *Caractéristique du Cloud Computing* :

Le Cloud Computing a des différentes caractéristiques, nous mentionnons les plus importants [13] :

- **Libre-service à la demande :** Les ressource informatique sont demander par le client à tout moment, et ces dernières sont fournie automatiquement sans aucune interaction avec le fournisseur de service.
- **L'accès universel :** Les ressources sont disponibles sur le réseau et accessibles via internet qui favorisent l'utilisation par des plates-formes hétérogènes de clients légers ou lourds (par exemple, téléphones portables, tablettes, ordinateurs portables et postes de travail).
- **Mutualisation des ressources :** Les ressources informatiques du fournisseur sont regroupées pour servir plusieurs consommateurs en utilisant un modèle multi-locataires, avec différentes ressources physiques et virtuelles dynamiquement attribué et réaffecté selon la demande des consommateurs. Il y a un sentiment d'emplacement indépendant dans la mesure où le client n'a généralement aucun contrôle ou connaissance sur l'emplacement des ressources fournies, mais peut être en mesure de préciser l'emplacement à un niveau plus élevé d'abstraction (par exemple, pays, État ou centre de données). Des exemples de ressources comprennent le stockage, le traitement, la mémoire et la bande passante du réseau.
- **Élasticité :** Les capacités peuvent être fournies et libérées de façon élastique, dans certains cas à l'extérieur et à l'intérieur en fonction de la demande. Pour le consommateur, les capacités disponibles pour la fourniture semblent souvent illimitées et peuvent être approprié en toute quantité à tout moment.
- **Facturation ou Paiement à l'usage :** Les systèmes Cloud contrôlent et optimisent automatiquement l'utilisation des ressources en tirant parti d'une capacité de compte1 à un certain niveau d'abstraction approprié au type de service (p. ex., stockage, traitement, bande passante et comptes d'utilisateurs actifs). L'utilisation des ressources peut être surveillés, contrôlés et signalés, assurant la transparence à la fois pour le fournisseur et consommateur du service utilisé.

1.7.3 Types de Cloud Computing :

Comme on a déjà vu dans la définition de NIST, le service Cloud Computing a quatre types ou modèle de déploiement (Cloud privé, Cloud public, Cloud Communautaire, Cloud hybride).

Qui correspondent à des usages différents [13] :

- **Cloud privé :** L'infrastructure Cloud est fournie pour une utilisation exclusive par une seule organisation comprenant plusieurs consommateurs (p. ex., unités opérationnelles). Il peut être détenu, géré et exploité par l'organisation, une tierce partie ou une combinaison de ces entités, et il peut exister sur les lieux ou à l'extérieur.
- **Cloud public :** L'infrastructure Cloud est fournie pour une utilisation ouverte par le grand public. Il peut être détenu, géré et exploité par une entreprise, un établissement d'enseignement ou un organisme gouvernemental, ou une combinaison de ces organismes. Il existe dans les locaux du fournisseur de cloud.
- **Cloud communautaire :** L'infrastructure Cloud est fournie pour une utilisation exclusive par une communauté spécifique de consommateurs d'organisations qui ont des préoccupations communes (p. ex., mission, exigences de sécurité, politiques et considérations de conformité). Il peut être détenu, géré et exploité par un ou plusieurs des organismes de la collectivité, un tiers partie, ou une combinaison de ceux-ci, et il peut exister sur ou hors des lieux.
- **Cloud hybride :** L'infrastructure Cloud est une composition d'au moins deux infrastructures infonuagiques distinctes (privées, communautaires ou publiques) qui demeurent des entités uniques, mais qui sont liées entre elles par une technologie normalisée ou exclusive qui permet la portabilité des données et des applications (p. ex., explosion du nuage pour équilibrer la charge entre les nuages)

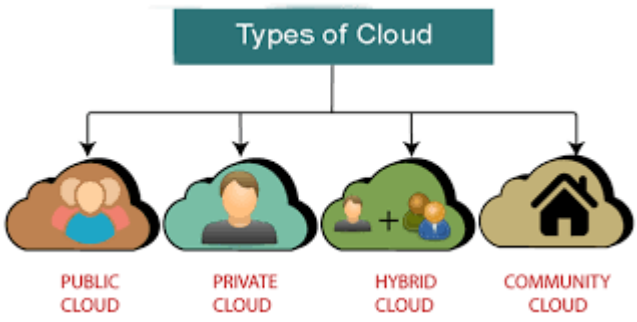


Figure 4 : Les quatre types du Cloud

1.7.4 Les modèles de service Cloud :

Selon la définition de NIST, il existe trois modèles de services Cloud [13] :

- **SaaS (Software as a Service, en anglais) :** La capacité fournie au consommateur est d'utiliser les applications du fournisseur fonctionnant sur une infrastructure cloud 2. Les applications sont accessibles à partir de divers appareils clients au moyen d'une interface client allégée, comme un navigateur Web (p. ex., courriel Web), ou d'une interface de programme. Le consommateur ne gère ni ne contrôle l'infrastructure Cloud sous-jacente, y compris les réseaux, les serveurs, les systèmes d'exploitation, le stockage ou même les capacités d'applications individuelles, à l'exception peut-être de paramètres de configuration d'applications spécifiques aux utilisateurs.
- **PaaS (Platform as a Service, en anglais) :** La capacité offerte au consommateur consiste à déployer sur l'infrastructure infonuagique des applications créées ou acquises par les consommateurs à l'aide de langages de programmation, de bibliothèques, de services et d'outils pris en charge par le fournisseur. Le consommateur ne pas gérer ou contrôler l'infrastructure cloud sous-jacente, y compris le réseau, les serveurs, systèmes d'exploitation, ou de stockage, mais a le contrôle sur les applications déployées et peut-être les paramètres de configuration de l'environnement d'hébergement d'applications.
- **IaaS (Infrastructure as a Service, en anglais) :** La capacité fournie au consommateur est de fournir des ressources de traitement, de stockage, de réseaux et d'autres ressources informatiques fondamentales où le consommateur est en mesure de déployer et d'exécuter des logiciels arbitraires, ce qui peut comprendre des systèmes d'exploitation et des applications. Le consommateur ne gère ni ne contrôle l'infrastructure infonuagique sous-jacente, mais il a le contrôle des systèmes d'exploitation, du stockage et des applications déployées ; et peut-être un contrôle limité de certains composants réseau (p. ex., pare-feu hôte).

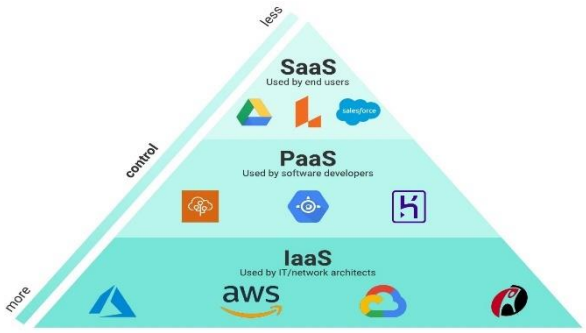


Figure 5 : les trois modèles Des services Cloud

1.8 Fog Computing :

Fog Computing est une plate-forme hautement virtualisée qui fournit des services de calcul, de stockage et de réseau entre les appareils finaux et les centres de données de cloud computing traditionnels. [14]

Ce paradigme a été introduit pour la première fois par Cisco en 2012 pour relever les défis des applications de l'IOT dans le Cloud computing.

1.8.1 *Définition :*

(Cisco) : « Le Fog étend le Cloud pour être plus proche des objets qui produisent et agissent sur les données de l'IdO. Ces dispositifs, appelés nœuds du Fog, peuvent être déployés n'importe où avec une connexion réseau : sur le sol d'une usine, au sommet d'un poteau électrique, le long d'une voie ferrée, dans un véhicule ou sur une plate-forme pétrolière. Tout appareil doté d'une connectivité informatique, de stockage et de réseau peut être un nœud du Fog. Les exemples incluent les contrôleurs industriels, les commutateurs, les routeurs, les serveurs intégrés et les caméras de vidéo-surveillance. »

1.8.2 Architecture du Fog Computing :

La Figure 06 illustre une architecture à trois niveaux [15] du Fog computing, l'une des architectures de base et largement utilisées dans la littérature.

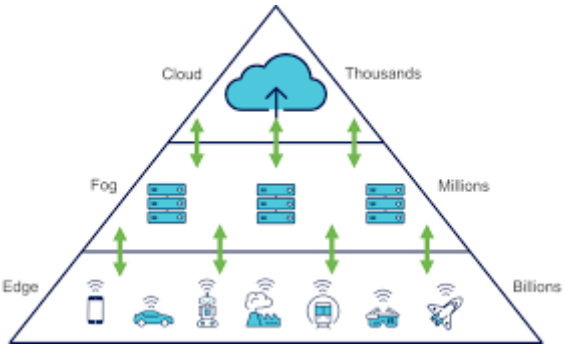


Figure 6 : L'architecture à trois niveaux du Fog computing

- **La couche Cloud :** La couche de nuage comprend de solides serveurs de calcul et de stockage pour être en mesure de faire le stockage d'une énorme quantité de données et une analyse informatique approfondie pour divers services d'application si nécessaire.
- **La couche Fog :** Cette couche est située sur le bord du réseau. La couche de Fog est une combinaison d'un grand nombre de nœuds de Fog (physiques ou virtuels). Ces nœuds de Fog sont largement répartis entre les dispositifs de nuage et d'extrémité. Ils peuvent être mobiles ou fixes en un seul endroit. Les nœuds de Fog ont la capacité de calculer, de transmettre et de stocker temporairement les données reçues. L'analyse et la prestation de services des applications en

temps réel se font dans la couche de Fog. En outre, les nœuds de Fog ont la possibilité de connexion et d'interaction avec les nœuds de Fog voisins.

- **La couche Edge :** C'est la couche la plus proche de l'environnement physique et de l'utilisateur final, y compris les dispositifs et capteurs IoT.

Dans cette architecture, chaque capteur ou dispositif d'extrémité est connecté à l'un des nœuds du Fog en appliquant une connexion filaire ou en appliquant des technologies sans fil comme WiFi, 3G, 4G, Bluetooth, réseau local sans fil, et ZigBee. En outre, les nœuds du Fog peuvent se connecter les uns aux autres par des technologies de communication sans fil ou filaire. [15]

1.8.3 Le Fog Computing vs le cloud computing :

L'informatique de brouillard (Fog Computing) est très similaire à l'informatique en nuage (Cloud Computing) dans le sens où les deux sont est constitué de systèmes virtuels fournissant flexibilité de la fourniture à la demande de ressources de calcul, de stockage et de réseau. Mais par rapport au Cloud, le Fog est mis en œuvre très près des utilisateurs finaux. [16]

Le tableau 1[16] résume la comparaison entre le Cloud et le Fog computing.

Tableau 1 : Comparaison entre le Cloud et le Fog Computing

Les critères	**Cloud computing**	**Fog computing**
Latence	Elevée	Basse
Localisation du service	Dans l'internet	Aux frontières du réseau
Distance Client-serveur	Plusieurs sauts	Un seul saut
Gigue de retard	Elevée	Très basse
Sécurité	Moins sécurisé	Plus sécurisé
Vulnérabilité	Haute probabilité	Très faible probabilité
Nombre de nœuds	Peu	Beaucoup
Support de mobilité	Limité	Supporté
Interaction en temps réel	Limité	Supporté
Géodistribution	Centralisé	Distribuée
Mobilité	Prise en charge limitée	Prise en charge

1.9 Edge computing (informatique de périphérie) :

Edge computing est une philosophie de mise en réseau visant à rapprocher le calcul de la source de données afin de réduire la latence et l'utilisation de la bande passante. En termes plus simples, l'informatique de périphérie signifie exécuter moins de processus dans le cloud et déplacer ces processus vers des endroits locaux, comme sur l'ordinateur d'un utilisateur, un périphérique IoT ou un serveur périphérique. Amener le calcul au bord du réseau minimise la quantité de communication longue distance qui doit se produire entre un client et un serveur. [17]

Le Fog computing et le Edge computing déplacent tous deux le calcul et le stockage vers la périphérie du réseau et plus près des utilisateurs, mais ces paradigmes ne sont pas identiques. Le tableau 02 montre brièvement la différence entre ces deux modèles.

Tableau 2 : Fog computing vs Edge computing

Critères	**Fog computing**	**Edge computing**
Déploiement	Déployé dans les locaux des utilisateurs mobiles	Déployé comme un centre de données traditionnel avec des capacités étendues.
Ressources	Appareil virtualisé avec stockage de données intégré, installation informatique et de communication.	Il utilise un serveur de périphérie similaire à un serveur de centre de données traditionnel.
Fonctionnement	Peut être adapté à partir de composants systèmes existant	Il est entièrement construit comme un nouveau système ou un petit centre de données cloud.
Ressources consumé par rapport à cloud	La consommation d'énergie du brouillard est inférieure à celle des services cloud, mais la surcharge est élevée par rapport au cloud.	Edge utilise moins de ressources que la surcharge initiale du cloud à créer est élevée par rapport au cloud.
Contrôle du cloud	La consommation d'énergie du brouillard est inférieure à celle des services cloud, mais la surcharge est élevée par rapport au cloud.	Serveur Edge utilisant les technologies cloud et la virtualisation utilisée pour contrôler les composants Edge
Contrôle d'opérateur de réseaux	Il peut ne pas être contrôlé par les opérateurs de réseau et il utilise une distribution personnalisée.	Permet aux opérateurs de réseaux mobiles d'améliorer les services existants avec Edge.

1.10Conclusion :

Dans ce chapitre, nous avons présenté un état de l'art qui nous permet de comprendre certaines généralités sur l'Internet des objets, ainsi que sur les paradigmes du Cloud, du Fog et de l'Edge Computing.

Dans le chapitre suivant, nous montrerons l'une des techniques d'indexation existantes

Chapitre 02 : Technique d'indexation arborescente dans l'espace métrique

2.1 Introduction :

L'IOT caractérise l'avenir dans lequel les appareils sont interconnectés via l'internet. En outre, la technologie de l'Internet des objets implique de nombreuses machines connectées pour détecter une énorme quantité de données. Le traitement et le stockage de ces données nécessitent des solutions d'indexation évolutives et efficaces dans les trois niveaux de Edge, Fog et Cloud pour les applications à grande échelle afin de permettre une recherche rapide et efficace de leurs données et services.

Dans ce chapitre, nous présentons les concepts de base de l'indexation basée sur les arbres, l'espace métrique, et un état de l'art sur les techniques d'indexation de l'espace métrique existantes dans la littérature.

2.2 Indexation arborescente dans l'espace métrique :

L'indexation est une étape de l'organisation des données qui permet, entre autres, un accès efficace lors de l'exécution de requêtes de similarité. Comme, le but d'un index est de fournir un accès rapide aux objets d'une base de données, en réduisant l'espace de recherche, le nombre de calculs, la distance entre les objets et le temps de recherche [18]. Les techniques d'indexation sont classées en fonction du type d'espace dans lequel elles sont créées. Il existe deux catégories principales

(i) Techniques d'indexation dans un espace multidimensionnel
(ii) Les techniques d'indexation dans l'espace métrique.

En effet, les objets à indexer ne sont pas que de simples vecteurs, ils sont plus complexes que cela. Tout ceci pose des problèmes sur les méthodes d'indexation et de recherche. Ainsi, le développement de l'indexation a été transféré des espaces multidimensionnels aux espaces métriques. Dans ce chapitre, nous donnons un aperçu de quelques techniques d'indexation dans les espaces métriques, qui doivent répondre à certaines contraintes pour être applicables à grande échelle [19], comme le montre la figure 07.

