République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Abderrahmane Mira

Faculté de la Technologie

Département d'Automatique, Télécommunication et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Télécommunications

Spécialité : Réseaux et Télécommunications

Thème

Allocations de services dans un environnement de Fog Computing

Préparé par :

- BENNICHE Hakima
- BOUYAHMED Wassila

Dirigé par :

Mme Z. AZIZOU

Mme K. MAMMERI

Examiné par :

M. Diboune

M. Kasmi

Année universitaire : 2023/2024

Remerciements

Tout d'abord, nous remercions dieu qui nous a donné le courage, la volonté et la patience pour réaliser ce travail et achever cette noble formation.

Nous présentons nos plus grands et sincères remerciements à tous ceux qui nous ont apporté leur soutien et leur aide de près et de loin.

Nous voudrions exprimer nos remerciements sincères à notre promotrice Mme **AZIZOU Zahia** et à notre Co-promotrice Mme **MAMMERI Karima** pour leurs contributions à la réalisation de ce travail.

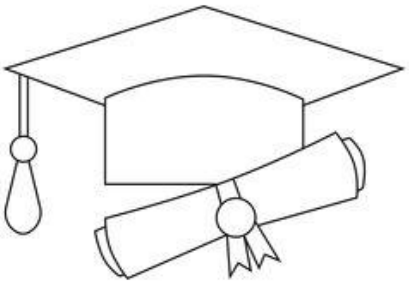
Nous voudrions également remercier :

- **Mr HADJI :** Notre chef de département.
- **Mr KASMI :** Notre chef de filière.
- **Mr TOUNSI :** Notre chef de spécialité.

.

Nous tenons aussi à exprimer nos remerciements à tous nos enseignants durant cette formation.

Et enfin, nous remercions les membres de jury d'avoir accepté de juger ce travail.



Dédicace

Je dédie ce travail :

À mes chers parents

« Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagnera toujours. Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices. Puisse ***Dieu****, le Très Haut, vous accorder santé, bonheur et longue vie et faire en sorte que jamais je ne vous décevrai. »*

À *mes chères sœurs :* ***A****icha et* ***C****hanez*

À *mon petit frère :* ***A****mir*

À *ma belle-sœur :* ***S****ouad*

À *toute la famille :* ***BOUYAHMED***

À *ma collègue :* ***B****enniche* ***H****akima*

À *tous mes amis qui m'ont soutenu et aidé de près ou de loin pour arriver là où je suis.*

À *mes tous les étudiants en Master2 Réseau et Télécom promo 2024*

E*t à tous mes enseignants.*

Wassila Bouyahmed

Dédicaces

Je dédie ce modeste travail :

À **mes chers parents**

Pour leur amour, leur soutien, leurs sacrifices, leurs encouragements et leurs prières tout au long de mes études. Que dieu les gardes et les protège.

À **mes chers frères Ali et Billal**

Pour leurs amour et soutien durant tout mon parcours.

À **ma sœur : Karima**

Qui a été toujours à mes côtés, m'a encouragé tout au long de mon parcours.

À **toute ma famille**

À **mes chers amis**

Qui m'ont soutenu de près ou de loin.

À **ma collègue : BOUYAHMED Wassila**

Pour son soutien moral, sa patience et compréhension tout au long de ce projet.

À **mes collègues** Master 2 RT de la promotion 2023/2024

Hakima Benniche

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

ABCI: Augmented Brain–Computer Interaction Game

ALG: Application Layer Gateway

AR: Augmented Reality

BPSO: Binary Particle Swarm Optimization

CD: Delay Sensitivity

CDN: Content Delivery Network

CL: Locality Requirement

CN: Network Constraints

COP: Combinatorial Optimization Problem

CPS: Cyber–Physical Systems

CPU: Central Processing Unit

CR: Resource Constraints

CT: Container

DAG: Directed Acyclic Graph

DM: Descent Method

EC: Edge Computing

EPC: Electronic Product Code

ETC: Expected Time to Compile / Complete

FC : Fog Computing

FEC: Fog/Edge Computing

FN: Fog Nodes

GPS: Global Positioning System

GRASP: Greedy Randomized Adaptive Search Procedure

IoT: Internet of Things

IoT-A: Internet of Things Architecture

IaaS: Infrastructure as a Service

ILP: Integer Linear Programming

LISP: Locator/ID Separation Protocol

MDC: Micro Data Center

MEC: Mobile Edge Computing

MI: Millions of Instructions

M2M: Machine-to-Machine

NAT: Network Address Translation

NFV: Network Functions Virtualization

NIC: Network Interface Card

OASIS: Organization for the Advancement of Structured Information Standards

OSGI: Open Service Gateway Iniative

PaaS: Platform as a Service

POP: Point of Presence

QoS: Quality of Service

RAM: Random Access Memory

RAN: Radio Access Network

RFID: Radio Frequency Identification

SA: Simulated Annealing

SaaS: Software as a Service

SA-DALO: Services Allocation based Discret Ant Lion Optimizer

SA-DSPSO: Services Allocation based on Discret Standard particle Swarms Optimization

SDN: Software Defined Networking

SPP: Service Placement Problem

S-PSO: Standard Particle Swarms Optimization

TS: Tabu Search

VM: Virtual Machines

VMM: Virtual Machines Monitor

VNIC: Virtual Network Interface Card

WSN: Wireless Sensor and Actuator Networks

Liste des Algorithmes

Introduction générale

Les technologies et les concepts numériques évoluent très rapidement et sont peu à peu en train d'articuler tous les secteurs de la vie quotidienne, en proposant des services allant du simple loisir aux services les plus critiques touchant à la santé ou à la sécurité des individus.

De nos jours, nous trouvons un peu partout des environnements auxquels des capteurs filaires et sans fils ont été intégrés et utilisés dans différents domaines comme la médecine (capteur de battements cardiaques), l'agriculture (détection de l'étanchéité du sol), météorologie (surveillance de température et de la vitesse de vent) et la garde forestier (la surveillance de feux de forêts)

Ces environnements dits intelligents sont combinés au Cloud Computing qui consiste à utiliser des serveurs informatiques distants par l'intermédiaire d'un réseau internet généralement pour stocker les données ou les exploiter, cependant pour répondre aux besoins changeants des applications modernes, en particulier dans le contexte de l'Internet des Objets et des applications sensibles à la latence , une extension de celui-ci est créée aux extrémités du réseau reconnue sous le nom de Fog Computing.

Le problème d'allocation de service dans le Fog Computing pose plusieurs défis qui nécessitent une approche holistique prenant en compte la distribution et l'hétérogénéité des ressources, la latence, la bande passante, l'équilibrage de charge, la sécurité, la mobilité et la consommation d'énergie. Des solutions efficaces à ce problème sont essentielles pour réaliser le plein potentiel du Fog Computing en termes de performance, de coût et de qualité de service.

L'allocation de services est cruciale pour prendre en charge les applications IoT (Internet of Things) sensibles au délai simultanément, pour gérer les requêtes et tâches d'une grande quantité d'objets répartis géographiquement, les ressources de calcul et de stockage de la couche Fog doivent être utilisées efficacement.

Cette allocation se base souvent sur des critères tels que la localisation des utilisateurs, la capacité de traitement des nœuds, les contraintes de latence et de bande passante, ainsi que les exigences de QoS (Quality of Service). Cependant, ce problème est considéré comme NP-

difficile en raison de sa nature combinatoire et de la multitude de facteurs à prendre en compte. En effet, le nombre de combinaisons possibles de services à allouer aux nœuds augmente de manière exponentielle avec le nombre de nœuds et de services disponibles, rendant la recherche d'une solution optimale impraticable en temps raisonnable pour des systèmes de taille significative.

Dans ce travail, nous proposons une étude sur l'allocation de services dans le Fog Computing. Notre étude se focalise sur l'utilisation de la métaheuristique ALO (Ant Lion Optimizer), pour résoudre le problème de l'allocation , la suite de ce mémoire est organisée comme suit : le chapitre 1 présentera les concepts d'IoT, Fog Computing et Cloud Computing et les notions fondamentales ainsi les différents algorithmes classiques et les différentes métaheuristiques d'optimisation, le chapitre 2 abordera le principe de l'allocation de services dans le Fog Computing, au niveau du chapitre 3 nous allons adapter la métaheuristique ALO pour optimiser l'allocation de services dans un environnement de Fog Computing pour en finir avec une étude comparative entre la métaheuristique proposée et SA-DSPSO.

1 Chapitre 1 : Généralités sur l'IoT, Fog et Cloud Computing

1.1 Introduction

Selon statista, il y'avait 42 milliards d'appareils IoT (Internet Of Things) dans le monde entier en 2022 et d'ici l'année 2025, ce nombre devrait dépasser environ 75 milliards d'appareils connectés. Tous ces appareils produiront une énorme quantité de données qui devront être traitées rapidement et de manière durable, Pour répondre à la demande croissante de solutions IoT, Fog Computing entre en action au même niveau que le Cloud.

Les infrastructures de Fog Computing constituent un élément majeur de l'Internet. Grâce à leur capacité de stockage et de calcul, ainsi que leur emplacement près des utilisateurs finaux, elles permettent de traiter les données à proximité de l'endroit où elles sont générées.