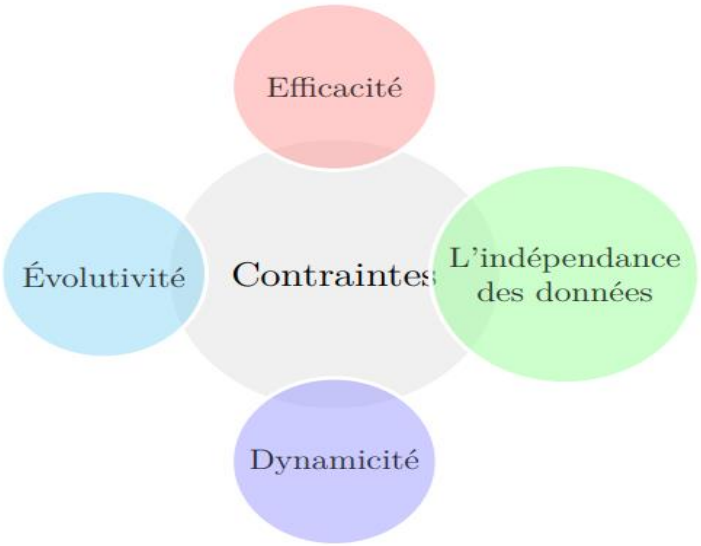


Figure 7 : Exigences d'indexation des données massives [19]

- **Evolutivité :** La structure d'index doit pouvoir indexer des volumes de données toujours croissants sans perte de performance (efficacité dans la réponse aux demandes).
- **Efficacité :** Contrairement aux structures d'accès physiques traditionnelles, qui seulement tentent de minimiser le nombre d'Entrée/Sortie sur le disque, un indice de recherche de similarité doit également prendre en compte les coûts de calcul parce que les calculs de similarité entre deux objets peuvent être très coûteux, ou même plus cher que les temps de lecture du disque.
- **Dynamique :** La structure de l'index doit être dynamique en ce qui concerne les changements de base de données, c.-à-d. ne pas nécessiter de réorganisations périodiques coûteuses, mais assurer une indexation continue.
- **Indépendance des données :** Les structures d'accès devraient fournir des bonnes performances pour toute distribution possible des données, c.-à-d. résoudre efficacement les problèmes liés à la malédiction des dimensions ou l'inefficacité dans les grandes dimensions.

Comme mentionné précédemment, les index dans l'espace multidimensionnel sont plus difficiles en raison de leur forte dépendance au type et à leurs propriétés géométriques. D'une part, cette dépendance rend très difficile l'implémentation d'une structure flexible pour répondre aux contraintes mentionnées dans la Figure 09. Et d'autre part, les techniques d'indexation et de recherche ne fonctionnent généralement pas dans les espaces multidimensionnels [20].

Ainsi, l'indexation est passée de manière incomplète des espaces multidimensionnels aux espaces métriques. Ces derniers ont été proposés comme une abstraction universelle pour les Big data ou les données massives. Parmi les principales caractéristiques de l'espace métrique qu'il ne nécessite pas une structure réelle des données, il ne nécessite qu'une fonction de distance, les propriétés de non-négativité, de symétrie et d'inégalité triangulaire des paires de points de données. L'avantage le plus important de l'indexation dans l'espace métrique est que tout type de données peut être indexé, car l'approche est basée sur le calcul des distances entre

les objets, ce qui nous permet de créer une structure d'indexation générale (capable de traiter tout type de données).

Dans ce travail, nous nous sommes concentrés sur les espaces métriques par rapport aux espaces multidimensionnels.

Dans cette section, nous avons réduit notre étude aux techniques d'indexation métrique (**techniques basées sur les arbres**). Deux approches principales de l'indexation dans les espaces métriques sont présentées ci-dessous :

- **Non-partitionnement de l'espace (partitionnement des données) :**
  L'idée principale du partitionnement des données est de créer des groupes de données, également appelés formes limites [21].
  Le M-tree [22] est une structure d'indexation métrique qui regroupe les données en boules en utilisant une fonction de distance. Elle est assez performante en haute dimension. Par contre, il souffre du problème de chevauchement qui augmente le nombre de calculs de distance pour répondre à une requête.
  L'arbre Slim [23], est une version du M-tree optimisée pour réduire le problème de recouvrement, ce dernier utilise le "fat-factor" qui fournit un moyen simple de museler le degré de recouvrement entre les nœuds d'un arbre métrique, l'arbre Slim* est une extension de l'arbre Slim.
  Le Super-M-tree [24] est une amélioration du M-tree par l'utilisation de différentes fonctions de distance (par exemple, la distance euclidienne, la distance de Hausdor, la distance Edict et la distance Dog-Keeper), le Super-M-tree est capable de répondre à la demande de sous-séquences ou de sous-ensembles tels que la recherche d'une séquence partielle similaire d'une scène dans un film, ou d'un objet similaire dans une image.
  Le MX-tree est une nouvelle structure d'indexation proposée par Murgante et al [25] basée sur la structure M-tree. Cette nouvelle structure adapte le concept de super-nœuds inspiré de la structure X-tree [26] qui évite la division insatisfaisante des nœuds, réduit le coût de calcul et reflète complètement à l'espace métrique où la complexité temporelle est réduite à $O(n^2)$ sans changement du niveau de paramètre, contrairement à la M-tree où la complexité temporelle devrait être $O(n^3)$.
  Pour résoudre le problème de recherche dans le M-tree. Le PM-tree [27] a été proposé pour combiner le mappage pivot avec le M-tree afin d'éviter les calculs de distance inutiles.
  L'arbre DSC (Dynamic Set of Clusters) [28] est une nouvelle structure métrique dynamique qui augmente la consommation de mémoire et le nombre de calculs de distance. Malheureusement, les structures d'indexation sont déformées en raison de données manquantes par différents domaines d'application où ces dernières produisent une fausse réponse à la requête. Pour résoudre ce problème, Brinis et al [29] proposent la structure Hollow tree, qui permet de gérer les données manquantes sans altérer la structure. L'arbre creux est une nouvelle méthode d'accès métrique qui fournit une stratégie pour construire des indices métriques. Cette stratégie consiste à indexer d'abord toutes les données complètes pour créer une structure cohérente, puis à insérer les éléments avec des valeurs manquantes dans les nœuds feuilles. Tout ceci est réalisé

par la technique ObAD (Observed Attribute Distance), qui permet de comparer les éléments avec des valeurs manquantes sur la base de fonctions de distance.

2.2.1 Partitionnement de l'espace :

Il existe deux sous-approches pour le partitionnement de l'espace :

2.2.1.1 Le partitionnement par boule :

La sélection des pivots dans l'approche de partitionnement en boules joue un rôle important dans la structure de l'index.
L'arbre VP (vantage point) a été proposé par Yianilos [30] et consiste à trouver l'élément médian parmi un ensemble d'objets. Cet élément est généralisé à l'arbre MVP (multi-way VP) où les nœuds de ces arbres sont divisés en quantiles, ce principe est conçu pour éliminer le problème de chevauchement.
Le MM (Memory based metric tree) [31] utilise l'approche de partitionnement des boules, mais à partir de l'intersection entre deux boules à la fois où il exploite les régions.
L'arbre oignon [32] est une extension de l'arbre MM, il a pour but de fournir une indexation rapide de données complexes à partir d'une mémoire dynamique qui découpe l'espace métrique.
L'arbre Ball [33] est un arbre binaire où chaque nœud est défini comme une boule de dimension d, contenant un sous-ensemble de points à rechercher. Chaque point est prêté à l'une ou l'autre des boules de la partition en fonction de sa distance au centre de la boule. L'arbre Ball* [34] où les distributions sont plus équilibrées et où les recherches par plus proche voisin sont plus efficaces.
L'arbre XM [35] est une nouvelle structure de partitionnement qui partitionne l'espace à l'aide de sphères en limitant le volume des régions extérieures des sphères ; en créant des régions étendues et en les insérant dans des listes liées appelées régions étendues, et aussi en excluant les ensembles vides qui ne contiennent pas d'objets.

2.2.1.2 Partitionnement par hyper plan :

L'hyperplan généralisé (GH-tree) [36] est une structure d'indexation binaire qui utilise une méthode récursive pour diviser l'espace en deux sous-espaces par l'hyperplan, ce dernier étant défini par deux points représentatifs ou pivots et le reste des points étant distribué en fonction de la distance entre ces pivots.
L'arbre GNAT [37] est une structure plus générale que l'arbre GH, car ce dernier utilise deux pivots dans chaque nœud interne, alors que l'arbre GNAT utilise m pivots (c'est-à-dire que l'arbre GH est un arbre m-air).
L'arbre Generalized Hyper-plane Bucketed (GHB) [38] est un arbre GH amélioré, l'objectif principal de cette méthode d'arbre est de créer une structure

d'indexation équilibrée avec un coût de construction minimum grâce à un nouveau type de nœuds qu'ils appellent bucket. Ces types de nœuds se trouvent au niveau des feuilles qui ont une capacité limitée pour stocker un sous-ensemble des données les plus similaires afin d'améliorer le processus de recherche.

L'arbre CD (Clustering-based Dynamic indexing) [39] cette méthode utilise la partition récursive de l'espace en deux régions. A chaque fois, deux pivots sont choisis et chacun est associé aux objets les plus proches.

L'arbre SPB (Space-filling curve and Pivot-based B +-tree) [40] l'objectif de cette technique est d'améliorer l'efficacité de la recherche, de supporter un grand nombre d'objets complexes et de réduire le coût en termes de stockage et de construction.

2.2.1.3 Partitionnement hybride :

Il existe certaines structures d'indexation qui utilisent à la fois le partitionnement par boules et par hyperplans. Par exemple :

L'IM-tree [41] est une structure proposée pour traiter le problème de dégénérescence de l'index posé par la quatrième région du MM-tree et de l'oignon. L'IM-tree sélectionne les deux points les plus éloignés comme pivots et divise la quatrième région en deux en utilisant un hyperplan et une hyper-sphère.

L'arbre NOBH (Non-Overlapping Balls and Hyper-planes tree) [42] divise l'espace métrique en hyperplans et en hyper-sphères pour organiser les données en régions non chevauchantes et réduire le nombre de distances de calcul nécessaires pour répondre aux requêtes.

Nous pouvons introduire une taxonomie simple, comme illustré dans le tableau ci-dessous, qui représente plusieurs techniques d'indexation dans les espaces métriques.

Tableau 3 : Taxonomie des techniques d'indexation arborescentes dans les espaces métrique

Approches			Application connexes
Partitionnement de l' espace	Boules	Arbre-VP	Reconnaissance des formes et traitement des images [43] Indexation et recherche d'images [44] Stockage des données de morphologie neuronale [45] Recherche de similarités sur le cloud computing [46] Détection de logiciels malveillants [47] Regroupement des images similaires [48]
		Arbre-mVP	Recherche d'images dans les systèmes de surveillance. [49]
		L'arbre-MM	Récupération d'images. [50]
		L'arbre Onion	
		Arbre-XM	Recherche d'informations sur le Web [51].
		Arbre Boule	Reconnaissance d'esquisses de visages [52]. Classification en haute dimension [53].
		Arbre Boule *	Regroupement et mise en correspondance pour la reconnaissance des classes d'objet [54]
	Hyper-plan	Arbre -GH	Recherche d'images par contenu [55]
		Arbre-GNAT	Indexation et recherche de similarité de données d'images de visages
		Arbre-EGNA	Accélérateur de requête de base de données
		Arbre-CD	Indexation et recherche de similarité de données d'images de visages
		Arbre-SPB	Recherche multimédia, Reconnaissance des formes et biologie computationnelle
		Arbre-GHB	Indexation et recherche de similarité de données d'images de visages
	Hybride	Arbre-IM	Reconnaissance des formes et biologie computationnelle. Accélérateur de requête de base de données
		Arbre-NOBH	Indexation et recherche de similarité de données d'images de visages.
Non partitionnement de l' espace	Arbre-M		Recherche de similarité dans une grande base de données multimédia [56] Recherche du voisin le plus proche [57] Accélérateur de requête de base de données [58] Système de recommandation [58] Classification [59]
	Arbre-Slim		Indexation de vidéo et recherche de similarité [60]
	Arbre-supre-M		Applications de recherche de similarités [24]
	Arbre-MX		Indexation et recherche de similarité
	Arbre-PM		Recherche de similarité dans les bases de données multimédias [61]
	Arbre-DSC		Recherche de similarité dans les bases de données multimédias [62]

	Arbre-Hollow	Stocker et récupérer de grands volumes de données [63]

2.3 Arbre BCCF :

L'arbre BCCF [64] (Binary tree based on Containers at the Cloud-Fog computing level) est une nouvelle méthode récemment proposée par l'équipe GADM du laboratoire LABSTIC. Cette technique vise à indexer les données massives de l'IdO. Elle est basée sur le partitionnement des données via l'algorithme K-means qui permet de regrouper les objets les plus similaires. A chaque étape du processus de construction récursif, des pivots sont initiés par les centres de gravité retenus par l'algorithme k-means. Illustration de la figure 08.

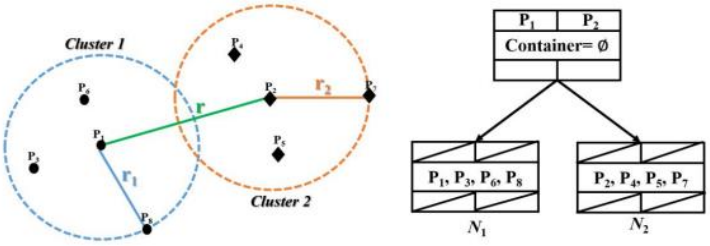


Figure 8 : Partitionnement d'espace avec l'arbre BCCF

La structure BCCF se compose de deux niveaux :

1. **Niveau nœuds internes :** Les nœuds de ce niveau contiennent uniquement deux pivots (p_1, p_2) et deux pointeurs vers les sous arbres gauche et droit. Pour les conteneurs, tous les nœuds de ce niveau ont des conteneurs vides.

2. **Niveau nœuds feuilles :** Les nœuds de ce niveau ne contiennent pas des pivots ni des pointeurs, mais contiennent seulement des conteneurs qui stockent des ensembles d'objets dont la taille d'un conteneur est inférieure ou égale à C_{max}.

2.4 Conclusion :

Dans ce chapitre, nous avons exposé l'état de l'art des techniques d'indexation à base d'arbres. Plusieurs méthodes ont été montrées qui se base sur les deux méthodes d'indexation, partitionnement et non-partitionnement de l'espace, nous pouvons introduire une légère taxinomie de certaines techniques d'indexation dans les espaces métriques.

Chapitre 03 : Technique d'indexation graphique dans l'espace métrique

3.1 Introduction :

Un graphe décrit les relations sur un ensemble d'entités. Grâce aux étiquettes des nœuds et des arcs, il peut décrire les attributs de l'ensemble des entités et des relations.
Les graphes étiquetés apparaissent dans de nombreux domaines de recherche tels que la conception de médicaments [1], la comparaison de structures de protéines [2], l'indexation de vidéos [3] et l'information sur le Web [4].
Dans ce chapitre, nous présentons les concepts de base de l'indexation graphique dans l'espace métrique ainsi qu'un état de l'art sur les techniques d'indexation dans l'espace métrique existantes dans la littérature.

3.2 Technique d'indexation graphique dans l'espace métrique :

Plusieurs travaux dans la littérature proposent l'utilisation de graphes de proximité pour accélérer la recherche de similarité. **Le tableau 04** résume les types de graphes utilisés et leurs principales propriétés. Le tableau montre que les graphes de recherche de similarité sont essentiellement des variantes de quatre types de graphes : le graphe du k-plus proche voisinage (k-NNG), le graphe du voisinage relatif (RNG), l'arbre d'approximation spatiale (SAT) et le graphe du petit monde navigable (NSW).
Par conséquent, cette section aborde les principaux types de graphes de proximité pour la recherche de similarité en fonction du type de graphe sur lequel ils sont basés. [7]

3.2.1 K-graphiques voisins les plus proches :

Le graphique du voisin *k* le plus proche *(k-NNG)* [65] est défini comme un graphique

$G = (V, E)$, où $E = \{(u, v, \delta(u, v)) \mid v \in NNk(u)_\delta\}$ tel que

$Nk(u)\delta$ est l'ensemble contenant les k plus proches voisins de u dans l'ensemble de sommets V en utilisant la fonction de similarité δ.

Les arcs de *k-NNG* peuvent être non dirigés ou dirigés, et le graphe peut être déconnecté en fonction de la valeur de *NN* et des propriétés de l'ensemble de données. De plus, il peut être utilisé pour les fonctions de similarité métriques et non métriques. [7]
Le tableau 1 montre que le *k-NNG* a plusieurs variantes qui ont été utilisées pour la recherche de similarité. L'une de ces variantes est le graphe réduit au *k-Degree (k-DRG)* [66]. L'idée principale du *k-DRG* est de construire un graphe à partir du *k-NNG* avec un nombre réduit d'arcs qui garantit que les *k* plus proches voisins d'un sommet peuvent être atteints en utilisant un algorithme glouton basé sur la propriété d'approximation spatiale.
Parmi les autres variantes, citons le *k-DRG* hiérarchique [67] et le graphe de voisinage à double couche *(DLG)* [68], qui ont été proposés par les auteurs du *k-DRG* dans des articles ultérieurs. Ces deux graphes hiérarchiques utilisent le *k-DRG* comme graphe de base. Dans le *DLG*, l'espace de recherche est réduit en utilisant des sommets représentatifs de la couche de base comme sommets de départ (graines) pour l'algorithme de recherche.

Enfin, le graphe du plus proche voisin (PBKNNG) [69] construit d'abord un *k-NNG* dirigé, puis ajoute les arcs redirigés aux arcs existants. Ce processus inclut également une heuristique pour élaguer certains des arcs ajoutés afin de limiter la quantité de mémoire consommée pour stocker les arcs. Cette approche est basée sur l'observation que la précision de la recherche du k-*NNG* s'améliore généralement lorsque le paramètre du *NN* augmente.

3.2.2 Graphiques de voisinage relatif :

Le graphique de voisinage relatif (RNG) est un sous-graphique du graphique de Delaunay avec des garanties que pour n'importe quelle paire de sommets il existe un ou plusieurs chemins qui relient la paire de sommets [70,71].

Formellement, la RNG est un graphique

G = (V, E) dont l'ensemble des arcs E est déterminé par une propriété de proximité P qui (u, v) E si et seulement si Bu(u, δ(u, v)) ∩ Bv(v, δ(v, u)) = ∅ où u, v ∈ Vet B(x,r) est une boule définie par x et un seuil de distance (rayon) r [70].

La **figure 09** illustre la propriété de proximité pour le RNG. Ce dernier est sans paramètre, c'est-à-dire que la quantité d'arêtes dans le graphique est définie par les propriétés de l'ensemble de données et, par conséquent, il est « auto-ajusté ».

Comme l'indique le tableau 1, le RNG propose deux variantes pour l'interrogation par similarité. La première variation est le graphique de la région hypersphérique (HRG) [72], un GNR dans lequel l'ensemble des sommets ne contient que les éléments qui sont des centres de régions hypersphériques.
Les régions hypersphériques sont formées par un groupe d'éléments satisfaisant une propriété de capacité (c.-à-d., le nombre maximum de nœuds dans une région) avec un rayon égal à la distance entre l'objet central et l'objet le plus éloigné de la région.

Une méthode récemment proposée qui s'inspire de la RNG est le graphique Fast Approximate Nearest Neighbor [73] (FANNG). Le FANNG utilise la règle d'occlusion pour connecter les sommets, qui est similaire au critère de voisinage RNG.
La règle d'occlusion stipule que s'il y a un arc reliant les sommets v_1 *et* v_2, il n'est pas nécessaire d'avoir un arc reliant le sommet v_1 à tout autre sommet v_3 qui est plus proche *de* v_2 *que de* v_1.
Cependant, la structure proposée ne donne des résultats exacts à une recherche avide que si l'élément de requête est identique à un sommet inséré dans le graphique. En conséquence, la propriété définie a été modifiée pour traiter des situations dans lesquelles l'intérêt principal est de rechercher à une distance *r* de l'élément de requête (Requête de plage). La propriété considérant la distance *r* de l'élément *q*, si $\delta(q, v_2) < \delta(q, v_1)$ il n'est pas nécessaire d'avoir un arc reliant de v_1 *à* v_3 *où* $\delta(q, v_3) < r$.

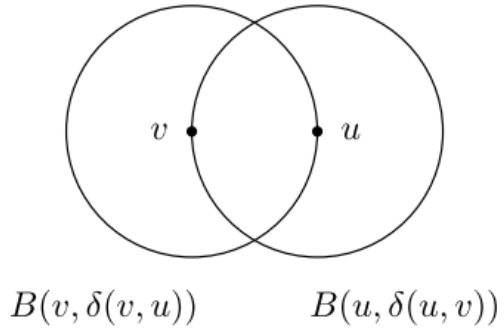


Figure 9 : Propriété de proximité de la RNG. Les sommets u et v sont reliés s'il n'y a pas de sommet à l'intersection des deux boules [7]

3.2.3 Graphiques du petit monde navigable :

Un graphique Small World (SW) est un graphique avec une propriété de clustering locale élevée, et avec des chemins courts parmi les sommets dans le graphique, en termes de nombre d'arcs [74]. En d'autres termes, une paire de sommets est reliée par un chemin qui est considéré comme petit par rapport à la taille du graphique. Le graphique Navigable Small World (NSW) [75,76] est basé sur une approximation de la topologie Delaunay Graph et Small World (SW), ce qui signifie que la plupart des nœuds peuvent être atteints de tous les autres nœuds par un petit nombre de sauts ou de pas. Ces caractéristiques sont obtenues par des arcs à longue portée qui conservent de telles propriétés de navigation.

Le graphique NSW peut être défini comme *G = (V, E)* dans lequel les sommets (nœuds) représentent les données complexes et les arcs sont des liens entre eux. Les sommets de la Nouvelle-Galles du Sud sont reliés par deux types d'arcs : les liens à courte portée, qui définissent la proximité locale entre les éléments, et les liens à longue portée, qui définissent les propriétés du petit monde dans le graphique. L'algorithme de construction de la NSW est basé sur l'insertion itérative des données complexes (vertices). Pour chaque sommet nouvellement inséré, une recherche dans le graphique est faite pour identifier et relier le sommet nouvellement inséré à ses *k* voisins les plus proches en considérant les sommets additionnés au moment. Cette idée d'algorithme est de construire dans la NSW une approximation au graphique de Delaunay. Les arcs nouvellement créés sont des liens de courte portée parce qu'il relie le sommet à leur *k* voisins les plus proches. Après plusieurs insertions, les arcs qui étaient auparavant à courte portée tendent à devenir des liens à longue portée.

Les auteurs de la NSW ont récemment proposé la version hiérarchique du graphique de la NSW, le graphique Hiérarchique Navigable Small World (HNSW) [77]. Comme son nom l'indique, dans HNSW, les arcs reliant les paires de sommets sont maintenus dans une hiérarchie : les liens longue portée sont maintenus sur la couche supérieure tandis que les liens courte portée sont insérés dans la couche inférieure. La recherche dans HNSW commence à partir de la couche supérieure et lorsque les sommets résultants pour la recherche dans la couche supérieure sont utilisés comme point de

départ sur la couche inférieure. Ce processus est répété jusqu'à ce que la recherche atteigne la couche inférieure. Ce processus de recherche est également utilisé pour insérer itérativement des sommets sur le graphique.

Tableau 4 : Types de graphiques pour les recherches de similarité. [7]

Graphe	Base du graphe	Approche	Type d'arc	Nombre d'arc par nœud	Référence
k-NNG	-	Relie un sommet à ses voisins k	Directe	K	[78,79]
Approximate k-NNG	k-NNG	Utilise des algorithmes de construction approximatifs (p. ex., NN-Descent)	Directe	K	[80, 81] [82,83]
PBKNNG	k-NNG	Ajoute des arcs inversés à k-NNG et taille les arcs excessifs en utilisant une heuristique particulière	Directe	K+ Arcs « bidirectionnels » non taillés	[84]
k-DRG	k-NNG	k-NNG avec un nombre réduit de arcs	Indirecte	$\leq k$	[66, 85]
DLG	k-NNG	Variation hiérarchique du k-DRG en utilisant les graines du graphique de base dans la couche supérieure	Indirecte	$\leq k$ dans la couche de base + arcs dans la couche supérieure	[86]
RNG	-	Sous-graphique Delaunay avec une règle d'élagage	Indirecte	Dépend de la distribution des données et de la dimensionnalité	[87]
HRG	RNG	Graphique régional inspiré du RNG (RNG connectant les centres d'hypersphère)	Indirecte	Identique à RNG avec des centres d'hypesphères comme vertices	[72]
MOBHRG	RNG	Variation du GRH avec réduction du chevauchement entre les régions hypersphériques	Indirecte	Identique à RNG avec des centres d'hypesphères comme vertices	[88]
FANNG	Inspiré par RNG	Utilise une règle d'occlusion similaire à RNG	Directe	Dépend de la distribution des données et de la dimensionnalité	[73]
NSW	k-NNG & SW	Relie les sommets à leurs voisins les plus proches en utilisant des arcs courts et longs	Indirecte	k + arcs longue portée (le cas échéant)	[75]
HNSW	NSW	Version hiérarchique de la NSW	Indirecte	k + arcs à longue portée dans les niveaux supérieurs	[77]

3.3 Voronoi diagramme :

Le diagramme de voronoï est une structure de données largement étudiée dans le domaine de la géométrie computationnelle [89] ; original, il caractérise les régions de proximité pour un ensemble de *O (k log k)* sites sur la planète où la distance des points est définie par leur distance euclidienne. Des algorithmes optimaux existent pour calculer le diagramme de voronoï en *O (k)* temps, et le diagramme de voronoï peut être représenté dans l'espace k, bien que de nombreuses extensions et généralisations aient été proposées [90]. Néanmoins, il est utile d'étudier le diagramme de voronoï graphique car il permet de résoudre efficacement de nombreux problèmes de réseau.

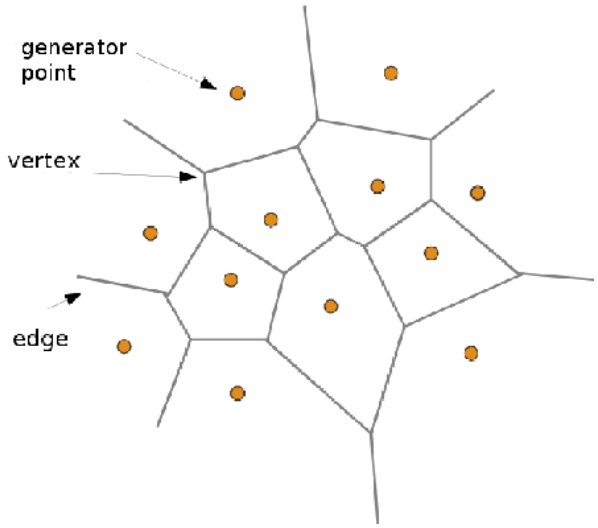


Figure 10 : Diagramme de voronoi.

3.4 Conclusion :

Dans ce chapitre, nous avons exposé l'état de l'art des techniques d'indexation basées sur les graphes. Plusieurs méthodes et types de graphes basés sur les graphes de *k*-plus proches voisins, les graphes de voisinage relatif et les graphes de petit monde navigable ont été montrés, nous pouvons introduire une légère taxinomie de certaines techniques d'indexation dans les espaces métriques.

Chapitre 04 : Indexation arborescente VS Indexation graphique

4.1 Introduction :

Les systèmes de bases de données sont de plus en plus utilisés pour gérer des données structurées complexes telles que les arbres et les graphes. L'objectif principal de notre étude est d'assurer la performance de l'indexation, ainsi que la flexibilité et la rapidité de la recherche d'information sur les données Ido. Pour atteindre cet objectif, plusieurs structures d'indexation ont été proposées, notamment le BCCF et le diagramme de Voronoï.

Dans ce chapitre, nous allons faire une comparaison entre les deux structures et nous proposons une meilleure structure pour notre travail.

4.2 Indexation *arborescente :*

Les techniques d'indexation visent à construire une structure de données qui organise la base de données afin de réduire le coût de la recherche de l'ensemble le plus similaire de candidats. La recherche de similarité efficace est un domaine de plus en plus important. L'objectif de la recherche par similarité est de trouver des objets qui sont proches d'une requête spécifique. Les techniques d'indexation continuent d'être améliorées en réponse à la recherche de grandes collections de données, où les processus deviennent de plus en plus difficiles. Plusieurs études antérieures se sont concentrées sur l'indexation à base d'arbres qui repose sur la division successive de l'espace. De plus, en raison de la croissance importante des données, ces régions croissent également rapidement. De plus, les volumes du sous-espace augmentent rapidement, ce qui pourrait conduire à une dégénérescence de l'index [64].

Benrazek et al [64] ont présenté une nouvelle structure d'indexation distribuée aux infrastructures de cloud et de fog computing. Cette structure est appelée (L'arbre BCCF). L'arbre BCCF est composé de deux niveaux :

- Le niveau des nœuds internes qui sont stockés au niveau du fog computing.
- Le niveau des nœuds feuilles qui contiennent les données réelles dans les conteneurs. Ces types de nœuds sont stockés au niveau du nuage.

La structure d'indexation proposée est adaptable aux infrastructures de cloud computing et de fog computing afin de profiter de leurs avantages qui représentent dans la plus puissante et réelle. La structure d'indexation proposée est adaptable aux infrastructures de cloud computing et de fog computing afin de profiter de leurs avantages qui représentent la capacité de traitement en temps réel la plus puissante fournie par le cloud computing en raison de sa proximité avec les capteurs et la plus grande capacité de stockage fournie par le cloud computing. L'arbre BCCF est une structure d'indexation basée sur le partitionnement de l'espace par l'algorithme de clustering k-means.

L'utilisation des k-means permet de séparer efficacement les objets dans les nœuds de l'arbre, ce qui améliore la qualité des processus de recherche. L'approche proposée vise à introduire une nouvelle structure d'indexation distribuée mieux adaptée aux technologies IoT émergentes pour améliorer la qualité de l'indexation des données IoT en temps réel.

4.2.1 Avantage de l'arbre BCCF :

- Structure hiérarchique.
- Séparation efficace des objets dans les nœuds de l'arbre.
- Amélioration de l'efficacité de l'algorithme de recherche.

4.2.2 Inconvénient de l'arbre BCCF :

- Construction coûteuse de l'arbre en raison de l'utilisation de l'algorithme K-means.
- Chevauchement entre les sous-arbres.
- Un grand nombre de nœuds visités pendant la recherche d'objets, ce qui influence négativement l'efficacité de l'algorithme de recherche.

Pour résoudre ces inconvénients et dans le but d'optimiser la structure du BCCF en minimisant le coût de construction de l'index tout en assurant la performance de l'indexation, ainsi que la flexibilité et la rapidité de la recherche d'information sur les données Ido.

Pour cela, *kemouguette* [102] a appliqué l'algorithme de clustring sur l'ensemble des données avant l'indexation. En effet, la division de données à croissance exponentielle en sous-ensembles à l'aide de boules induit une dégénérescence de l'index en raison du décalage inhérent au partitionnement de l'espace. La structure d'index qu'elle propose a montré des résultats expérimentaux intéressants et compétitifs, tant dans la construction que dans la récupération de requêtes similaires utilisant le parallélisme lors de la traversée des nœuds d'index. Cela pourrait être une meilleure alternative pour l'indexation et la récupération des données Ido.

4.3 Indexation graphique :

Les graphiques sont bien adaptés pour explorer les relations et les voisins de manière agile. Ils sont utilisés dans des applications telles que les systèmes de recommandation et les réseaux sociaux [91,92]. Pour modéliser un espace de similarité comme un graphique, une approche courante consiste à utiliser des sommets pour représenter des données complexes et des bords reliant deux sommets comme étant la relation de similarité entre deux données complexes [93,94,95,96]. Les paires des sommets peuvent être reliées par les bords selon les conditions spéciales. Notez que dans cette approche, le type de graphique est défini par la façon dont les sommets sont connectés.