Dans ce chapitre, nous allons introduire des définitions et des explications de certaines notions utilisées dans le présent mémoire. Nous commencerons ce chapitre par expliciter l'IoT, Par la suite, nous verrons tous les détails sur le Cloud Computing et le Fog Computing.

1.2 Concepts de base

Pour poser les bases de notre compréhension commune, commençons par définir les termes clés et exposer les principes fondamentaux qui sous-tendent ce domaine

1.2.1 Edge computing

Comme son nom l'indique, Edge Computing (EC) désigne les calculs réalisés à la périphérie du réseau. EC vise à surmonter les limitations associées au modèle basé sur le Cloud Computing. Elle sert comme un intermédiaire entre les utilisateurs finaux/appareils et le Cloud, fournissant des fonctionnalités de traitement et de stockage à un grand nombre d'appareils IoT finaux **[1]**.

La proximité des dispositifs de périphérie permet de minimiser la charge de calcul sur les centres de données situés loin dans le Cloud. La réponse en temps réel est ainsi améliorée, tout comme la latence est réduite **[2,1]**.

Un autre avantage de EC est sa nature distribuée et sa prise en charge de la mobilité des appareils au sein de réseaux hétérogènes. En fait, des efforts de recherche importants sont

actuellement menés dans le domaine de l'informatique mobile en périphérie MEC (Mobile Edge computing). Selon **[2]**, la couche périphérique peut être mise en œuvre selon trois modes : le MEC, le FC (Fog Computing) et le Cloudet Computing (CC) **[1]**.

Le MEC est le déploiement de nœuds intermédiaires capables de traiter et de stocker des informations au sein de la station de base des réseaux mobiles cellulaires, permettant ainsi des fonctionnalités de Cloud Computing à l'intérieur du réseau d'accès radio (RAN) **[1]**.

Le Cloudlet est une version réduite du Cloud qui utilise des appareils dédiés offrant des fonctionnalités de type Cloud. Dans **[3]**, il a été démontré que le temps de réponse est réduit de 900 ms à 169 ms simplement en déplaçant le calcul du Cloud vers la périphérie dans une application de reconnaissance faciale.

Parmi les avantages de l'EC on peut citer **[1] :**

1. Dispersion géographique ;
2. Sécurité améliorée, car les données cryptées se déplacent plus loin dans le réseau central;
3. Fournit une meilleure réponse en temps réel que le modèle basé sur le Cloud;
4. Meilleure évolutivité grâce à la virtualisation;
5. Limite les goulots d'étranglement potentiels de la communication.

1.2.2 Fog/Edge Computing (FEC)

Il est important de noter que les dispositifs FC ne se situent pas nécessairement à la périphérie du réseau, mais plutôt à proximité de celle-ci. En revanche, les dispositifs de périphérie se trouvent souvent à la limite du réseau et constituent souvent le premier point de contact avec les dispositifs IoT finaux **[1]**.

En substance, les dispositifs FC et EC sont tous deux proches des dispositifs IoT finaux, mais les dispositifs EC sont souvent plus proches. Dans de nombreux travaux, les termes "Fog Computing" et "Edge Computing" ont été utilisés de manière interchangeable. Certains considèrent le FC comme un élément du paradigme de EC et du micro centre de données (MDC) pour l'IoT **[4,1].**

Les services de FC et d'EC sont tous les deux situés à proximité des utilisateurs finaux. Cependant, l'EC réside dans les dispositifs de périphérie, tandis que le FC réside dans les dispositifs de périphérie de réseau, généralement à un ou quelques sauts de réseau de la périphérie. La plateforme EC a une énergie limitée et un stockage restreint, et appartient à la classe des dispositifs contraints. L'augmentation du nombre d'applications IoT peut entraîner une concurrence plus élevée pour les ressources et une latence supplémentaire **[31].** En substance, la concurrence pour les ressources de l'EC est plus importante que celle du FC en raison de sa proximité avec les dispositifs IoT finaux. De plus, l'EC se concentre davantage sur le domaine des objets, tandis que le Fog Computing se concentre davantage sur le domaine de l'infrastructure **[1]**.

Le FEC repose sur certains piliers, notamment la sécurité, l'évolutivité, l'ouverture, l'autonomie, la fiabilité, l'agilité, l'organisation hiérarchique et la programmabilité, qui sont inhérents à la fois au FC et à l'EC. En tant que tel, la motivation pour intégrer à la fois le FC et l'EC est basée sur leurs particularités respectives :

1. Ils complètent tous deux les fonctionnalités offertes par le Cloud et sont situés entre les utilisateurs finaux et les centres de données.
2. Les deux peuvent être physiquement colocalisés avec des points d'accès, des unités de bord de route, des stations de base, des routeurs, des commutateurs et des passerelles.
3. Ils sont tous les deux principalement déployés sans fil et offrent une faible latence, une faible gigue et une cognition au sein du système.
4. Ils fournissent tous les deux des services de calcul dans des emplacements géographiques distribués afin de minimiser la charge sur le cloud.

En fonction de leurs fonctionnalités, le FC et l'EC peuvent se chevaucher et coexister au sein du systèmes cyber-physique CPS (Cyber–Physical Systems). La figure 1.1 **[1]** montre une représentation graphique du modèle FEC IoT avec différentes couches.

1.2.3 Virtualisation

La virtualisation consiste à créer une version virtuelle d'un dispositif ou d'une ressource, comme un serveur, un dispositif de stockage, un système d'exploitation ou une ressource réseau. Cela permet de partitionner un ordinateur ou serveur physiques en plusieurs machines

virtuelles. Chaque machine virtuelle peut alors interagir de manière indépendante, et exécuter différents systèmes d'exploitation ou applications tout en partageant les ressources d'un seul ordinateur hôte. En créant plusieurs ressources à partir d'un seul ordinateur ou serveur, la virtualisation améliore l'extensibilité et les charges de travail, tout en réduisant le nombre de serveurs utilisés, la consommation d'énergie, les coûts d'infrastructure et la maintenance **[29]**.

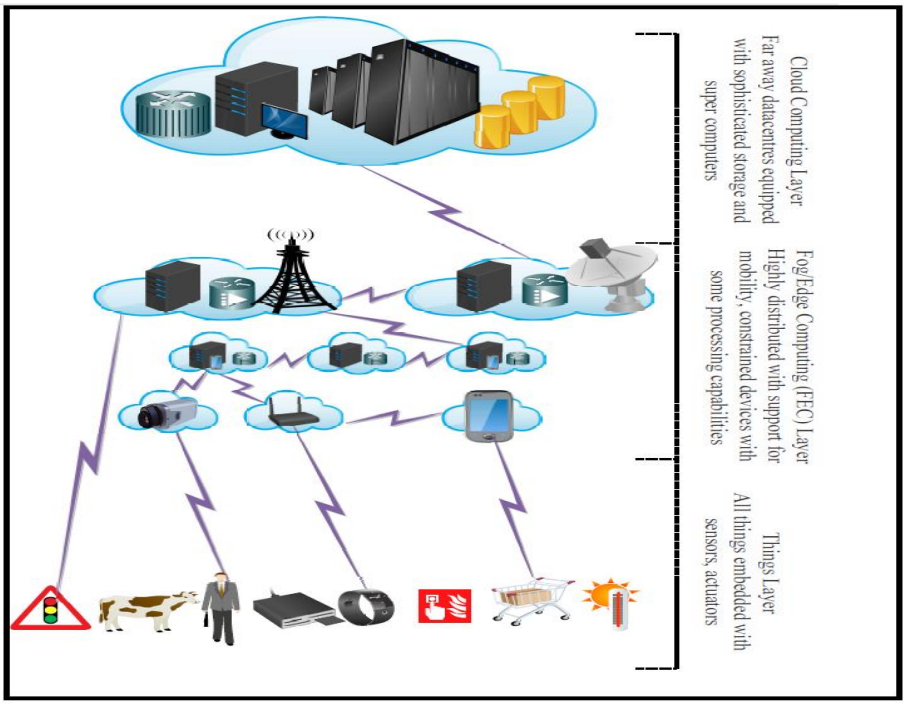


Figure 1.1 : Architecture FEC et son interaction dans le continuum Cloud-to-Things

1.2.4 Types de virtualisation

Dans la virtualisation, on peut distinguer quatre catégories. La première est la virtualisation serveur, le deuxième est la virtualisation applicative, le troisième est la virtualisation réseau, le quatrième est la virtualisation du stockage **[29]**.

1.2.4.1 Virtualisation serveur

La technique de virtualisation du serveur fait référence à l'utilisation d'un hyperviseur pour permettre à la machine hôte d'exécuter plusieurs instances virtuelles en même temps. Ces dernières sont communément appelées des VMs (Virtual Machines), tandis que l'hyperviseur

est connu sous le nom de *VMM* (Virtual Machines Monitor*),* c'est-à-dire gestionnaire de *VM* **[29].**

Ces nouvelles notions sont définies comme suit :

1. **Machine virtuelle (VM) :** est un conteneur logiciel totalement isolé capable d'exécuter son système d'exploitation et ses applications comme s'il s'agissait d'un véritable ordinateur. Une VM se comporte exactement comme un ordinateur physique. Elle contient son propre matériel virtuel : *CPU (*Central Processing Unit*),* mémoire *RAM,* disque dur et carte d'interface réseau *(NIC)* basés sur le logiciel **[29]**.