Larissa C. Shimomura, Rafael Seidi Oyamada, Marcos R. Vieira et Daniel S. Kaster [7] ont présenté un sondage sur les méthodes graphiques pour la recherche de similarités. Ils ont d'abord passé en revue la documentation sur les graphiques de proximité et les méthodes de recherche de similarité fondées sur des graphiques. Ensuite ils ont présenté les principales approches de construction et les algorithmes de recherche de similarité dans les structures graphiques pour des résultats exacts et approximatifs. Enfin, ils ont effectué une analyse expérimentale approfondie des graphiques de base en fonction de leurs paramètres de construction et de recherche à l'aide d'un environnement de calcul commun et d'un ensemble complet de données réelles et synthétiques.

Tableau 5 : Algorithmes de construction pour les graphiques de proximité (n est la taille de l'ensemble de données). [7]

Méthode	Graphe	Approche	Exactitude	Cout
Brute-force k-NNG	k-NNG	Recherche des voisins les plus proches	Exacte	$O(n^2)$
Récursive Partition-based	k-NNG	Utilise une structure de données secondaire (Arbre de contrôle de la division)	Exacte	$O(n^{1.27})$ *faible dimensionalité* $O(n^{1.9})$ *haute dimensionnalité*
Pivot-based	k-NNG	Utilise une structure de données secondaire (indice pivot)	Exacte	$O(n^{1.27})$ *faible dimensionalité* $O(n^{1.9})$ *haute dimensionnalité*
NN-Descent	k-NNG	Recherche approximative des voisins les plus proches	Approchée	$O(n^{1.14})$
k-NNG construction using Lanczos bisection	k-NNG	Diviser pour régner en utilisant bisection Lanczos récursive pour diviser les données	Approchée	$O(dn^t)$
k-NN construction for visual descriptors	k-NNG	Diviser pour régner et propagation de quartier	Approchée	$O(dn \log n)$
Fast construction using LSH	k-NNG	Utilise LSH pour partitionner les éléments dans petit groupe	Approchée	$O(l(d + \log n)n)$
Brute-force RNG	RNG	Vérifie-s'il y a un élément « à l'intérieur ». La zone d'intersection de deux éléments	Exacte	$O(n^3)$
RNG construction from DT	RNG	Élimination des bords de la Delaunay Triangulation à l'aide de la technique des lignes de balayage	Exacte	$O(n \log n)$
NSW construction	NSW	Insertion incrémentale de vertex ajoutant des bords à longue portée reliant les voisins actuels	Approchée	

Ils ont également discuté de l'applicabilité des algorithmes de recherche à différents types de graphes, en montrant des exemples de scénarios où les algorithmes de recherche ne renvoient pas de réponses. Sur la base des résultats d'expériences sur des ensembles de données réels, ils ont observé le comportement général des types de graphes en fonction de leur construction primaire et des paramètres de recherche. Ils ont observé que, bien que l'algorithme de recherche exacte nécessite certains calculs de distance, le temps de recherche est prohibitif par rapport aux algorithmes de recherche approximative.

Tableau 6 : Algorithmes de recherche de similarité pour les graphiques de proximité. [7]

Méthode	**Approche**	**désavantages**
Greedy search (GS)	Utilisation directe de la propriété d'approximation spatiale	La qualité dépend du sommet initial
GNNS	Généralisation d'un GS avec plusieurs redémarrages ; nombre fixe de pas gourmands	Nécessite des paramètres (nombre de redémarrages, étapes gourmandes et expansions)
Parallel greedy search	Version parallèle de GS avec plusieurs sommets initiaux	nécessite l'utilisation de vertices initiaux
Breadth-first parallel grid search	Variation de la PSG au moyen d'une expansion de la recherche axée sur l'étendue	Les sommets initiaux de l'EFB dépendent des résultats GS
Multi-search with shared list	Variation de la recherche avide avec redémarrages en utilisant une liste partagée parmi les recherches à partir de différents départs	Nécessite des paramètres
Algorithms using seeds	Choisit des graines spécifiques pour lancer la recherche	Nécessite une sélection de semences
Exact (Metric)	Estime les limites inférieure et supérieure en utilisant les propriétés d'espace métriques pour élaguer les éléments	Limité aux distances métriques

Leurs résultats ont également montré que la NGS surpasse les autres méthodes en termes de temps de construction. Cependant, le grand nombre de voisins par sommet dans NSW augmente le temps de requête par rapport à k-NNG et NN-Descent, pour la même valeur de NN. Pour RNG, ils ont montré qu'il rivalise ou surpasse k-NNG et NN-Descent pour les petits NN, en particulier dans les ensembles de données à haute dimension. Bien que le RNG présente l'inconvénient majeur d'avoir un algorithme de construction d'une grande complexité, sa construction est sans paramètre. Pour les méthodes k-NNG, NN-Descent et NSW, ils ont trouvé un compromis entre le temps de construction et le temps d'interrogation. Plus le nombre de voisins par sommet augmente, plus le temps d'interrogation et le nombre de redémarrages nécessaires pour obtenir des réponses proches des bonnes réponses diminuent. En comparant les paramètres des graphes pour un taux de rappel donné, aucun graphe n'a été gagnant dans tous les cas. [7]

Chongsheng et al [97] discutent d'un problème important dans le traitement des requêtes spatiales, c'est-à-dire la recherche spatiale des plus proches voisins. Ils mentionnent que de nombreuses approches de pointe utilisent le diagramme de Voronoï pour améliorer le traitement des requêtes KNN car il possède des structures de données efficaces pour explorer un voisinage local dans un espace géométrique.

Ils ont également mentionné que l'approche la plus proche de GridVoronoi est la structure VoR-Tree proposée dans [98]. Les deux méthodes utilisent le diagramme de Voronoï pour exploiter pleinement sa puissance dans l'exploration d'un voisinage local dans un espace géométrique. De plus, Grid-Voronoi et VoR-Tree peuvent aider le diagramme de Voronoï à trouver rapidement le voisinage d'une requête. Cependant, il existe deux différences principales entre eux. Premièrement, VoR-Tree utilise un R-Tree pour accéder au voisinage de la requête dans l'espace géographique, mais la complexité temporelle du R-Tree pour trouver le voisinage est *O(log n)*. En revanche, GridVoronoi n'utilise que (presque) *O(1)* temps pour atteindre ce voisinage car il calcule simplement le carré virtuel correspondant pour les points de requête NN. Deuxièmement, pour accélérer le calcul pour NN, VoR-Tree associe chaque point de données dans R-Tree à une structure qui contient la cellule de Voronoï pour ce point de données et ses cellules de Voronoï voisines. Étant donné que le voisinage dérivé dans la première étape peut contenir plusieurs MBR se chevauchant dans le R-tree, il peut y avoir de nombreux points de données dans ce voisinage que VoR-Tree doit vérifier pour chaque point de données si la cellule de Voronoï dans la structure associée contient le point de requête. Avec GridVoronoi, il suffit de vérifier dans la hashmap la cellule de Voronoï correspondante du carré calculé dans la première étape. V* - Diagram [99] est un travail bien connu qui utilise le diagramme de Voronoï pour traiter les requêtes KNN mobiles. Il calcule une région sûre en considérant conjointement l'emplacement de la requête, les objets de données et l'espace de recherche actuel. Ce faisant, le coût de calcul pour fournir des réponses continues aux requêtes KNN

mobiles peut être considérablement réduit. Les auteurs de [100] utilisent également le diagramme de Voronoï pour traiter les requêtes KNN mobiles. Dans leur méthode, au lieu de vérifier continuellement la validité des zones de sécurité et de les recalculer si elles sont invalidées, ils utilisent un petit ensemble d'objets de sécurité (ensemble influent). Cela permet aux utilisateurs d'éviter le coût élevé de la précompilation du diagramme ordre-k de Voronoï en le calculant localement et à la volée.

Chongsheng et al [97] présentent GridVoronoi qui est un diagramme de Voronoï complété par une grille virtuelle pour rendre la recherche NN efficace. Ils adoptent la grille virtuelle pour trouver rapidement la cellule de grille où se trouve le point de requête, puis utilisent le diagramme de Voronoï qui est très efficace pour explorer le voisinage local de la ou des cellules de Voronoï correspondantes qui contiennent/intersectent la cellule de grille pour le plus proche voisin spatial. Des expériences sur quatre ensembles de données du monde réel ont confirmé l'efficacité de GridVoronoi. Grâce à ces expériences à grande échelle, ils ont montré que l'efficacité temporelle de GridVoronoi dépend à la fois du nombre de points de données et de la longueur de la cellule de grille adoptée.

Le diagramme de Voronoï possède de nombreuses propriétés excellentes. Premièrement, les diagrammes de Voronoï ont des propriétés mathématiques intéressantes et surprenantes. Deuxièmement, les diagrammes de Voronoï sont un outil puissant pour résoudre les problèmes de calcul.

4.3.1 Avantage d'indexation graphique :

Les avantages d'indexation graphique sont [101] :

- ✓ Portée de recherche rapidement.
- ✓ L'arithmétique des ensembles de points est facile et la structure des données est simple.
- ✓ Stabilité dynamique de la structure des données.

4.3.2 Inconvénient d'indexation graphique :

Les inconvénients d'indexation graphique sont [101] :

- Nécessitent un grand espace de rangement
- L'arithmétique des courbes ou des zones est très difficile.
- Les nœuds n'acceptent pas des vecteurs de plusieurs dimensions (juste 2D).

4.4 Conclusion :

Dans ce chapitre, nous avons établi une comparaison entre l'indexation arborescente et l'indexation graphique en mentionnant les travaux antérieurs, les avantages et les inconvénients de chaque méthode.
Notre objectif est d'optimiser l'indexation et la recherche de données, en fonction des études pré-acquises nous avons choisi de combiner les deux méthodes pour atteindre notre objectif.

Chapitre 05 : Proposition

5.1 Introduction :

L'objectif principal de cette étude est d'assurer la performance de l'indexation, ainsi que la flexibilité et la rapidité de la recherche d'information sur les données Ido. Pour atteindre cet objectif, plusieurs structures d'indexation ont été proposées, notamment l'indexation arborescente comme l'arbre BCCF et l'indexation graphique comme le diagramme de Voronoï.

Malheureusement, ces deux techniques présentent des inconvénients mentionnés dans les chapitres précédents qui diminuent l'efficacité de ses structures d'indexation.

Pour résoudre ces problèmes, il suggère :

- Proposer l'application de la fonction de réduction des dimensions multiples en 2 dimensions (2D).
- Utiliser la table d'hachage pour s'assurer qu'il n'y a pas de perte d'information (conserver l'information).
- Combiner les deux structures d'indexation (arbre et graphe) afin d'améliorer la qualité et l'efficacité des algorithmes de recherche.

5.2 L'architecture de système proposé :

L'architecture du système d'indexation proposé est montrée dans la figure 11 qui présente les principaux composants et leurs tâches respectives.

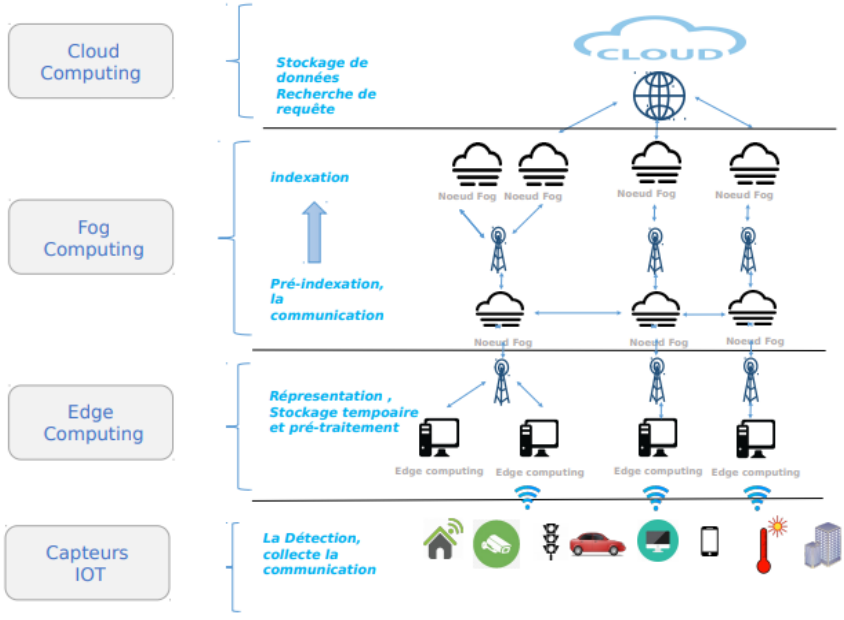


Figure 11 : Architecture de système d'indexation [102]

Le scénario du système proposé se déroule dans la couche des objets connectés ou couche des capteurs Ido. Cette couche joue un rôle important dans l'indexation des données d'Ido : (1) détection, (2) collecte, et (3) communication. Cette couche est constituée de millions de capteurs connectés hétérogènes provenant de différents domaines (industriel, commercial, santé, villes intelligentes, transport, agriculteurs, etc.), ces capteurs génèrent de grandes quantités de données.

Le rôle de la couche Edge est (1) la représentation, (2) le stockage temporaire, et (3) le prétraitement, Les données reçues par la couche inférieure (capteurs IoT) sont des données de différents types (chaîne de caractères, vecteur, etc.) Les nœuds Edge vont représenter ces données dans des formats lisibles par une machine Puis les nœuds Edge traitent les données avant de les envoyer à la couche Fog pour l'indexation.
La couche Fog est constituée de deux niveaux : Le rôle du premier niveau est (1), convertir les vecteurs en 2d et les stocker dans une table de hachage, le rôle du deuxième niveau est l'indexation, les nœuds Fog indexeront ces données par la structure d'indexation proposée (combinaison entre l'arbre et la méthode graphique), Notre structure est stockée dans cette couche pour développer un mécanisme d'indexation efficace.

La couche Cloud, son rôle est de : (a) découvrir, (b) interroger, également (c) stocker et (d) analyser. Les données réelles indexées dans la couche du brouillard seront stockées et traitées dans le nuage en raison de sa grande capacité de stockage et de traitement.
Dans le processus de recherche de similarité, les utilisateurs du système envoient leurs requêtes de recherche à la couche informatique du brouillard, et lorsque la requête est reçue par le nœud de brouillard de niveau inférieur, ce dernier envoie la requête aux nœuds de brouillard correspondants et le nœud de brouillard exécute un algorithme de recherche de similarité sur l'arbre correspondant.

Lorsque les nœuds de brouillard localisent l'ensemble des réponses dans la structure d'index locale, ils envoient des requêtes au compte en nuage pour récupérer les données réelles recherchées et les renvoyer au nœud de brouillard, qui combine les réponses reçues, sélectionne la bonne réponse et l'envoie à l'utilisateur.

5.3 Prétraitement : réduction des données et le hachage :

Pour améliorer l'indexation et la recherche et pour une vision globale des données, nous avons choisi d'utiliser le graphe de Voronoï, ce dernier est une structure qui devise l'espace en polygones disjoint, tel que nous Considérons un ensemble limité de points, appelés points générateurs, dans le plan euclidien (en général, les générateurs peuvent être n'importe quel type d'objet spatial). Nous associons tous les emplacements dans l'avion à leur ou leurs générateurs les plus proches. L'ensemble des emplacements assignés à chaque générateur forme une région appelée polygone de Voronoi ou cellule de Voronoi, de ce générateur. L'ensemble des polygones de Voronoi associés avec tous les générateurs est appelé le diagramme de Voronoi en ce qui concerne le groupe électrogène. Les polygones de Voronoi d'un diagramme de Voronoi sont collectivement exhaustifs parce que chaque emplacement dans le plan est associé à au moins un générateur. Les polygones sont également mutuellement exclusifs, à l'exception de leurs limites. Les limites des polygones, appelés bords de Voronoi, sont l'ensemble des emplacements qui peuvent être assignés à plus d'un générateur. Les polygones de Voronoi qui partagent les mêmes bords sont appelés polygones adjacents et leurs générateurs sont appelés générateurs adjacents, mais l'inconvénient de cette structure est que : les nœuds de ce type de graphe n'indexent pas des vecteurs de n dimensions *(nD)* tel que nos data set est un ensemble de vecteurs de plusieurs dimensions *(nD)*.
Pour résoudre ce problème, nous avons choisi comme solution la réduction (1) des vecteurs *nD* vrs *2D* et de les stocker dans une table d'hachage (2) pour conserver les données et éliminer le problème de perte d'information.