2. **Hyperviseur (*VMM)* :** est un programme qui permet à plusieurs systèmes d'exploitation de différents types (Linux, Mac ou Windows) de partager un seul hôte matériel. Il implante les mécanismes d'isolation et de partage des ressources matérielles **[29]**.

L'hyperviseur contrôle l'accès aux ressources en allouant à chaque machine virtuelle ce dont elle a besoin, tout en assurant qu'elles ne s'interfèrent pas mutuellement. Quant à la communication avec l'extérieur, l'hyperviseur peut fournir plusieurs techniques pour rendre accessible ou non les *VMs.* Il peut réaliser cette tâche par l'assignation d'adresses *IP (Internet Protocol)* · et numéros de port en utilisant différentes techniques d'accès réseau aux *VMs* (mode pont *(bridge)*, mode hôte *(host only)*, *NAT* (*Network Address Translation)*, reseau privée *VLAN* (*Virtual Local Area Network)* **[29]**.

1.2.4.2 Virtualisation applicative

La virtualisation applicative est un procédé permettant d'ajouter une couche d'abstraction entre le système d'exploitation et les applications. Cette technique permet d'exécuter des applications sur un poste client sans les installer sur ce poste. En effet, ces applications sont réellement installées sur un serveur virtuel distant. La virtualisation des applications permet d'exécuter des applications depuis un poste client indépendamment du système d'exploitation. Ce procédé permet d'éliminer le problème d'incompatibilité des applications **[29]**.

1.2.4.3 Virtualisation réseau

La virtualisation des réseaux consiste à créer des réseaux logiques qui s'appuient sur des ressources matérielles. Ce concept permet à plusieurs instances virtuelles (ou réseaux virtuels) de coexister et de partager les· ressources d'une ou de plusieurs infrastructures physiques de réseaux. En effet, plusieurs technologies existantes de réseau se basent sur ce concept. Parmi les techniques qui permettent de déployer des réseaux virtuels à travers une même infrastructure réseau physique nous retrouvons : les VLANs, les réseaux privés virtuels *VPN* (*Virtual Privat Network),* les réseaux de superposition ou de recouvrement (*overlay networks)* **[29]**.

La virtualisation de réseau peut s'avérer utile à héberger des machines virtuelles et des conteneurs, ou idéalement à un fournisseur de Cloud ou Fog Computing, en permettant à ceux-ci de mettre à disposition de leurs clients des environnements réseaux totalement isolés en se basant sur une infrastructure réseau commune **[29]**.

Les composantes de base de la virtualisation du réseau sont :

- Cartes réseaux virtuelles VNIC (Virtual Network Interface Card) : ces cartes sont des périphériques réseaux virtuels avec les mêmes interfaces de liaison de données qu'une carte d'interface réseau physique *(*Network Interface Gard - NIC*).* En outre, les ressources du système traitent les VNIC comme s'il s'agissait de cartes d'interface réseau physiques **[29]**.

- Commutateurs virtuels : il s'agit des logiciels intégrés à l'hyperviseur et qui se comportent comme des commutateurs matériels, ces logiciels permettent l'interconnexion des systèmes virtuels (Machines virtuelles, Conteneurs) entre eux ou avec des éléments réseaux physiques (cartes réseaux, commutateurs, etc.) **[29]**.

1.2.5 Centre de données

Un data center, ou centre de données, est un site comportant une ou plusieurs salles de grande surface hébergeant un grand nombre de serveurs informatiques, de baies de stockage, ainsi que des locaux techniques dédiés à l'alimentation électrique et à la climatisation **[27]**.

1.3 Internet des objets

"L'Internet des objets n'est pas seulement une technologie, c'est une nouvelle façon de penser." - Kevin Ashton, pionnier de l'IoT. Dans cette section, nous explorerons les principes fondamentaux de ce domaine en pleine expansion.

1.3.1 Définition

L'IoT est une infrastructure de réseau dynamique et globale dans laquelle les objets physiques (ex. accessoires, vêtements, machines, véhicules, etc.) sont autonomes et auto-configurables **[25].** Ces objets sont censés être capables d'envoyer et de recevoir des données sur Internet, de surveiller leur environnement et d'effectuer des actions basées sur le partage d'information, grâce à l'utilisation des technologies de l'information et de télécommunication **[24].**

1.3.2 Architecture en couches d'IoT

Bien que nous ne puissions pas couvrir toutes les possibilités et permutations, le groupe d'architectures suivant devrait vous permettre de mieux comprendre les considérations de conception de base et les couches fonctionnelles primaires typiques dans une pile IoT de bout en bout.

Couche de perception : le fonctionnement principal de l'IoT, c'est-à-dire la collecte d'informations se fait au niveau de la couche de perception à l'aide de différents appareils tels que la carte à puce, l'étiquette RFID, les réseaux de lecteurs et de capteurs, etc. Il a une fonction de détection complète à travers le Système RFID pour obtenir des informations sur les objets à tout moment et n'importe où. Chaque étiquette électronique RFID a un identifiant unique appelé code de produit électronique (EPC) qui est le seul identifiant de recherche attribué à chaque cible physique. Des informations supplémentaires sur le produit sont données par une suite de chiffres qui lui sont imposés tels que le fabricant et la catégorie de produit avec sa date de fabrication et sa date d'expiration, etc **[98]**.

Couche réseau : les données collectées par les capteurs étaient envoyées à Internet via la couche réseau avec l'aide d'ordinateurs, de réseau sans fil / filaire et d'autres composants. Par

conséquent, la couche réseau est principalement responsable de transmettre des informations avec la caractéristique de livraison fiable, par conséquent cette couche comprend également la fonctionnalité de la couche de transport **[98]**.

Couche application : analyse des informations reçues et prise de décisions de contrôle pour atteindre sa fonction de traitement intelligent par connexion, identification et contrôle entre objets et appareils. Les moyens de renseignement utilisent une technologie informatique intelligente telle que le Cloud Computing et traitent les informations pour un contrôle intelligent, comme ce qu'il faut faire et quand faire les choses. La figure 1.2 **[98]** montre l'architecture en trois couches d'IoT.

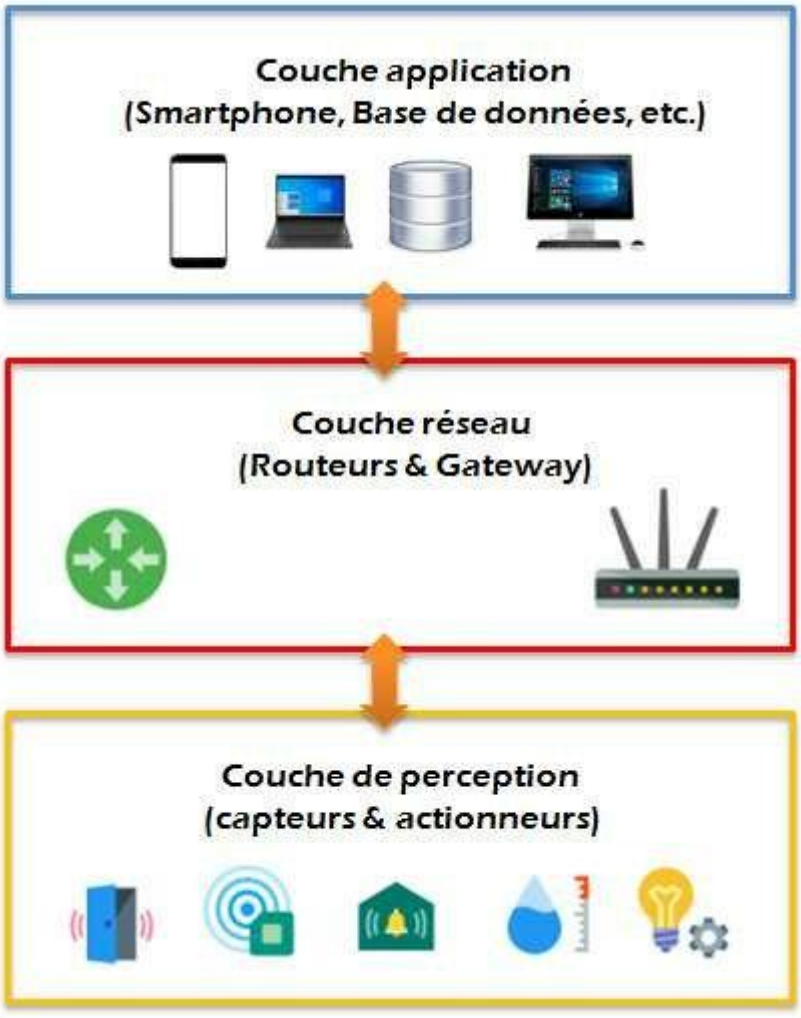


Figure 1.2 : Architecture en trois couches.

1.3.3 Domaines d'applications

Les domaines d'application de l'IoT sont classifiés par Atzori et al dans **[30]** comme suit :

• **Transport et logistique :** Les voitures, les bus, les taxis ainsi que les trains sont de plus en plus équipés de capteurs, d'actionneurs et de puissance de traitement. Les routes elles-mêmes et les marchandises transportées sont également équipées par des étiquettes et des capteurs qui envoient des informations aux sites de contrôle et aux véhicules de transport pour mieux acheminer le trafic, aider à la gestion des dépôts, et fournir les marchandises transportées. L'une des applications réussies de l'IoT dans le secteur des transports est Waze **[9]**, une application mobile de navigation GPS en temps réel. Cette application offre à ses utilisateurs des informations sur l'état du trafic en temps réel. Les utilisateurs, à leur tour, peuvent modifier la carte (sens de circulation, nouvelles rues, bouchons, etc.) afin de l'améliorer ou de suivre l'évolution du réseau routier. Actuellement, le nombre d'utilisateurs de Waze dépasse les 100 millions dans le monde **[10]**.

Une autre application de transport utilisant l'IoT est celle de Skydrone; une entreprise française qui offre à ses clients la possibilité de livrer leurs objets par drones autonomes **[8]**. Les drones disposent d'une capacité de poids de charge (variable selon le modèle du drone) et ils sont équipés d'un système de détection d'obstacles aériens qui leur permettant d'être autonomes. Cette solution de transport aérien permet des livraisons rapides et flexibles.

• **Télémédecine :** La télémédecine est une forme de pratique médicale à distance utilisant des technologies de l'information et de la communication. Elle permet d'établir un diagnostic ou d'assurer un suivi pour un patient à distance. Par exemple, dans un système de santé à distance, la concentration quotidienne en oxygène, la glycémie et la pression artérielle d'un patient sont collectées automatiquement par des capteurs et ensuite transmises aux organismes de suivi. Une autre application de télémédecine consiste à offrir des soins rapides pour les personnes ayant subi un accident ou un problème de santé. Pour cette dernière, Google propose une solution à base de drones pour porter des secours dans diverses situations d'urgence. Ces drones livrent un kit de premiers secours, des outils de diagnostic ou des médicaments **[7]**.

• **Environnements intelligents :** Les maisons, les villes et les usines sont des exemples majeurs d'environnements intelligents. Les applications de ce domaine consistent à collecter et

traiter des données de l'environnement pour réaliser des économies, sécuriser ou améliorer la vie des citoyens. Par exemple, dans le cas des maisons et des bureaux intelligents, le chauffage peut être adapté selon les conditions de la météo; l'éclairage de la pièce peut être changé en fonction de l'heure de la journée; les incidents peuvent être évités avec des systèmes de surveillance et d'alarme appropriés; et l'énergie peut être économisée en éteignant automatiquement les équipements électriques non nécessaires **[30]**. Dans le cas de l'industrie, des usines peuvent être enrichies par des capteurs et des actionneurs pour contrôler la chaîne de production, trier les produits défaillants ou arrêter le processus de production.

• **Relation sociale :** Les applications relevant de ce domaine sont celles qui permettent à l'utilisateur d'interagir avec d'autres personnes afin de maintenir et de développer des relations sociales. En effet, les objets peuvent automatiquement déclencher la transmission de messages à des amis pour leur permettre de savoir ce que la personne fait ou ce qu'elle a fait dans le passé. Quelques applications dans ce domaine sont : de partager des avis sur des produits, des services ou des endroits **[30]**.

• **Applications du futur :** Les applications décrites dans les sections précédentes sont déjà déployées. En dehors de celles-là, nous pouvons envisager de nombreuses autres applications futures. Parmi ces applications on peut noter : l'œil intelligent **[5]**. Cette technologie est très similaire aux lunettes de Google **[6]**. Elle consiste à équiper les lentilles de contact avec des capteurs et des capacités de connectivité comme le Wi-Fi et le Bluetooth pour fournir de nombreuses fonctions telles que parcourir des cartes, lire des emails ou des messages, naviguer sur Internet ou capturer des vidéos.

1.4 Les modèles de communication pour l'IoT

Nous décrirons trois modèles de communication différents pour l'IoT :

1.4.1 Communication machine à machine

Ce modèle de communication représente plusieurs appareils, qui peuvent se connecter et échanger des informations directement entre eux, sans aucune assistance matérielle intermédiaire. Ces appareils sont capables de se connecter les uns aux autres sur différents types de réseaux, notamment, mais sans s'y limiter, les réseaux Internet. Par exemple, la figure 1.3

[28] montre qu'un interrupteur intelligent communique avec la lumière intelligente via Bluetooth 4.0.

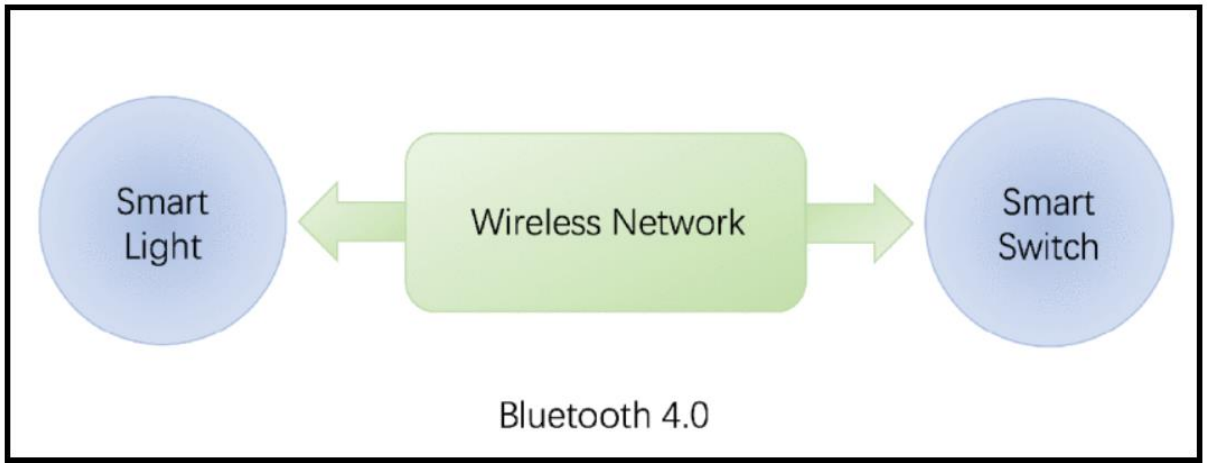


Figure 1.3 : Exemple d'une communication Machine-to-Machine

Ces réseaux de communication de dispositif à dispositif permettent aux appareils d'échanger des informations selon des protocoles de communication hybrides. Ces protocoles combinent un protocole appareil-à-appareil et un protocole spécifique pour répondre aux exigences de qualité de service (QoS). Ce modèle est couramment utilisé dans de nombreuses applications, telles que les systèmes de domotique ou le contrôle automatique des systèmes électriques, qui communiquent entre eux en envoyant de petits paquets de données et ont des besoins en débit de données relativement faibles. Les dispositifs IoT typiques de ce type sont, entre autres, les serrures de porte connectées, les interrupteurs intelligents et les lumières connectées, qui eux aussi n'échangent généralement que des petits paquets de données **[28]**.

Du point de vue de l'utilisateur, le problème de la communication machine à machine (M2M) est le manque de compatibilité, différents appareils de différents fabricants utilisant des protocoles différents. Prenons l'exemple des appareils domotiques : les appareils utilisant le protocole Z-Wave ne peuvent pas communiquer avec ceux utilisant le protocole ZigBee. Ces problèmes de compatibilité limitent le choix et l'expérience de l'utilisateur **[28]**.

1.4.2 Communication machine vers Cloud

Dans un modèle de communication machine vers Cloud, les appareils IoT font appel à un fournisseur de services applicatifs Cloud, ou stockent des données sur un disque de stockage Cloud, en raison des limitations des capacités de calcul ou d'espace de stockage des appareils.

Cette approche nécessite normalement l'assistance de stratégies de communication préexistantes, telles que les connexions filaires conventionnelles ou Wi-Fi.

Bien que la communication machine vers Cloud permette de résoudre les problèmes du modèle machine à machine, ce modèle dépend du réseau traditionnel, et la bande passante et les ressources réseau limitent les performances de ce modèle de communication. Pour améliorer les performances du modèle de communication machine vers Cloud, il est nécessaire d'optimiser la structure du réseau**,** la figure 1.4 **[28]** montre un exemple d'une communication machine vers cloud**.**

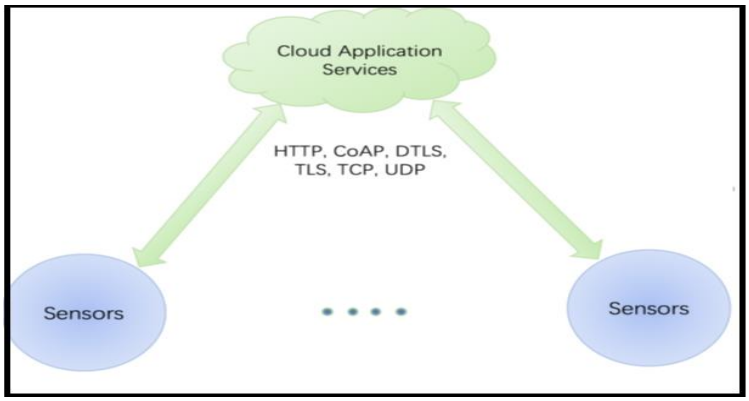


Figure 1.4 : Exemple d'une communication machine vers Cloud

1.4.3 Communication machine vers passerelle

Dans le modèle machine vers passerelle (voir la figure 1.5) **[28]**, le modèle de passerelle de couche application ALG (Application Layer Gateway) est considéré comme un intermédiaire. Au niveau de la couche application, des schémas de contrôle de sécurité basés sur des logiciels ou d'autres fonctionnalités telles que des algorithmes de traduction de données ou de protocoles s'exécutent sur une passerelle ou un autre périphérique réseau, qui agit comme un pont intermédiaire entre les appareils IoT et les services applicatifs Cloud. Cela améliore la

sécurité et la flexibilité du réseau IoT, déplace une partie de la tâche de calcul vers la couche application et réduit considérablement la consommation d'énergie des appareils IoT. Par exemple, le téléphone mobile intelligent peut servir de passerelle, exécutant certaines applications pour communiquer avec les appareils IoT et le Cloud **[28]**.