5.3.1 Réduction des données :

Pour résoudre le problème de dimensionnalité on a choisi la réduction de dimensionnalité des données en utilisant la méthode ***Non-Negative Matrix Factorization (NMF)*** cette dernière a le principe trouver deux matrices non négatives (W, H) dont le produit se rapproche de la matrice non négative X. Cette factorisation peut être utilisée par exemple pour la réduction de la dimensionalité, la séparation à la source ou l'extraction thématique.
La fonction objective est : [105]

$$
\begin{aligned}
&0.5 * ||X - WH||\_Fro^2 \\
&+ alpha * l1\_ratio * ||vec(W)||\_1 \\
&+ alpha * l1\_ratio * ||vec(H)||\_1 \\
&+ 0.5 * alpha * (1 - l1\_ratio) * ||W||\_Fro^2 \\
&+ 0.5 * alpha * (1 - l1\_ratio) * ||H||\_Fro^2
\end{aligned}
$$

Où :

$$
\begin{aligned}
&||A||\_Fro^2 = \sum\_{i,j} A\_{ij}^2 \text{ (Frobenius norm)} \\
&||vec(A)||\_1 = \sum\_{i,j} abs(A\_{ij}) \text{ (Elementwise L1 norm)}
\end{aligned}
$$

5.3.2 *Table d'hachage :*

Afin de garder l'information et ne pas la perte nous proposons le stockage dans une table d'hachage tel qu'elle est un type de structure de données qui stocke des paires clé-valeur. La clé est envoyée à une fonction de hachage qui effectue des opérations arithmétiques sur celle-ci. Le résultat (communément appelé valeur de hachage ou hachage) est l'index de la paire clé-valeur dans la table de hachage. [103]

Complexité d'hachage :

Tableau 7 : Complexité de tableau d'hachage [103]

Opération	**Meilleur des cas**	**Pire des cas**
Recherche	*O(1)*	*O(n)*
Insertion	*O(1)*	*O(n)*
Suppression	*O(1)*	*O(n)*

5.4 *Indexation : Combinaison du graphe de Voronoï et des arbres :*

Afin d'améliorer le coût de construction, nous proposons le graphe de Voronoï au lieu des algorithmes *k*-means et DBSCAN, qui ont un coût élevé.
Tout d'abord, nous allons sélectionner un échantillon (4) de nos données et les indexer en Voronoï (5), qui [21] divise l'espace en régions disjointes. Chaque région disjointe, appelée cellule de Voronoï, est un polygone convexe. Chaque cellule de Voronoï peut avoir un certain nombre de cellules de Voronoï environnantes, dont chacune partage au moins un côté/une arête avec cette cellule. Par définition du diagramme de Voronoï, le centre de chaque cellule de Voronoï est occupé par un point de données (POI) dans l'ensemble de données. En d'autres termes, chaque cellule de Voronoï doit contenir un point de données (POI) qui est centré dans la cellule de Voronoï. De plus, selon les propriétés du diagramme de Voronoï, le plus proche voisin de chaque point de requête situé dans la zone de la cellule de Voronoï doit être le point de données (POI) correspondant qui est centré dans la cellule de Voronoï. Si un point d'interrogation se trouve sur le côté de deux cellules de Voronoï voisines, alors les distances entre le point d'interrogation et les deux points de données dans les deux cellules de Voronoï sont les mêmes.

Nous présentons ses sommets sous forme de clusters (6) et exécutons une fonction de distance qui est une fonction métrique ou de distance est une fonction d(x, y) qui définit la distance entre les éléments d'un ensemble. Elle est généralement utilisée pour calculer la similarité entre des points de données [102]. Dans cette étude, l'objectif est de réduire le nombre d'opérations effectuées pour répondre à une requête donnée et nous utilisons cette fonction pour calculer la distance entre les autres vecteurs du dataset et les clusters créent (7), afin d'indexer ces vecteurs aux clusters les plus proches selon la structure arborescente (8).

Le processus de construction d'un arbre consiste à insérer des objets de haut en bas. Tout d'abord, le premier ensemble de données arrive. L'algorithme de recherche à deux axes de l'extérieur est utilisé pour tous les objets afin de diviser le conteneur en deux conteneurs de sorte que chaque élément du conteneur appartienne à son cluster le plus proche. Ensuite, cet algorithme est utilisé pour les conteneurs à deux feuilles, ce qui transforme la feuille en un nœud interne, et crée deux nœuds feuilles.

La figure 12 montre notre proposition d'indexation.

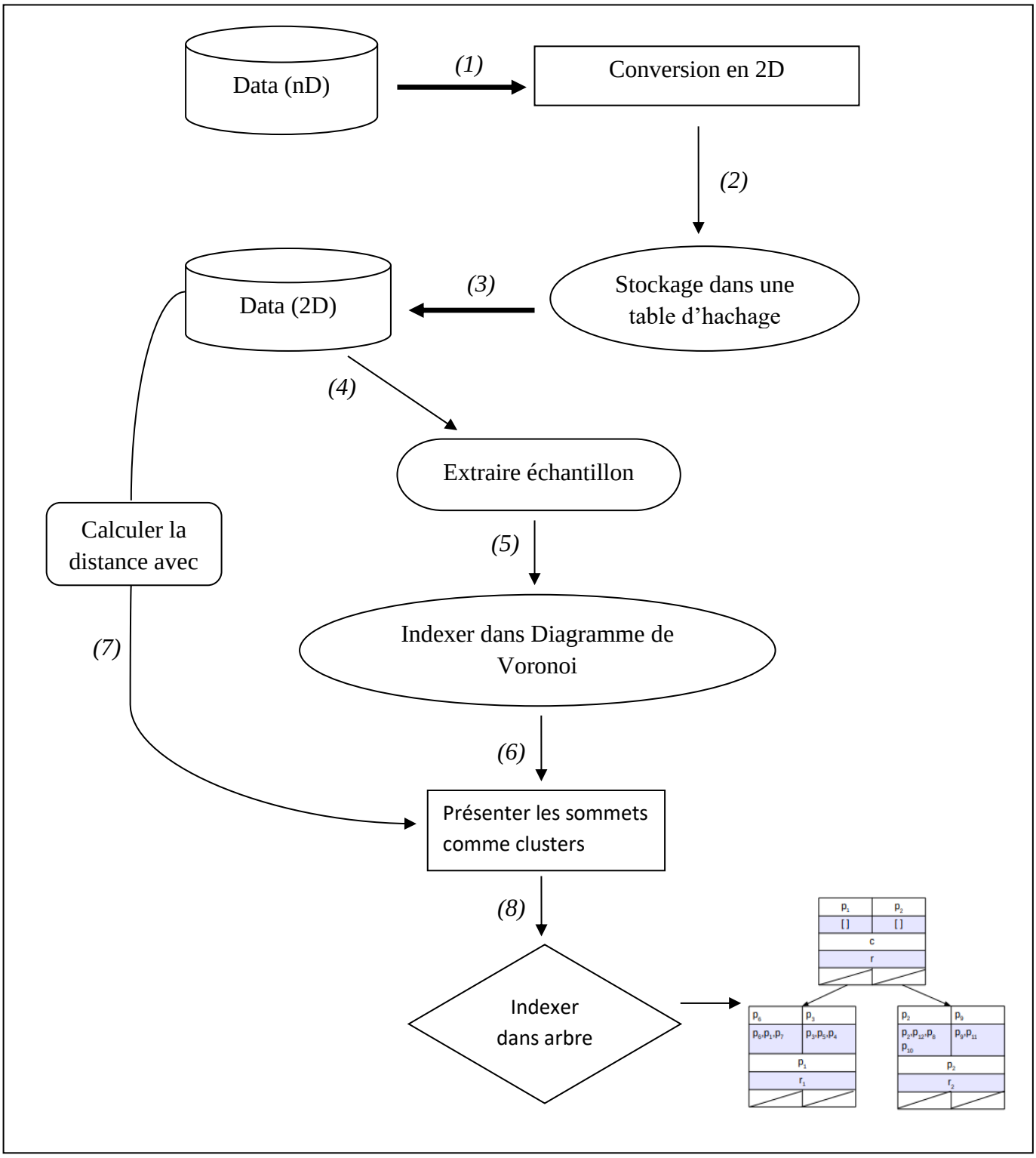


Figure 12 : Proposition de schéma explicatif de notre système.

5.5 La recherche Knn :

Dans cette section, nous décrivons une méthode de recherche permettant d'accéder rapidement aux données. La procédure de recherche dans notre cadre se compose de deux sections :

i. ***Section 1 :*** *Estimation d'arbre la plus proche que l'objet recherché*

1. Réduire le vecteur nD vers un vecteur 2D :

Nous utilisons la méthode ***Non-Negative Matrix Factorization (NMF)*** de réduction qui nous donne le vecteur 2D qui correspond à notre vecteur de plusieurs dimensions. Afin de lancer la fonction de recherche.

2. Calculer la distance entre le vecteur et les clusters :

Nous utilisons une fonction de distance entre le vecteur réduit et les clusters existants, de telle sorte que la distance minimale représente l'arbre qui a la plus forte probabilité de contenir ce vecteur.

ii. ***Section 2 :*** *lancer la recherche knn sur l'arbre obtenue.*

Dans une requête de plus proches voisins, nous recherchons les k objets les plus similaires à un élément donné. La requête n'est rien d'autre qu'un élément de référence q et le nombre d'éléments recherchés k.

L'ensemble des réponses n'est jamais vide, sauf dans des cas particuliers triviaux. En outre, sa taille est définie au préalable par l'utilisateur. Ce type de requêtes peut être formalisé par la définition suivante :

Soit (O, d) un espace métrique. Soit $q \in E$ un élément requête. Soit $k \in N$ le nombre recherché de réponses. Alors $NN(O, d, q, k)$ définit une requête kNN, dont la valeur est $S \subseteq E$ de telle sorte que $|S| = k$ (sauf si $|E| < k$) et $\forall (s, e) \in S \times E, d(q, s) \leq d(q, e)$.[21]

Dans le cas où plusieurs objets se trouvent à la même distance de la requête, le choix de fait d'une manière arbitraire

On lance l'algorithme de recherche Knn (Algorithme 1) [108] dans l'arbre la plus proche obtenue dans la section 1. Par l'utilisation de la fonction de distance entre le vecteur recherché et les deux pivots respectivement, tout en descendant dans l'arbre.

On trouve ensuite un ensemble de distance calculé. Pour trouver les plus proches voisins d'une donnée, il suffit de trier ces distances là en fonction de leurs distances croissantes par l'utilisation d'une fonction de tri.
Comme résultat de la recherche, les premiers objets triés dans une liste sont retournés. Il doit sélectionner les k e objets les plus proches.

Algorithm 1 : K nearest-nieghbor algorithm (KNN) [108]

Input : *X, Training data ; y, object to be classified*
Output : *Classification for y*

1. *For each object a ϵ X do*
2. *Compute d(y,a), the distance between y and a*
3. *End for*
4. *Select N⊆X, the set of k closest trainig objects for y*
5. *Class(y)◄— max of classes (k closest objects)*
6. *Output Class*

La figure 13 montre notre structure proposée dans la phase de recherche :

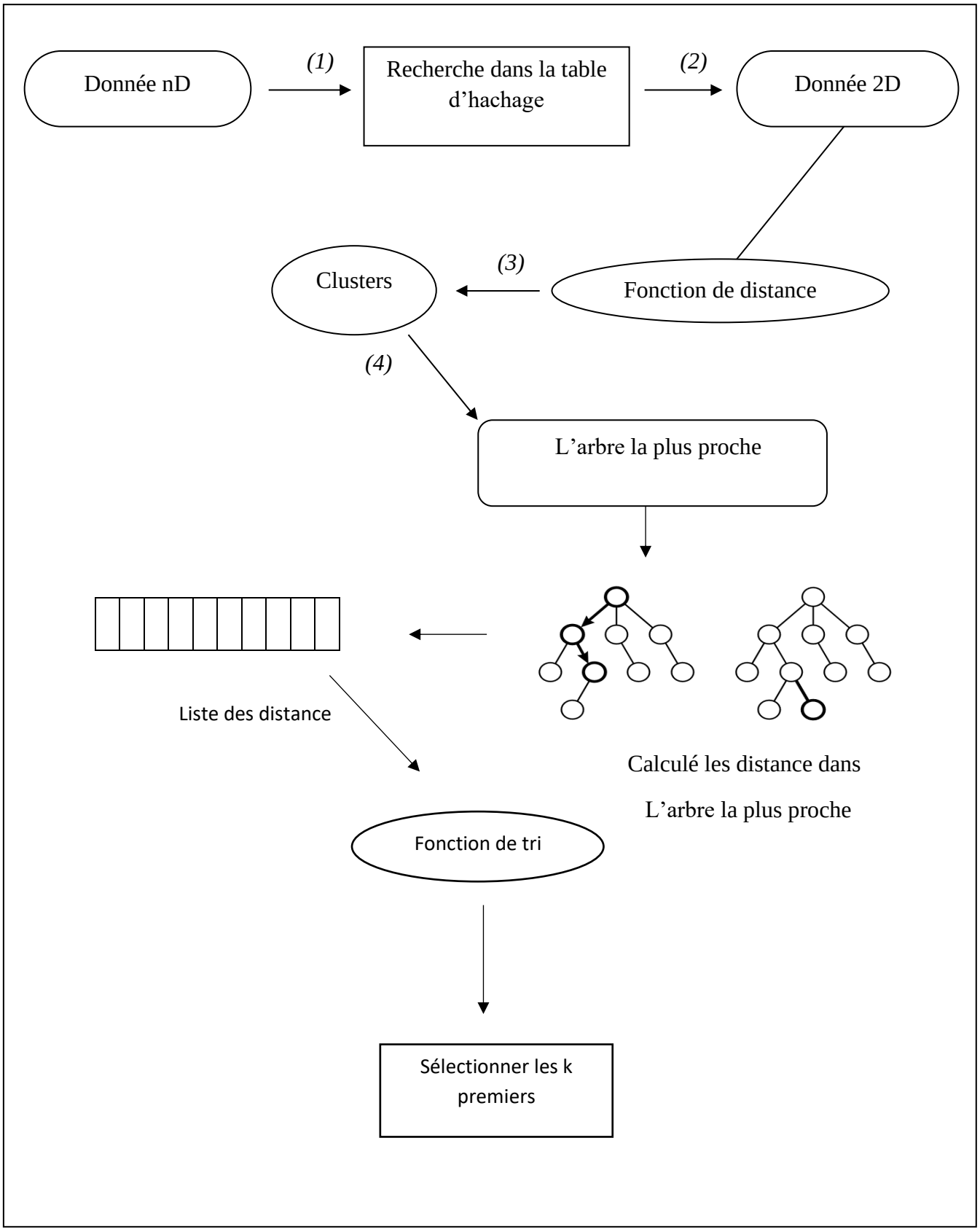


Figure 13 : Architecture d'algorithme de recherche

5.6 Conclusion :

Ce chapitre a été consacré à la partie conception, nous avons présenté la méthode de prétraitement (conversion de données et hachage), et la structure d'indexation proposée (combinaison entre le graphe et l'arbre) pour gérer les données IoT massives. L'objectif principal de cette architecture est de développer un nouveau mécanisme efficace pour le stockage et la récupération des données.

Chapitre 06 :
Implémentation et résultat

6.1 Introduction :

Notre proposition vise à minimiser le coût de d'indexation des données et à répondre efficacement aux requêtes de recherche, Dans ce chapitre, nous fournissons une évaluation expérimentale de cette structure. De plus, nous comparons cette structure avec les méthodes d'indexation existantes dans les espaces métriques, à savoir BCCF.

6.2 Collections indexées :

Pour démontrer l'efficacité de notre approche, nous avons réalisé toutes les expériences sur des données réelles avec des distributions de données sensiblement différentes afin de démontrer l'applicabilité étendue de notre indice. Trois jeux de données différents ont été utilisés pour les expériences et le tableau 08 ci-dessous présente quelques caractéristiques de ces jeux de données.

Dataset 1 (Tracking) : Le premier ensemble de données représente un ensemble de vecteurs de caractéristiques d'objets mobiles obtenus par un simulateur de suivi d'objets utilisant des caméras sans fil dans le multimédia réseau sans fil de capteurs lors d'une simulation aléatoire [64].

Dataset 2 WARD (Wearable Action Recognition Database) : Le deuxième ensemble de données représente une base de données pour la reconnaissance d'activités humaines à l'aide de capteurs portables. [106]

Tableau 8 : Caractéristiques des datasets utilisées

Dataset	***Nombre d'objet***	***Dimension***	***Echantillon***
Tracking a moving object dataset	62 702	20	100
WARD (Wearable Action Recognition Database)	1 000 000	5	1000

6.3 ***Evaluation de l'algorithme de construction :***

Pour prouver l'efficacité de notre proposition combinaison entre graph et arbre (GA), nous la comparons avec la structure BCCF* et BCCF, en utilisant deux critères :

- Le nombre de distance calculé

- Le nombre de comparaison effectué

Nous avons expérimenté sur huit prototypes (les six versions BCCF∗ + BCCF+GA) pour les ensembles de données précédents et avec les mêmes paramètres.

Traking

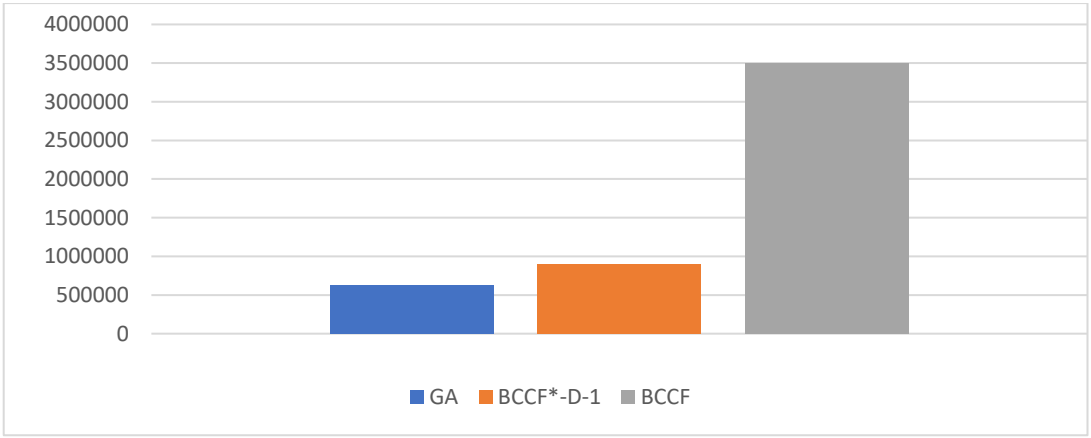
Temp de creation de Voronoi	***Complexité de creation de voronoi***
0,004 s	*O (n log n) = 6,64*

Ward

Temp de creation de Voronoi	***Complexité de creation de voronoi***
0,019 s	*O (n log n) = 9,96*

Figure 14 : temp et complexité de la création de voronoi

Traking



Ward

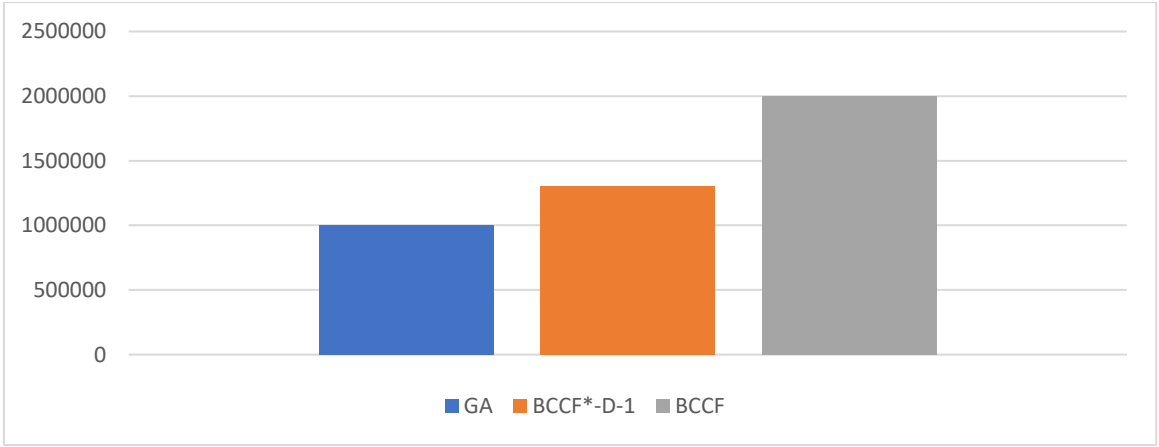
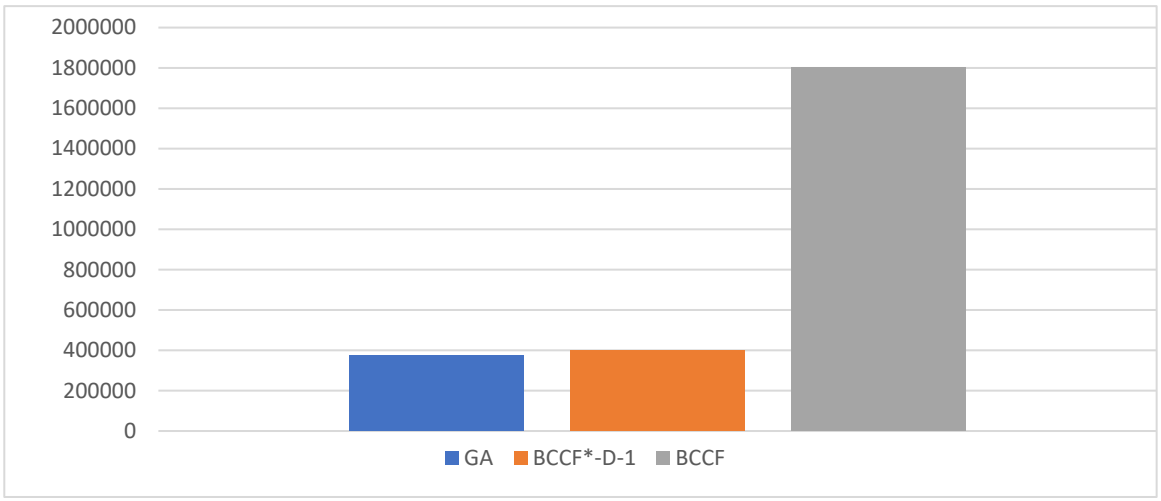


Figure 15 : Nombre des distances calculés

Traking



Ward

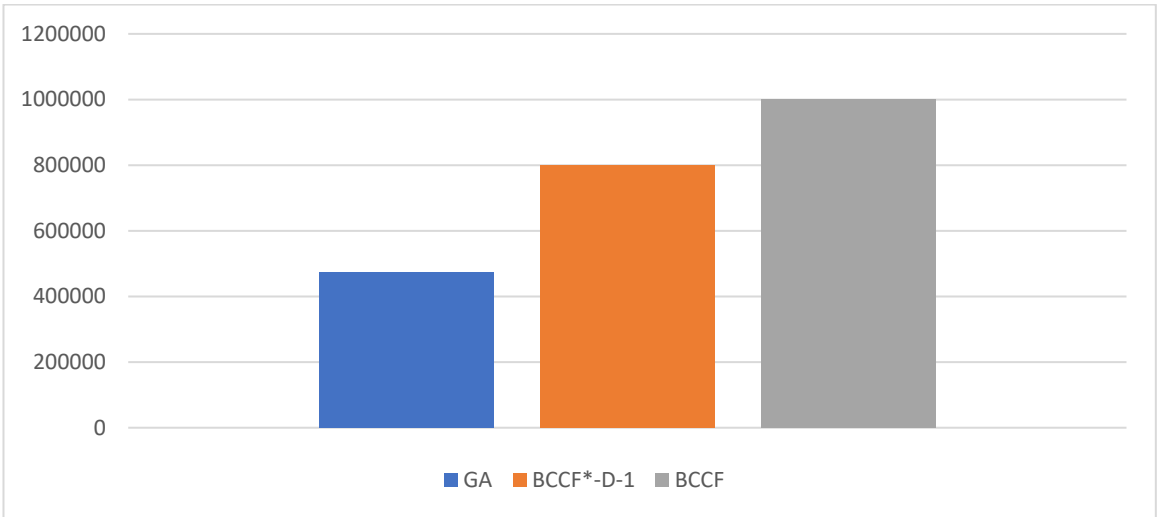


Figure 16 : Nombre des Comparaisons effectués

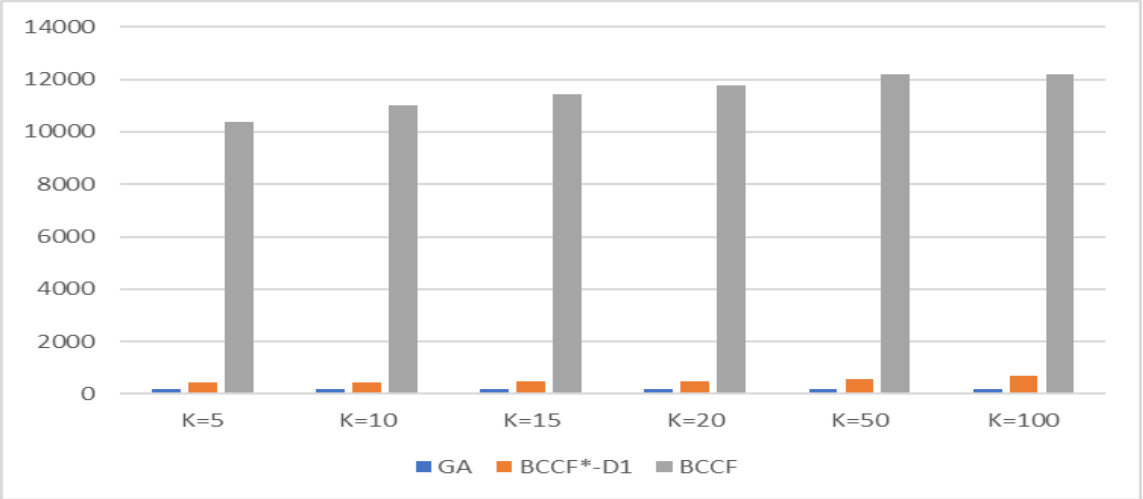
Les figures 15 et 16 décrivent le nombre de calculs de distance et le nombre de comparaisons pour le GA (combinaison des graphes et arbres). Dans les figures 15,16 nous constatons clairement que la proposition GA est la plus performante, par rapport aux autres propositions. Ceci est important et prouve que notre proposition est compétitive avec la version précédente dans son algorithme de construction.

6.4 Evaluation de la recherche Knn :

Nous sommes arrivés à la phase d'expérimentation de l'algorithme de recherche kNN. Pour prouver l'efficacité de notre proposition combinaison entre graph et arbre(GA), nous la comparons avec la structure BCCF* et BCCF, en utilisant ces critères :

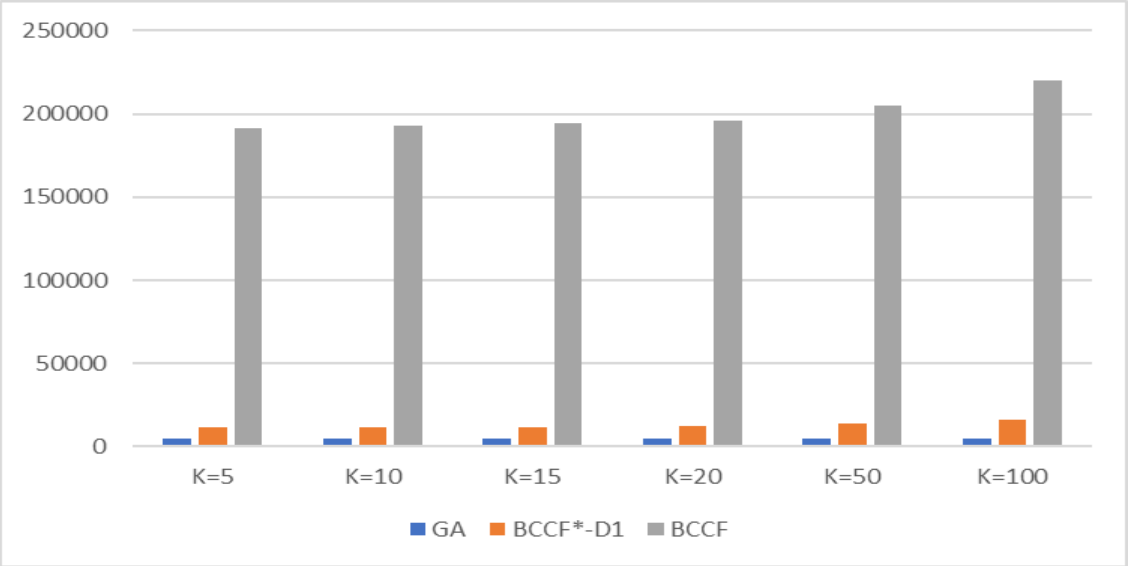
- Le nombre moyen de distances calculées
- Le nombre moyen de comparaisons effectuées
- Le temps exact de la recherche.