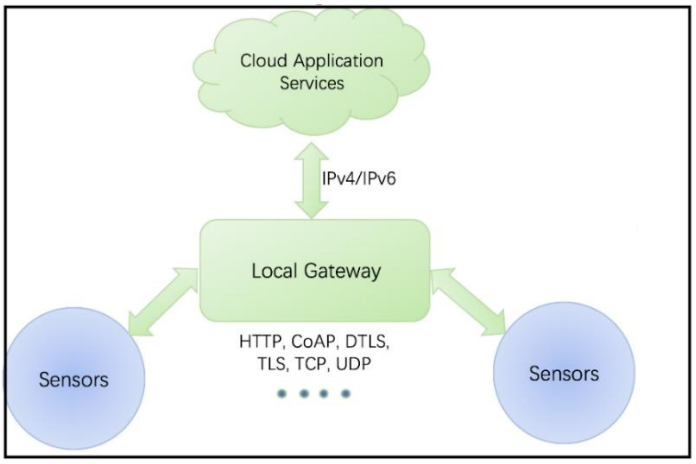


Figure 1.5 : Exemple d'une communication machine vers passerelle.

1.5 Cloud Computing

Dans cette section, nous dévoilerons les principes fondamentaux de Cloud Computing

1.5.1 Définition

Le Cloud Computing est un modèle permettant un accès réseau omniprésent, pratique et à la demande à un pool partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement provisionnées et libérées avec un effort de gestion minimal ou une interaction avec le fournisseur de services. Ce modèle Cloud est composé de cinq caractéristiques essentielles, de trois modèles de services et de quatre modèles de déploiement **[11]**. La figure 1.6 **[16]** montre un réseau IoT basé sur le cloud.

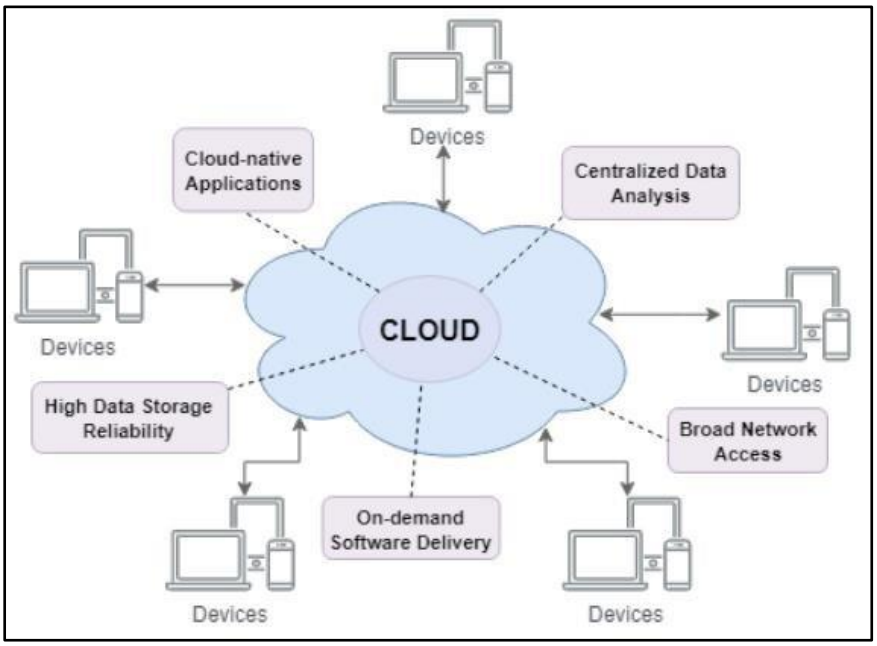


Figure 1.6 : Réseau IoT basé sur le Cloud

1.5.2 Caractéristiques de Cloud Computing

Les caractéristiques fondamentales de Cloud Computing **[11]** sont :

1.5.2.1 Le libre-service à la demande

Le libre-service à la demande (On-demand self-service) permet à un consommateur d'approvisionner de façon autonome des ressources informatiques, telles que l'espace de stockage sur un serveur et la bande passante du réseau, en fonction de ses besoins et de manière automatique, sans nécessiter d'interaction humaine avec chaque fournisseur de services.

1.5.2.2 Un large accès réseau

Les capacités sont disponibles sur le réseau et accessibles via des mécanismes standards qui favorisent l'utilisation par des plateformes clientes hétérogènes légères ou lourdes (par exemple, téléphones portables, tablettes, ordinateurs portables et stations de travail).

1.5.2.3 Le pooling de ressources

Le pooling de ressources consiste à regrouper les ressources informatiques du fournisseur afin de servir plusieurs clients en utilisant un modèle multi-tenant. Différents types de

ressources physiques et virtuelles sont attribués et réattribués de manière dynamique en fonction de la demande des clients.

Ce modèle offre une indépendance vis-à-vis de l'emplacement physique des ressources. En effet, le client n'a généralement aucun contrôle ni connaissance de la localisation exacte des ressources fournies, mais peut parfois spécifier un emplacement à un niveau d'abstraction plus élevé (par exemple, pays, région ou centre de données).

1.5.2.4 L'élasticité rapide

L'élasticité rapide permet de provisionner et de libérer des capacités de manière élastique, parfois automatiquement, pour s'étendre rapidement ou se réduire en fonction de la demande. Pour le consommateur, les capacités disponibles pour le provisionnement semblent souvent illimitées et peuvent être utilisées en toute quantité à tout moment.

1.5.2.5 Le service mesuré

Les systèmes Cloud contrôlent et optimisent automatiquement l'utilisation des ressources en s'appuyant sur une capacité de comptage (metering) à un niveau d'abstraction adapté au type de service (par exemple, stockage, traitement, bande passante et comptes d'utilisateurs actifs). L'utilisation des ressources peut être surveillée, contrôlée et rapportée, ce qui assure la transparence pour le fournisseur et le consommateur du service utilisé.

1.5.3 Modèles de service

Parmi les modèles de service de Cloud Computing, on peut citer **[11]** :

1.5.3.1 Logiciel en tant que service (SaaS)

La capacité fournie au consommateur est d'utiliser les applications du fournisseur fonctionnant sur une infrastructure Cloud. Ces applications sont accessibles depuis divers terminaux clients via une interface légère, comme un navigateur Web (par exemple, messagerie électronique en ligne), ou une interface de programme.

Le consommateur ne gère ni le contrôle l'infrastructure Cloud sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation, le stockage, ou même les capacités

individuelles des applications, sauf peut-être pour des paramètres de configuration d'application spécifiques à chaque utilisateur.

1.5.3.2 Plateforme en tant que service (PaaS).

La capacité fournie au consommateur est de déployer sur l'infrastructure Cloud des applications créées ou acquises par le consommateur, en utilisant des langages de programmation, des bibliothèques, des services et des outils pris en charge par le fournisseur. Le consommateur ne gère ni ne contrôle l'infrastructure Cloud sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation ou le stockage, mais il a le contrôle sur les applications déployées et éventuellement sur les paramètres de configuration de l'environnement d'hébergement des applications.

1.5.3.3 Infrastructure en tant que service (IaaS)

La capacité fournie au consommateur est de provisionner du traitement, du stockage, des réseaux et d'autres ressources informatiques fondamentales où il peut déployer et exécuter des logiciels arbitraires, pouvant inclure des systèmes d'exploitation et des applications. Le consommateur ne gère ni le contrôle l'infrastructure Cloud sous-jacente, mais il a le contrôle sur les systèmes d'exploitation, le stockage et les applications déployées ; il peut également avoir un contrôle limité sur certains éléments du réseau comme les pares-feux d'hôtes **[11]**.

1.5.4 Modèles de déploiement

Dans la littérature, il y a quatre modèles de déploiement : Cloud privé, Cloud communautaire, Cloud public et Cloud hybride **[11]**

1.5.4.1 Cloud privé

L'infrastructure Cloud est provisionnée pour l'usage exclusif d'une seule organisation regroupant plusieurs consommateurs comme des unités commerciales. Elle peut être détenue, gérée et exploitée par l'organisation, par un tiers, ou par une combinaison des deux, et elle peut exister sur site ou en dehors des locaux de l'organisation.

1.5.4.2 Cloud communautaire

L'infrastructure Cloud est provisionnée pour l'usage exclusif d'une communauté spécifique d'utilisateurs provenant d'organisations partageant des préoccupations communes (par exemple, la mission, les exigences de sécurité, les politiques et la conformité). Elle peut être détenue, gérée et exploitée par une ou plusieurs organisations de la communauté, par un tiers, ou par une combinaison des deux, et elle peut exister sur site ou en dehors des locaux de l'organisation.

1.5.4.3 Cloud public

L'infrastructure Cloud est provisionnée pour un usage ouvert par le grand public. Elle peut être détenue, gérée et exploitée par une entreprise, un organisme universitaire ou gouvernemental, ou par une combinaison de ceux-ci. Elle existe dans les locaux du fournisseur de Cloud.

1.5.4.4 Cloud hybride

L'infrastructure Cloud est une composition de deux ou plusieurs infrastructures Cloud distinctes (privée, communautaire ou publique) qui restent des entités uniques, mais qui sont reliées entre elles par une technologie standardisée ou propriétaire permettant la portabilité des données et des applications.

1.6 Obstacles rencontrés par le Cloud Computing

D'après les recherches de **[22]** les obstacles rencontrés par le Cloud Computing sont :

- Disponibilité et continuité de service;
- Données propriétaires (Lock-in);
- Confidentialité de données et audit;
- Transfert des données et congestion du réseau;
- Performance imprévisible;
- Beugues des systèmes distribués à large échelle.

1.7 Du Cloud au Fog

Pour répondre au gaspillage de ressources créé par le Cloud, une nouvelle technologie appelée le « Fog » ou le « Edge » est en train d'être développée par des chercheurs industriels et académiques. Le Fog n'a pas vocation à remplacer le Cloud, mais plutôt à l'étendre pour le rendre plus efficace et moins gourmand en énergie. L'idée est de rajouter des machines supplémentaires, situées aussi près que possible des utilisateurs, de façon à raccourcir les distances réseau. Pour reprendre l'exemple de la vidéoconférence avec des collègues dans la même ville que moi, l'utilisation de serveurs « Fog » à proximité nous permettrait de considérablement réduire la consommation de ressources inutiles. On ne gagne cependant pas à tous les coups : si je veux communiquer avec un ami aux États-Unis, alors on ne pourra pas éviter d'envoyer beaucoup de données à travers l'océan Atlantique d'une façon ou d'une autre. Mais on sait que la majorité des interactions sur Internet réunissent des utilisateurs peu distants les uns des autres. Le Fog prend également tout son sens dans le contexte du développement extrêmement rapide de l'Internet des Objets, c'est-à-dire l'ensemble de tous les objets communicants présents dans notre vie quotidienne, en entreprise, chez les pouvoirs publics, etc. Par exemple, un agriculteur peut vouloir installer des capteurs dans ses champs pour mesurer les conditions météo, l'humidité des sols à tel ou tel emplacement, la croissance des végétaux, et ainsi de suite. Le traitement automatisé des données collectées peut par exemple permettre de contrôler à granularité très fine la quantité d'eau qui doit irriguer telle ou telle partie du champ. Même si chaque capteur produit peu de données, l'ensemble des objets connectés dans le monde produit un trafic réseau total qui croît beaucoup plus rapidement que la capacité de l'Internet. Puis viendra le temps du transfert industriel et du déploiement à grande échelle. De très nombreux groupes industriels majeurs se positionnent sur le sujet et lancent des investissements colossaux pour être prêts le moment venu à proposer des services « Fog » commerciaux fiables et efficaces à leurs clients : Orange, Google, Microsoft, Hewlett Packard, Ericsson, Nokia, etc. Comme souvent dans ce genre de domaine, les premiers clients seront sans doute des entreprises de haute technologie. Puis la technologie se démocratisera peu à peu et deviendra accessible à des pouvoirs publics, des petites entreprises, et enfin au grand public **[18]**. La figure 1.7 **[23]** illustre l'évolution de l'architecture Cloud vers le Fog.

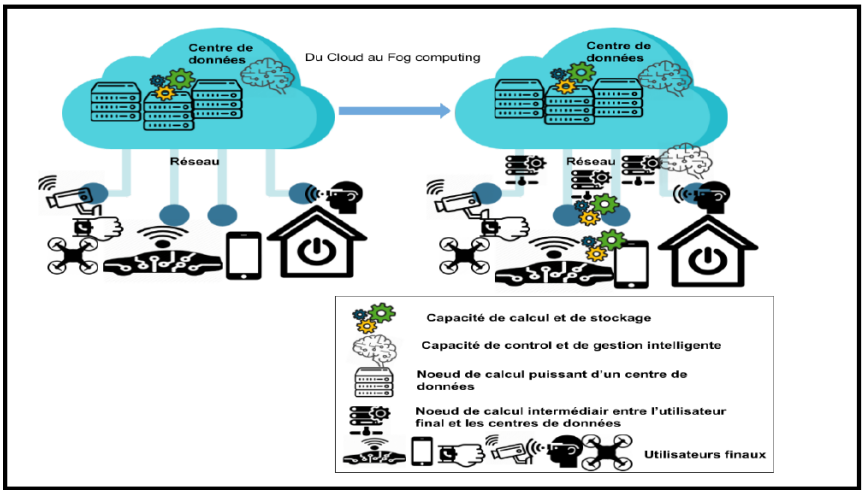


Figure 1.7 : Illustration de l'évolution de l'architecture Cloud vers le Fog

1.8 Fog Computing

Le Fog Computing est un paradigme de ressources horizontales, physiques ou virtuelles proposé par Cisco en 2012 qui réside entre les appareils finaux et Cloud ou centres de données traditionnels. Ce paradigme prend en charge les applications verticalement isolées et sensibles à la latence en fournissant une connectivité informatique, de stockage et de réseau omniprésente, évolutive, en couches, fédérée et distribuée **[12, 13,14]**.

Le Cloud et le Fog se complètent pour former un continuum de services interdépendants et mutuellement bénéfiques entre les points de terminaison et le Cloud pour rendre le stockage, le contrôle, le calcul et la communication disponibles n'importe où le long du continuum. Cependant, on peut conclure que le Fog Computing présente de multiples avantages par rapport au Cloud Computing. De plus, il est excellent pour résoudre les problèmes de performance et de localité car ses services et ressources spécifiques sont virtualisés là où il se trouve au bord du réseau **[15]**. La figure 1.8 **[16]** montre un réseau IoT basé sur le Fog.

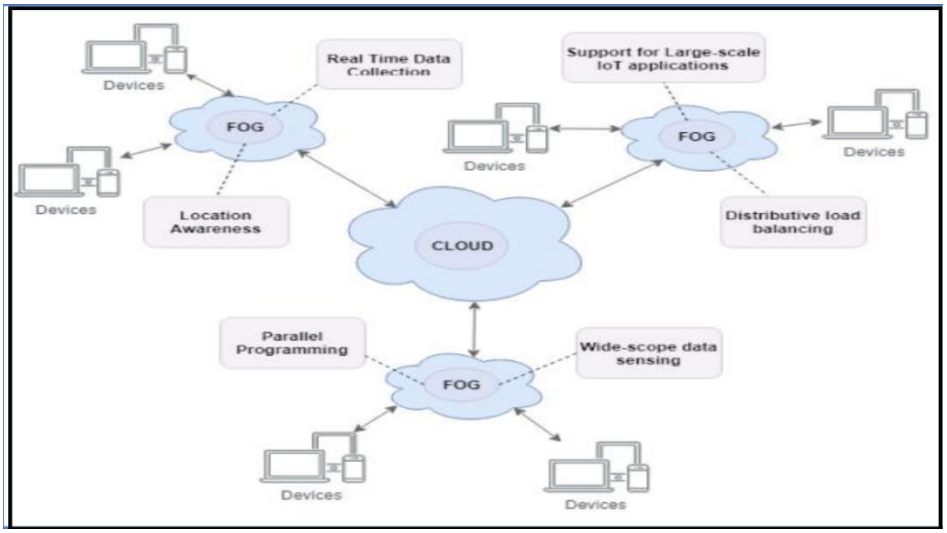


Figure 1.8 : Réseau IoT basé sur le Fog

1.8.1 Caractéristiques du Fog Computing

Parmi les caractéristiques du Fog Computing on peut citer **[17]**

1.8.1.1 Conscience de l'emplacement contextuel et faible latence

Les origines du Fog peuvent être retracées aux premières propositions soutenant les points d'extrémité avec des services riches en périphérie du réseau, y compris des applications nécessitant une faible latence (par exemple, jeux, diffusion vidéo et réalité augmentée). Étant donné que les nœuds Fog sont généralement très proches des points d'extrémité de l'IoT, l'analyse et la réponse aux données générées par ces points d'extrémité sont beaucoup plus rapides que celles d'un service Cloud centralisé.

1.8.1.2 Distribution géographique

En contraste marqué avec le Cloud plus centralisé, les services et applications ciblés par le Fog exigent des déploiements largement distribués. Par exemple, le Fog jouera un rôle actif dans la fourniture de services de streaming de haute qualité aux véhicules en mouvement, à travers des mandataires et des points d'accès positionnés le long des autoroutes et des voies ferrées.