Traking



	GA	***BCCF*-D1***	***BCCF***
K=5	190	442	10375
K=10	190	457	11001
K=15	190	472	11420
K=20	190	487	11787
K=50	190	577	12188
K=100	190	683	12188

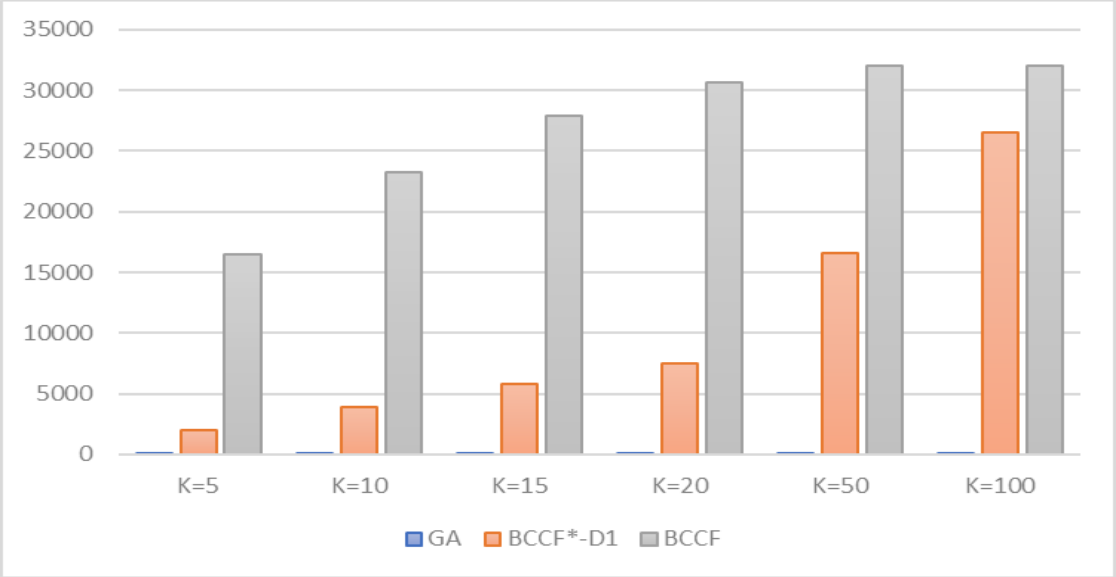
Ward



	GA	***BCCF*-D1***	***BCCF***
K=5	4843	11268	191421
K=10	4843	11609	192971
K=15	4843	11949	194519
K=20	4843	12277	196064
K=50	4843	14075	205301
K=100	4843	16391	220344

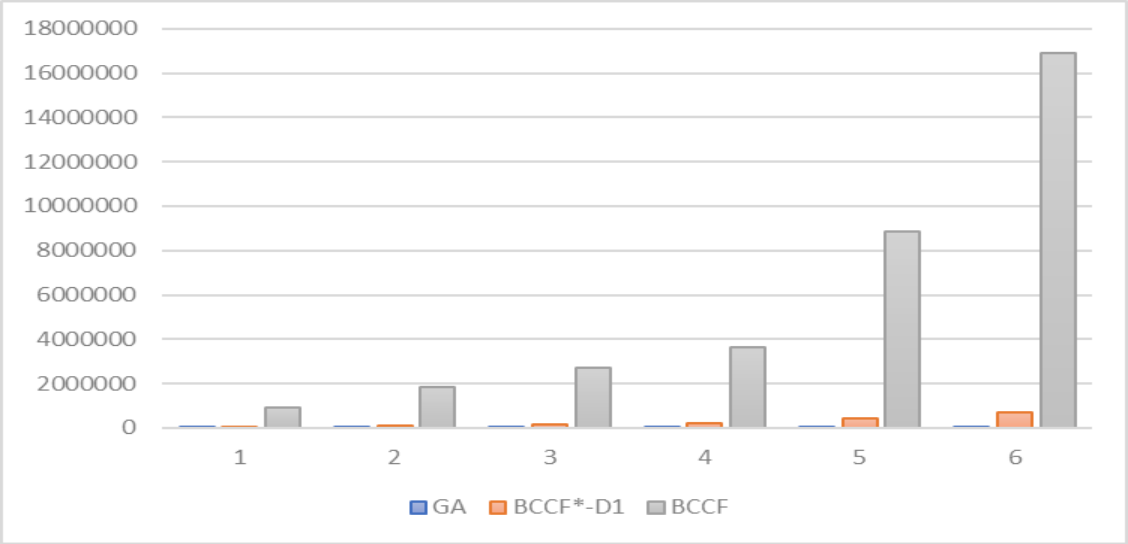
Figure 17 : nombre de distance calculé pour la recherche Knn

Traking



	GA	***BCCF*-D1***	***BCCF***
K=5	80	1997	16447
K=10	80	3912	23235
K=15	80	5752	27911
K=20	80	7517	30606
K=50	80	16532	32025
K=100	80	26547	32025

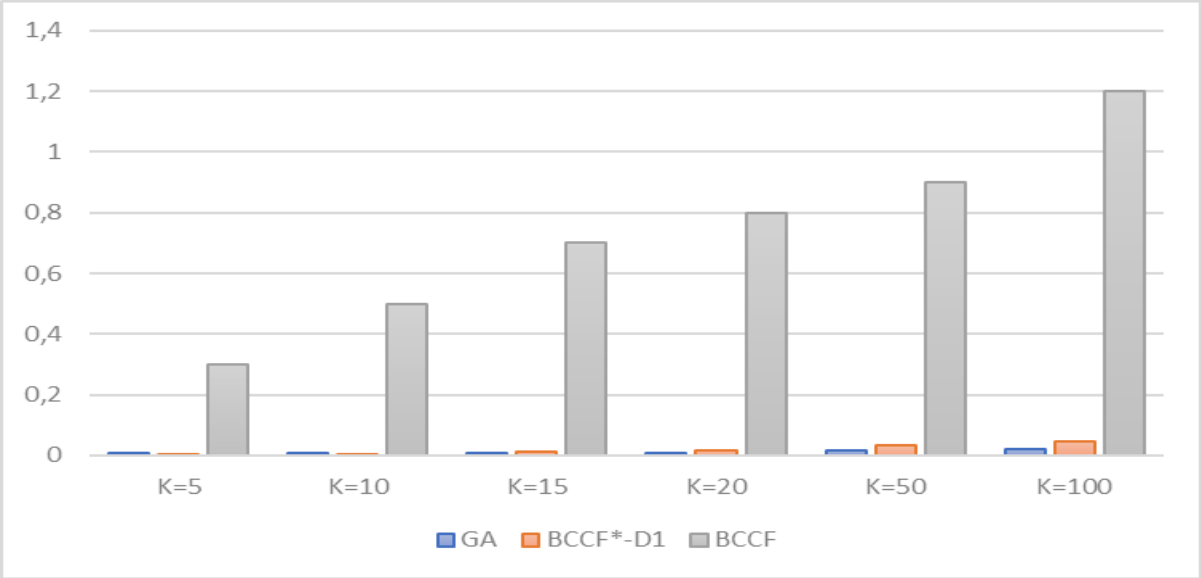
Ward



	GA	***BCCF*-D1***	***BCCF***
K=5	20290	50660	922127
K=10	20290	99285	1831442
K=15	20290	146210	2733010
K=20	20290	191466	3626850
K=50	20290	430391	8828086
K=100	20290	724564	16888098

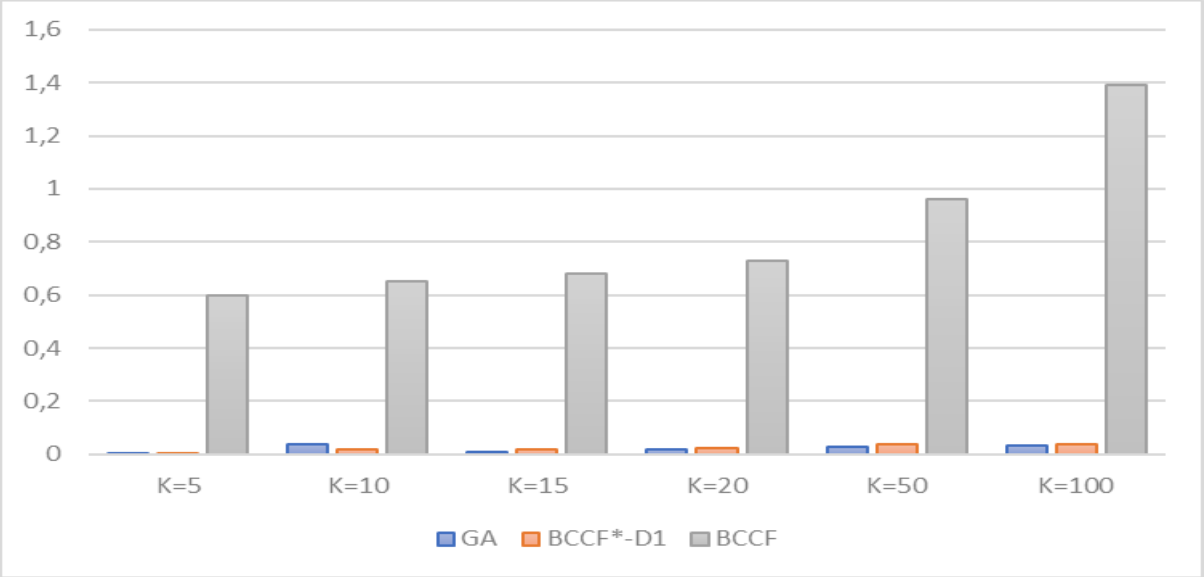
Figure 18 : nombre de comparaison effectué pour la recherche Knn

Traking



	GA	***BCCF*-D1***	***BCCF***
K=5	0,006	0,004	0,3
K=10	0,007	0,005	0,5
K=15	0,009	0,011	0,7
K=20	0,010	0,016	0,8
K=50	0,016	0,032	0,9
K=100	0,021	0,045	1,2

Ward



	GA	***BCCF*-D1***	***BCCF***
K=5	0,003	0,002	0,6
K=10	0,04	0,02	0,65
K=15	0,011	0,021	0,68
K=20	0,02	0,025	0,73
K=50	0,029	0,037	0,96
K=100	0,032	0,039	1,39

Figure 19 : temp de la recherche Knn (seconde)

La figure 16 et la figure 17 montrent le nombre de distances et de comparaisons dans les cas de KNN (k = 5, 10, 15, 20, 50, et100) successivement.

La figures 18 montre le temps de la recherche dans GA de KNN (k=5, 10, 15, 20, 50, et 100) successivement. En fonction de l'ensemble des résultats des figures précédentes, les conclusions suivantes peuvent être tirées :

- Le nombre moyen de distance calculé et de comparaison effectué reste le même pour n'importe quel k parce que on a calculé les distances et on les trier dans une liste avec une fonction prédéfinie dans python qui a une complexité de (n log n)
- Notons également, en comparant les trois méthode BCCF, BCCF∗− D − 1 et GA : la combinaison entre Graphe et arbre est moins coûteuse que l'arbre BCCF∗− D − 1. Et l'arbre BCCF est le plus coûteux.
- En comparant maintenant l'arbre BCCF, BCCF∗− D − 1 avec GA au temps de recherche, nous constatons qu'il y a un avantage en moyenne de notre approche par rapport au temps de recherche.

6.5 Conclusion :

Dans ce chapitre, nous avons présenté une nouvelle approche de l'indexation dans les espaces métriques, une méthode appelée "GA" (combinaison entre indexation arborescente et graphique). Nous avons proposé des algorithmes de construction basés sur l'estimation du chevauchement et des méthodes de décision. Par la suite, nous avons proposé un algorithme de recherche des k plus proches voisins. Dans la première proposition de la phase de construction, le GA a montré ses performances par rapport au BCCF∗− D − 1 et aussi par rapport à la version originale de BCCF. Dans la deuxième partie, nous avons montré, également par des expériences, que l'algorithme de recherche des k plus proches voisins dans GA est toujours le plus performant sur les deux collections utilisées dans cette étude.

Conclusion générale :

Cette étude a pour premier objectif d'optimiser l'indexation des données en minimisant le coût de construction de l'index tout en garantissant la performance de l'indexation, ainsi que la flexibilité et la rapidité de la recherche d'information sur les données Ido.

Pour atteindre cet objectif, nous avons appliqué le graphe de Voronoi à un échantillon de l'ensemble des données avant l'indexation arborescente sur les clusters obtenus par Voronoi. La structure d'index que nous proposons a montré des résultats expérimentaux intéressants et compétitifs, tant dans la construction que dans la récupération de requêtes similaires. Cela pourrait être une meilleure alternative pour l'indexation et la récupération des données Ido.

Les perspectives que nous pouvons tirer de ce travail pour les travaux futurs sont les suivantes :

- Améliorer le temps des algorithmes de recherche tout en étant basé sur un environnement bien contrôlé.
- Mise en application de ces techniques dans des domaines précis.

Bibliographie :

[1] : Christian Borgelt, and Michael R. Berthold. “Mining Molecular Fragments: Finding Relevant Substructures of Molecules”, ICDM, 2002, pp. 51-58.

[2] : Huan, W. Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A. Tropsha, “Mining Protein Family Specific Residue Packing Patterns from Protein Structure Graphs”, In Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB), 2004, pp. 308-315.

[3] : B. T. Messmer, and H. Bunke, “A Decision Tree Approach to Graph and Subgraph Isomorphism Detection”, Pattern Recognition, December 1999, Vol. 32, No. 12, pp. 1979-1998

[4] : S. Raghavan and H. Garcia-Molina, “Representing Web Graphs”, In Proceedings of the IEEE Intl. Conference on Data Engineering, 2003.

(chap 3)

[5] : Luigi Atzori, Antonio Iera et Giacomo Morabito. “The internet of things : A survey”. In : Computer networks 54.15 (2010), p. 2787-2805.

[6] : internet des objets. • Number of IoT devices 2015-2025 | Statista (date d’accées : 28/03/2022)

[7] : L.C. Shimomura, R.S. Oyamada, M.R. Vieira et al., A survey on graph-based methods for similarity searches in metric spaces, Information Systems (2020) 101507, https://doi.org/10.1016/j.is.2020.101507.

[8] : Perry Xiao. Designing Embedded Systems and the Internet of Things (IoT) with the ARM mbed. John Wiley & Sons, 2018.

[9] : ZHANG Ying-conga, YU Jingb, — A Study on the Fire IOT Development Strategy‖, Int. J. of Safety and Security Eng., Vol. 4, No. 2 (2014) 135–142

[10] : Debasis Bandyopadhyay , Jaydip Sen ,” Internet of Things - Applications and Challenges in Technology and Standardization”,Wireless Pers Commum,9 April 2011.

[11] : Rafiullah Khan et al. “Future internet : the internet of things architecture, possible applications and key challenges”. In : 2012 10th international conference on frontiers of information technology. IEEE. 2012, p. 257-260.

[12] : Luis M Vaquero et al. A break in the clouds : towards a cloud definition. 2008.

[13] : Peter Mell, Tim Grance et al. “The NIST definition of cloud computing”. In : (2011).

[14] : Flavio Bonomi et al. “Fog computing and its role in the internet of things”. In : Proceedings of the first edition of the MCC workshop on Mobile cloud computing. 2012, p. 13-16

[15] : Mostafa Haghi Kashani, Amir Masoud Rahmani et Nima Jafari Navimipour : Quality of service-aware approaches in fog computing. International Journal of Communication Systems, page e4340, 2020.

[16] : Mohamed Firdhous, Osman Ghazali et Suhaidi Hassan. “Fog computing : Will it be the future of cloud computing ?” In : The Third International Conference on Informatics & Applications (ICIA2014). 2014

[17] : What is edge computing ?, howpublished = https://www.cloudflare.com/ learning/serverless/glossary/what- is- edge- computing/, note = Accessed : 15/04/2022.

[18] : Volker Gaede et Oliver Günther. “Multidimensional access methods”. In : ACM Computing Surveys (CSUR) 30.2 (1998), p. 170-231.

[19] : Mohamed Amine Ferrag et al. “Big IoT Data Indexing : Architecture, Techniques and Open Research Challenges”. In : 2019 International Conference on Networking and Advanced Systems (ICNAS). IEEE. 2019, p. 1-6.

[20] : Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In : Communications of the ACM 18.9 (1975), p. 509-517

[21] : Zineddine Kouahla. “Indexation dans les espaces métriques Index arborescent et parallélisation”. Thèse de doct. Université de Nantes, 2013.

[22] : MP PaoloCiaccia : M-tree : An efficient access method for similarity search in metric spaces. In Proceedings of the 23rd VLDB Conference, Athenes, Greece, pages 357–368, 1997.

[23] : Caetano Traina, Agma Traina, Bernhard Seeger et Christos Faloutsos : Slim-trees : High performance metric trees minimizing overlap between nodes. In International Conference on Extending Database Technology, pages 51–65. Springer, 2000.

[24] : Jörg P Bachmann. “The SuperM-Tree : Indexing metric spaces with sized objects”. In : arXiv preprint arXiv :1901.11453 (2019).

[25] : Shichao Jin, Okhee Kim et Wenya Feng. “M X-tree : A Double Hierarchical Metric Index with Overlap Reduction”. In : International Conference on Computational Science and Its Applications. Springer. 2013, p. 574-589.

[26] : Stefan Berchtold, Daniel A Keim et Hans-Peter Kriegel. “The X-tree : An index structure for high-dimensional data”. In : Very Large Data-Bases. 1996, p. 28-39.

[27] : Lu CHEN et al. “Pivot-based Metric Indexing.(2017)”. In : Proceedings of the VLDB Endowment : 43rd International conference, Munich Germany, 2017 August 28-September. T. 1.

[28] : Gonzalo Navarro et Nora Reyes. “New dynamic metric indices for secondary memory”. In : Information Systems 59 (2016), p. 48-78.

[29] : Safia Brinis, Caetano Traina et Agma JM Traina. “Hollow-tree : a metric access method for data with missing values”. In : Journal of Intelligent Information Systems 53.3 (2019), p. 481-508.

[30] : Peter N Yianilos : Data structures and algorithms for nearest neighbor search in general metric spaces. In Soda, volume 93, pages 311–21, 1993.

[31] : Ives Rene Venturini Pola, Caetano Traina et Agma Juci Machado Traina : The mm-tree : A memory-based metric tree without overlap between nodes. In East European Conference on Advances in Databases and Information Systems, pages 157–171. Springer, 2007.