1.8.1.3 Réseaux de capteurs à grande échelle

Pour surveiller l'environnement et le Smart Grid (réseau électrique intelligent) sont d'autres exemples de systèmes intrinsèquement distribués, nécessitant des ressources informatiques et de stockage distribué.

1.8.1.4 Très grand nombre de nœuds

En conséquence de la large géo-distribution, comme en témoignent les réseaux de capteurs en général, et le Smart Grid en particulier.

1.8.1.5 Support pour la mobilité

Il est essentiel pour de nombreuses applications Fog de communiquer directement avec des appareils mobiles, et donc de prendre en charge des techniques de mobilité telles que le protocole LISP (Locator/ID Separation Protocol) qui découplent l'identité de l'hôte de l'identité de localisation, et nécessitent un système d'annuaire distribué.

1.8.1.6 Interactions en temps réel

Les applications Fog importantes impliquent des interactions en temps réel plutôt que du traitement par lots.

1.8.1.7 Prédominance de l'accès sans fil

Bien que le Fog Computing soit utilisé dans des environnements câblés, l'échelle importante des capteurs sans fil dans l'IoT exige une analyse et un calcul distribués. Pour cette raison, le Fog convient très bien aux réseaux d'accès IoT sans fil.

1.8.1.8 Hétérogénéité

Les nœuds Fog se présentent sous différentes formes et seront déployés dans une grande variété d'environnements, tout comme les appareils à partir desquels ils collectent les données peuvent varier en termes de forme et de capacité de communication réseau.

1.8.1.9 Interopérabilité et fédération

Le support transparent de certains services (les services de streaming en temps réel en sont un bon exemple) nécessite la coopération entre différents fournisseurs. Par conséquent, les composants Fog doivent être capables d'interopérer, et les services doivent être fédérés entre domaines.

1.8.1.10 Support pour l'analyse en temps réel et l'interaction avec le Cloud

Le Fog est positionné pour jouer un rôle significatif dans l'ingestion et le traitement des données à proximité de la source au moment où elles sont produites. Alors que les nœuds Fog fournissent une localisation permettant une faible latence et une conscience contextuelle, le Cloud assure une centralisation globale de nombreuses applications nécessitent à la fois la localisation par le Fog et la globalisation par le Cloud, notamment pour les analyses et le Big Data. Le Fog est particulièrement adapté aux analyses en continu en temps réel par opposition aux analyses historiques sur Big Data généralement effectuées dans un centre de données.

1.8.2 Nœuds Fog

Les nœuds Fog (Fog Nodes FN) sont des éléments de calcul intermédiaires du réseau d'accès des appareils intelligents situés entre le Cloud et les appareils intelligents. Les nœuds Fog peuvent être des éléments physiques ou virtuels et sont étroitement liés aux appareils intelligents ou aux réseaux d'accès. Les nœuds Fog fournissent généralement une forme de gestion des données et de service de communication entre la couche périphérique où résident les appareils intelligents et le Cloud. Les nœuds Fog, en particulier les éléments virtuels, également appelés cloudlets, peuvent être fédérés pour fournir une expansion horizontale de la fonctionnalité sur des géolocalisations dispersées **[17]**.

1.8.3 Type d'équipements Fog

Les équipements de calcul considérés dans le Fog, qu'ils soient physiques ou virtuels, sont très hétérogènes aussi bien au niveau de leur profil de consommation énergétique que des capacités de calcul offertes et des technologies de communication qu'ils utilisent. Les nœuds peuvent à la fois posséder des spécificités de nœuds réseau c'est-à-dire faire communiquer d'autres équipements tout en ayant la capacité d'héberger des services et de faire du calcul. On

trouve également des points d'accès qui sont uniquement chargés de l'acheminement de l'information. Voici dans ce qui suit une liste non exhaustive des équipements qu'on trouve actuellement dans des infrastructures Fog **[23]** :

- Nœud physique : Passerelle, Raspberry, Micro-contrôleur, Noeud point d'accès, ordinateur personnel, autres objets ayant une capacité de calcul.
- Nœud virtuel : Machine virtuel, Container.
- Micro ou Nano centres de données regroupant des nœuds physiques et virtuels ayant des capacités de calculs réduites.

1.8.4 L'architecteur de Fog Computing

Le Fog présente une infrastructure virtuelle et géo-distribuée intégrant un grand nombre d'équipements hétérogènes et qui sont interconnectés. Ci-après, nous décrivons chaque niveau de l'infrastructure du Fog Computing illustrée dans la Figure 1.9 **[19]**.

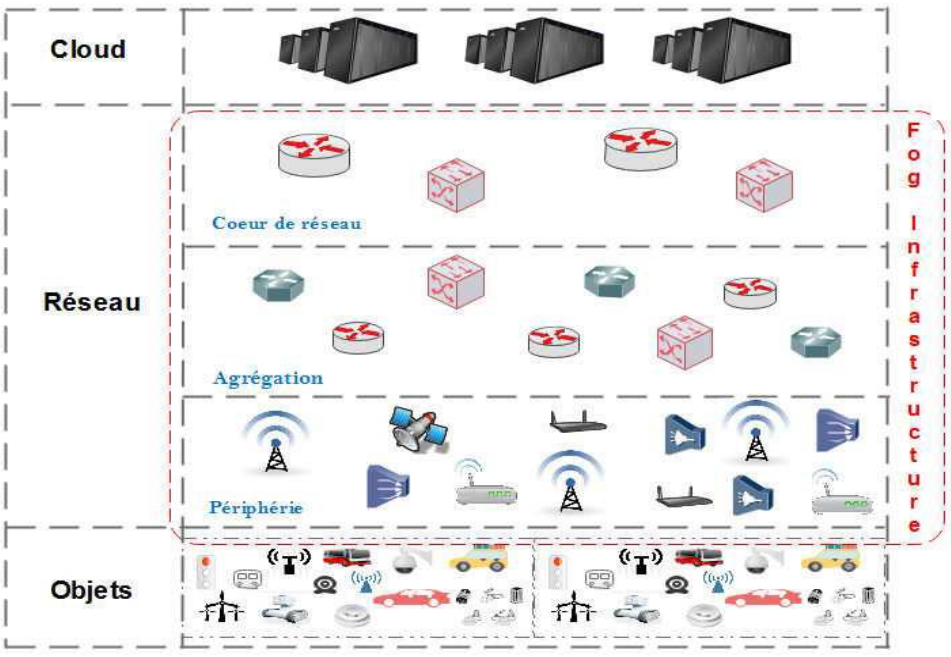


Figure 1.9 : Infrastructure du Fog Computing

- **Les objets :** c'est le niveau le plus proche des utilisateurs finaux et de leur environnement physique. Il inclut des objets sans fils, intelligents, mobiles ou fixes comme les capteurs, les robots, les smartphones, et les caméras connectées. Ces équipements sont géo-distribués à grande échelle. Ils sont chargés de surveiller l'environnement et transmettent des informations aux niveaux supérieurs pour l'analyse et le stockage par le biais des technologies de communication comme le Wi-Fi, Bluetooth ou le réseau cellulaire **[20,19]**.
- **Le réseau :** ce niveau couvre plusieurs couches du réseau (les périphéries, l'agrégation et le cœur de réseau). Il regroupe des dispositifs de réseau comme les switchs, les routeurs et les POP (Point of Presence). Ce niveau est utilisé pour héberger des applications nécessitant un temps de réponse court, ainsi que pour faire l'agrégation, le filtrage et le prétraitement des données avant de les envoyer au Cloud. D'autres fonctionnalités peuvent être déployées dans ce niveau comme celles du CDN (Content Delivery Network) pour mettre en caches des données, et du SDN (Software Defined Networking) et du NFV (Network Functions Virtualization) pour administrer le réseau **[20, 21,19]**.
- **Le Cloud :** c'est le niveau le plus haut dans l'infrastructure. Il s'agit de serveurs puissants distribués dans des centres de données distants permettant d'héberger des applications destinées, par exemple, à l'analyse du Big Data et au stockage de données permanent **[19]**.

1.8.5 Modèles de service et de déploiement dans le Fog Computing

Bien que le Fog Computing et le Cloud Computing présentent des différences architecturales fondamentales, il existe des similitudes importantes dans leurs modèles de services et de déploiement (mentionné précédemment dans la section 1.5.3 et 1.5.4).

1.8.6 Les Avantages du Fog Computing pour supporter les applications IoT

Le Fog Computing a été pensé comme nouveau paradigme de calcul pour pallier les limites du Cloud Computing et répondre aux besoins des applications sensibles au délai et celles qui utilisent intensivement les ressources réseau. Les infrastructures Fog Computing offrent les avantages suivants **[23]** :

1. Réduction du trafic réseau et de la consommation énergétique des centres de données.

2. Amélioration de la sécurité en rapprochant le traitement et le stockage des données utilisateurs et en utilisant des méthodes de sécurité plus complexes qui ne peuvent pas être directement intégrées dans les équipements IoT.