[32] : Caio César Mori Carélo, Ives Renê Venturini Pola, Ricardo Rodrigues Ciferri, Agma Juci Machado Traina, Caetano Traina Jr et Cristina Dutra de Aguiar Ciferri : Slicing the metric space to provide quick indexing of complex data in the main memory. Information Systems, 36(1):79–98, 2011.

[33] : Weiguo Wan et Hyo Jong Lee : Deep feature representation and balltree for face sketch recognition. International Journal of System Assurance Engineering and Management, pages 1–6, 2019.

[34] : Mohamad Dolatshah, Ali Hadian et Behrouz Minaei-Bidgoli : Ball*- tree : Efficient spatial indexing for constrained nearest-neighbor search in metric spaces. arXiv preprint arXiv :1511.00628, 2015.

[35] : Zineddine Kouahla, Adeel Anjum, Sheeraz Akram, Tanzila Saba et José Martinez : Xm-tree : data driven computational model by using metric extended nodes with non-overlapping in high-dimensional metric spaces. Computational and Mathematical Organization Theory, 25(2):196– 223, 2019

[36] : Jeffrey K Uhlmann. “Satisfying general proximity/similarity queries with metric trees”. In : Information processing letters 40.4 (1991), p. 175-179.

[37] : Sergey Brin. “Near neighbor search in large metric spaces”. In : (1995).

[38] : Zineddine Kouahla et Adeel Anjum. “A Parallel Implementation of GHB Tree”. In : IFIP International Conference on Computational Intelligence and Its Applications. Springer. 2018, p. 47-55.

[39] : Yuchai Wan, Xiabi Liu et Yi Wu : Cd-tree : A clustering-based dynamic indexing and retrieval approach. Intelligent Data Analysis, 21(2):243–261, 2017.

[40] : Lu Chen et al. “Efficient metric indexing for similarity search and similarity joins”. In : IEEE Transactions on Knowledge and Data Engineering 29.3 (2015), p. 556-571.

[41] : Zineddine Kouahla et José Martinez. “A new intersection tree for content-based image retrieval”. In : 2012 10th International Workshop on Content-Based Multimedia Indexing (CBMI). IEEE. 2012, p. 1-6.

[42] : Ives Renê Venturini Pola, Caetano Traina Jr et Agma Juci Machado Traina. “The NOBH-tree : Improving in-memory metric access methods by using metric hyperplanes with non-overlapping nodes”. In : Data & Knowledge Engineering 94 (2014), p. 65-88.

[43] : Keyu Yang et al. “Distributed similarity queries in metric spaces”. In : Data Science and Engineering 4.2 (2019), p. 93-108

[44] : Shi-guang Liu et Yin-wei Wei. “Fast nearest neighbor searching based on improved VP-tree”. In : Pattern Recognition Letters 60 (2015), p. 8-15.

[45] : Il’ya Markov. “VP-tree : Content-based image indexing”. In : Proceedings of the Spring Young Researcher’s. T. 7. 2007, 00268a.

[46] : Marcus Adamsson et Aleksandar Vorkapic. A comparison study of Kd-tree, Vp-tree and Octree for storing neuronal morphology data with respect to performance. 2016.

[47] : DT-Tri Nguyen et al. “An index scheme for similarity search on cloud computing using mapreduce over docker container”. In : Proceedings of the 10th International Conference on ubiquitous information management and communication. 2016, p. 1-6.

[48] : Tri DT Nguyen et Eui-Nam Huh. “An efficient similar image search framework for large-scale data on cloud”. In : Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication. 2017, p. 1-8.

[49] : Hua Cheng et al. “Distributed indexes design to accelerate similarity based images retrieval in airport video monitoring systems”. In : 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). IEEE. 2015, p. 1908-1912.

[50] : Zineddine Kouahla et José Martinez. “A new intersection tree for contentbased image retrieval”. In : 2012 10th International Workshop on ContentBased Multimedia Indexing (CBMI). IEEE. 2012, p. 1-6

[51] : Claudia Deco et al. “XM-Tree, a new index for Web Information Retrieval”. In : Journal of Computer Science & Technology 8 (2008).

[52] : Weiguo Wan et Hyo Jong Lee. “Deep feature representation and ball-tree for face sketch recognition”. In : International Journal of System Assurance Engineering and Management (2019), p. 1-6.

[53] : Ting Liu, Andrew W Moore et Alexander Gray. “Efficient exact k-NN and nonparametric classification in high dimensions”. In : Proceedings of the 16th International Conference on Neural Information Processing Systems. 2003, p. 265-272.

[54] : Bastian Leibe, Krystian Mikolajczyk et Bernt Schiele. “Efficient clustering and matching for object class recognition.” In : BMVC. 2006, p. 789-798.

[55] : Kouahla Zineddine, Ferrag Mohamed Amine et Anjum Adeel. “Indexing Multimedia Data with an Extension of Binary Tree–Image Search by Content– ”. In : International Journal of Informatics and Applied Mathematics 1.1 (), p. 47-55.

[56] : Phuc Do et Trung Hong Phan. “A Distributed M-Tree for Similarity Search in Large Multimedia Database on Spark”. In : Handbook of Research on Multimedia Cyber Security. IGI Global, 2020, p. 146-164.

[57] : Suman Saha. “Community Detection in Complex Network Metric Space Nearest Neighbor Search Low Rank Approximation and Optimality”. In : (2020).

[58] : CHAI KAH HIENG. HARDWARE BASED ACCELERATOR FOR DATABASE QUERY USING M-TREE. 2018.

[59] : Dilek Küçük Matcı et Uğur Avdan. “Comparative analysis of unsupervised classification methods for mapping burned forest areas”. In : Arabian Journal of Geosciences 13.15 (2020), p. 1-13.

[60] : Henrique Batista da Silva et al. “Video similarity search by using compact representations”. In : Proceedings of the 31st Annual ACM Symposium on Applied Computing. 2016, p. 80-83.

[61] : Tomás Skopal, Jaroslav Pokornỳ et Vaclav Snasel. “PM-tree : Pivoting Metric Tree for Similarity Search in Multimedia Databases.” In : ADBIS (Local Proceedings). 2004, p. 803-815.

[62] : Gonzalo Navarro et Nora Reyes. “New dynamic metric indices for secondary memory”. In : Information Systems 59 (2016), p. 48-78

[63] : Gang Chen et al. “Metric similarity joins using MapReduce”. In : IEEE Transactions on Knowledge and Data Engineering 29.3 (2016), p. 656-669.

[64] : Ala-Eddine Benrazek et al. “An efficient indexing for Internet of Things massive data based on cloud-fog computing”. In : Transactions on emerging telecommunications technologies 31.3 (2020), e3868

[65] : S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, An optimal algorithm for approximate nearest neighbor searching fixed dimensions, J. ACM 45 (6) (1998) 891–923.

[66] : K. Aoyama, K. Saito, T. Yamada, N. Ueda, Fast Similarity Search in Small-World Networks, Springer, Berlin Heidelberg, 2009, pp. 185–196.

[67] : K. Aoyama, A. Ogawa, T. Hattori, T. Hori, A. Nakamura, Zero-resource spoken term detection using hierarchical graph-based similarity search, in: IEEE Int’l Conf. on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014, pp. 7093–7097, http://dx.doi.org/10.1109/ICASSP.2014.6854976.

[68] : K. Aoyama, A. Ogawa, T. Hattori, T. Hori, Double-layer neighborhood graph based similarity search for fast query-by-example spoken term detection, in: Proc. of the IEEE Int’l Conf. on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 5216–5220.

[69] : M. Iwasaki, Pruned bi-directed k-nearest neighbor graph for proximity 141 search, in: Proc. of the Int’l Conf. Similarity Search and Applications (SISAP), 142 2016, pp. 20–33.

[70] : A. Ocsa, C. Bedregal, E. Cuadros-Vargas, A new approach for similarity queries using neighborhood graphs, in: Brazilian Symp. on Databases, 2007, pp. 131–142.

[71] : G. Toussaint, Proximity graphs for nearest neighbor decision rules: Recent progress, in: Proc. of the Symp. on the INTERFACE, 2002, pp. 17–20.

[72] : O.U. Florez, S. Lim, HRG: A Graph Structure for Fast Similarity Search in Metric Spaces, Springer, 2008, pp. 57–64.

[73] : B. Harwood, T. Drummond, Fanng: Fast approximate nearest neighbour graphs, in: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 5713–5722.

[74] : D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440–442, http://dx.doi.org/10.1038/30918.

[75] : Y. Malkov, A. Ponomarenko, A. Logvinov, V. Krylov, Approximate nearest neighbor algorithm based on navigable small world graphs, Inf. Syst. 45 (2014) 61–68.

[76] : Y. Malkov, A. Ponomarenko, A. Logvinov, V. Krylov, Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces, in: Proc. of the Int'l Conf. Similarity Search and Applications (SISAP), in: Lecture Notes in Computer Science, 7404, Springer, 2012, pp. 132–147, http://dx.doi.org/10.1007/978-3-642-32153-5_10.

[77] : Y.A. Malkov, D.A. Yashunin, Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, IEEE Trans. Pattern Anal. Mach. Intell. (2018) 1, http://dx.doi.org/10.1109/TPAMI.2018.2889473.

[78] : R. Paredes, E. Chávez, Using the k-nearest neighbor graph for proximity searching in metric spaces, in: String Processing and Information Retrieval SPIRE, Springer Berlin Heidelberg, 2005, pp. 127–138.

[79] : R. Paredes, E. Chávez, K. Figueroa, G. Navarro, Practical construction of k-nearest neighbor graphs in metric spaces, in: Experimental Algorithms: Int'l Workshop, WEA 2006, Cala Galdana, Menorca, Spain, May 24-27, 2006. Proc., Springer, Berlin Heidelberg, 2006, pp. 85–97.

[80] : W. Dong, C. Moses, K. Li, Efficient k-nearest neighbor graph construction for generic similarity measures, in: Proc. of The World Wide Web Conference (WWW), 2011, pp. 577.

[81] : Y.-M. Zhang, K. Huang, G. Geng, C.-L. Liu, Fast knn graph construction with locality sensitive hashing, in: Proc. of the Machine Learning and Knowledge Discovery in Databases (ECML-PKDD, 2013, pp. 660–674.

[82] : J. Chen, H.-R. Fangand, Y. Saad, Fast approximate knn graph construction for high dimensional data via recursive lanczos bisection, Journal of Machine Learning Research 10 (2009) 1989–2012

[83] : J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, S. Li, Scalable k-nn graph construction for visual descriptors, in: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 1106–1113.

[84] : M. Iwasaki, Pruned bi-directed k-nearest neighbor graph for proximity search, in: Proc. of the Int'l Conf. Similarity Search and Applications (SISAP), 2016, pp. 20–33.

[85] : K. Aoyama, K. Saito, H. Sawada, N. Ueda, Fast approximate similarity search based on degree-reduced neighborhood graphs categories and subject descriptors, in: ACM SIGKDD, 2011, pp. 1055–1063.

[86] : K. Aoyama, A. Ogawa, T. Hattori, T. Hori, Double-layer neighborhood graph based similarity search for fast query-by-example spoken term detection, in: Proc. of the IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 5216–5220.

[87] : A. Ocsa, C. Bedregal, E. Cuadros-Vargas, A new approach for similarity queries using neighborhood graphs, in: Brazilian Symp. on Databases, 2007, pp. 131–142.

[88] : O.U. Florez, X. Qi, A. Ocsa, Mobhrg: Fast k-nearest-neighbor search by overlap reduction of hyperspherical regions, in: Proc. of the IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP), 2009, pp. 1133–1136.

[89] : F.P Preparata and M.L Shamos. Computational Geometry. Spring-Verlag (1985).

[90] : F. Aurenhammer. Voronoi Diagrams – A Survey of a Faundamental Geometric Data Structure. ACM Computing Survey 23 (1991) 345-405.

[91] : K. Lee, K. Lee, Escaping your comfort zone: a graph-based recommender system for finding novel recommendations among relevant items, Expert Systems with Applications 42 (10) (2015) 4851 – 4858, http://dx.doi.org/ https://doi.org/10.1016/j.eswa.2014.07.024.

[92] : R. Angles, C. Gutierrez, Survey of graph database models, ACM Comput. Surv. 40 (1) (2008) http://dx.doi.org/10.1145/1322432.1322433.

[93] : R. Paredes, E. Chávez, Using the k-nearest neighbor graph for proximity searching in metric spaces, in: String Processing and Information Retrieval SPIRE, Springer Berlin Heidelberg, 2005, pp. 127–138.

[94] : A. Ocsa, C. Bedregal, E. Cuadros-Vargas, A new approach for similarity queries using neighborhood graphs, in: Brazilian Symp. on Databases, 2007, pp. 131–142.

[95] : Y. Malkov, A. Ponomarenko, A. Logvinov, V. Krylov, Approximate nearest neighbor algorithm based on navigable small world graphs, Inf. Syst. 45 (2014) 61–68.

[96] : W. Dong, C. Moses, K. Li, Efficient k-nearest neighbor graph construction for generic similarity measures, in: Proc. of The World Wide Web Conference (WWW), 2011, pp. 577.

[97] : CHONGSHENG ZHANG, GEORGE ALMPANIDIS, FAEGHEH HASIBI et GAOJUAN FAN, Gridvoronoi: An Efficient Spatial Index for Nearest Neighbor Query Processing, IEEE Access (2019)

[98] : M. Sharifzadeh and C. Shahabi, ‘‘Vor-tree: R-trees with Voronoi diagrams for efficient processing of spatial nearest neighbor queries,’’ Proc. VLDB Endowment, vol. 3, nos. 1–2, pp. 1231–1242, 2010.

[99] : S. Nutanong, R. Zhang, E. Tanin, and L. Kulik, ‘‘The v*-diagram: A querydependent approach to moving KNN queries,’’ Proc. VLDB Endowment, vol. 1, no. 1, pp. 1095–1106, 2008.

[100] : C. Li, Y. Gu, J. Qi, G. Yu, R. Zhang, and W. Yi, ‘‘Processing moving KNN queries using influential neighbor sets,’’ Proc. VLDB Endowment, vol. 8, no. 2, pp. 113–124, 2014

[101] : Xuesheng Zhao, Jun Chen and Renliang Zhao, « Dynamic Spatial Indexing Model Based on Voronoi », Proceedings of the International Symposium on Digital Earth (1999).

[102] : kemouguette Ibtissem, Optimisation de la structure d’indexation de l’arbre BCCF pour des données hétérogènes dans un environnement Edge/Fog/Cloud Computing, mémoire du master, université 08 mai 1945 Guelma (2021).

http://dspace.univ-guelma.dz/jspui/handle/123456789/11767

[103] : What is a hash table, https://fr.wikipedia.org/wiki/Table_de_hachage, accédé : 19/05/2022

[104] : A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, vol. 501. Hoboken, NJ, USA: Wiley, 2009.

[105] : w3cub, scikit_learn documentation, https://docs.w3cub.com/scikit_learn/modules/generated/sklearn.decomposition.nmf , accédé : 02/06/2022

[106]: bdd. https://people.eecs.berkeley.edu/yang/software/WAR/WARD1 . zip,2019. Accessed : 03/06/2022

[107] : Vaccari I, Chiola G, Aiello M, Mongelli M, Cambiaso E. MQTTset, a new dataset for machine learning techniques on MQTT. Sensors. 2020;20(22):6578.

[108] : Wenfeng Hou , Daiwei Li , Chao Xu , Haiqing Zhang , Tianrui L, « An Advanced Nearest Neighbor Classification Algorithm Based on KD-tree », 2018 IEEE International Conference of Safety Produce Informatization (IICSPI)

[109] : Fatima Zahra Fagroud, Sanaa Elfilali, Hicham Toumi et al. “IOT et Cloud Computing : état de l’art”. In : Colloque sur les Objets et systèmes Connectés. 2019.

[110] : Internet des objets. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. (date d’accès : 06/06/2022).