3. Supporter le très grand nombre d'équipements IoT et leur distribution sur de larges zones géographiques par une structure hiérarchique, ce qui peut minimiser la sous-utilisation des ressources.

4. Améliorer la durée de vie des objets connectés en transférant leur charge vers des noeuds voisins (ou d'autres nœuds Fog ou Cloud).

5. Supporter des applications exigeantes en temps de réponse.

6. Une application IoT est liée à ses capteurs et ses actionneurs et ces derniers peuvent être éparpillés sur une large zone. La conscience de la localité du Fog est une caractéristique importante qui peut aider à améliorer la disponibilité et à la gestion efficace des échanges.

1.8.7 Les défis du Fog Computing pour supporter les applications IoT

Le Fog semble être l'infrastructure idéale pour supporter les futures applications IoT. Cependant, il reste encore un certain nombre de verrous à lever pour qu'il puisse être efficacement utilisé. Voici une liste non exhaustive des défis apportés par l'utilisation du Fog en tant que paradigme de calcul **[23]**.

1. Gérer efficacement les interactions entre les nœuds à la périphérie et les centres de données (offloading).

2. La découverte et l'estimation des ressources.

3. La problématique du placement et du déploiement des entités de gestion et de contrôle d'un environnement Fog.

4. La gestion du compromis entre efficacité d'utilisation des ressources de calculs et celle des ressources de communication.

5. Le problème du stockage distribué des données, leur persistance et leur intégrité.

6. L'absence d'un modèle économique pour les services offerts par les infrastructures Fog Computing.

1.8.8 Comparaison entre le Fog Computing et le Cloud Computing

Tableau 1.1 **[29]** illustre une comparaison entre le Cloud Computing et Fog Computing

Eléments	Cloud Computing	Fog Computing
Latence	Élevée	Faible
Matériel	Stockage et puissance de calcul évolutifs	Stockage et puissance de calcul limités
Emplacement des serveurs	Au sein d'Internet	En périphérie du réseau local
Distance entre client et serveur	Plusieurs sauts	Un saut
Environnement de travail	Bâtiment de la taille d'un entrepôt avec systèmes de climatisation	Extérieur (par exemple, rues, jardins) ou intérieur (par exemple, restaurants)
Mesures de sécurité	Définies	Difficile à définir
Attaque sur les données	Moins probable	Forte probabilité
Déploiement	Centralisé	Distribué
Détection de position	Non	Oui

Tableau 1.1 : Comparaison entre le Fog Computing et le Cloud Computing

1.8.9 Les domaines d'utilisation de Fog Computing

Il existe de nombreux domaines importants où le Fog Computing peut jouer un rôle essentiel dans différentes applications IoT. Cette section donne un aperçu de diverses applications IoT qui peuvent bénéficier du Fog Computing **[29]**.

Voitures connectées

Selon Cisco, les véhicules autonomes représentent une nouvelle tendance pour les voitures. De nombreuses fonctionnalités avantageuses peuvent être ajoutées aux voitures, en s'appuyant sur le Fog Computing et la connexion internet, telles que la direction automatique,

la conduite « mains libres » ou le stationnement automatique, qui supprime le besoin d'un conducteur derrière le volant pour garer le véhicule .En utilisant le Fog Computing à la place du Cloud, les collisions et autres accidents peuvent être réduits car il ne souffre pas de la latence inhérente à l'approche centralisée du Cloud, permettant ainsi de sauver des vies, littéralement **[29]**.

Feux de circulation intelligents

Le Fog Computing permet aux feux de circulation d'ouvrir les routes en fonction de la détection de feux clignotants. Il détecte la présence de piétons et de cyclistes et mesure la distance et la vitesse des véhicules à proximité. L'éclairage des capteurs s'allume lorsqu'il identifie des mouvements et vice-versa. Les feux de circulation intelligents peuvent être considérés comme des nœuds Fog synchronisés entre eux pour envoyer des messages d'avertissement aux véhicules à proximité. Les interactions entre le Fog et les véhicules, ainsi que les points d'accès, sont améliorés grâce au WiFi, à la 3G, aux feux de circulation intelligents et aux unités en bord de route **[29]**.

Les maisons connectées (Domotique)

L'IoT connecte de nombreux capteurs et appareils au sein d'une maison. Cependant, ces appareils proviennent de différents fabricants et fonctionnent sur des plateformes distinctes, ce qui complique leur interaction. De plus, certaines tâches nécessitent une grande quantité de calcul et de stockage. Le Fog Computing permet de résoudre bon nombre de ces problèmes. Il intègre toutes les plateformes différentes et dote les applications de domotique de ressources flexibles. Le Fog Computing présente de nombreux avantages pour les applications de sécurité domestique. Il fournit une interface unifiée pour intégrer tous les appareils indépendants. De plus, il offre des ressources élastiques permettant le stockage, le traitement en temps réel et une faible latence **[29]**.

Réseaux de capteurs et d'actionneurs sans fil (WSN)

L'une des principales caractéristiques des WSN est leur capacité à augmenter la durée de vie de la batterie en fonctionnant principalement à faible consommation d'énergie. Les actionneurs sont utilisés pour gérer l'opération de mesure et modifier le comportement, formant

ainsi un système en boucle fermée. Les actionneurs peuvent être considérés comme des nœuds Fog qui fournissent différentes actions pour contrôler les appareils finaux équipés de capteurs. Ces WSN (Wireless Sensor and Actuator Networks) nécessitent moins de bande passante, moins d'énergie et une très faible puissance de traitement **[29]**.

Surveillance de la santé et suivi d'activité

Le Fog Computing joue un rôle essentiel dans les soins de santé. Il permet un traitement en temps réel et des réponses aux événements, ce qui est crucial dans ce domaine. De plus, l'interaction d'un grand nombre d'appareils médicaux pour le stockage à distance, le traitement et la récupération des dossiers médicaux à partir du Cloud nécessite une connexion réseau fiable, ce qui n'est pas toujours disponible. Le Fog Computing permet également de résoudre les problèmes de connectivité réseau et de trafic **[29]**.

IoT et systèmes cyber-physiques

L'intégration du Fog Computing avec l'IoT et les systèmes cyber-physiques devient possible. L'IoT est un réseau capable d'interconnecter tous les appareils du monde entier grâce à une adresse identifiée via internet et les télécommunications **[29]**.

D'un autre côté, le CPS (Cyber–Physical Systems) est une combinaison d'éléments physiques et informatiques. L'intégration des CPS avec l'IoT transformera le monde en une réalité physique basée sur l'ordinateur. Le Fog Computing est conçu avec la notion de systèmes embarqués, par exemple les véhicules connectés, les appareils médicaux et autres. En combinant le Fog Computing avec l'IoT et les CPS, il deviendra possible de développer des dispositifs médicaux intelligents, des bâtiments intelligents, ainsi que des systèmes agricoles et robotiques **[29]**.

Réalité augmentée

La réalité augmentée (Augmented Reality AR) désigne des systèmes qui ajoutent des informations virtuelles au monde réel. L'AR est devenu une application importante grâce à la miniaturisation, l'accélération et la généralisation des appareils informatiques. Les applications de AR sont extrêmement sensibles à la latence, car un léger retard dans la réponse de l'application peut nuire à l'expérience utilisateur. Le Fog Computing a le potentiel de jouer un

rôle clé dans le domaine de la AR en exploitant à la fois les serveurs Fog et Cloud pour permettre des applications en temps réel **[29]**.

1.9 Problème d'optimisation

Un problème d'optimisation consiste à minimiser, ou maximiser, une fonction objectif ***f*** dans le but de trouver la (ou les) solution(s) optimale(s) x^* parmi un ensemble de solutions noté *S*, appelé espace de recherche ou espace de solutions **[33]**.

Un problème de minimisation est défini comme suit :

$$f(x^*) \leq f(x), \forall x \in S \, , soit \quad \min_{x \in S} f(x) \tag{1.1}$$

Dans ce cas la fonction objectif ***f*** est connue comme une fonction de coût. De même, pour un problème de maximisation **[33]** :

$$f(x^*) \geq f(x), \forall x \in S \, , soit \quad \max_{x \in S} f(x) \tag{1.2}$$

Dans ce cas la fonction objectif *f* est connue comme une fonction d'utilité, de profit, ou de fitness.

Résoudre un problème d'optimisation, consiste alors à trouver la solution optimale (optimum global), qui est la meilleure solution possible de la fonction objectif *f.* Cependant, il peut exister des solutions intermédiaires, qui sont également des optimums, mais uniquement pour un sous-espace restreint de l'espace de recherche, ces solution appelées optimums locaux **[33]**.

Les problèmes d'optimisation se divisent en deux classes : les problèmes avec des variables continues, et les problèmes avec des variables discrètes ou les problèmes combinatoires **[32]**. La solution pour un problème continu est souvent représentée par un ensemble de nombres réels, tandis que la solution pour un problème combinatoire est souvent représentée par ensemble finie de nombres entiers **[33]**.

1.9.1 Problème d'optimisation discret (combinatoire)

Un problème d'optimisation combinatoire COP (Constraint Optimization Problem) est une discipline utilisant conjointement différentes techniques des mathématiques discrètes, de la recherche opérationnelle et de l'informatique, afin de résoudre des problèmes d'optimisation dont la structure sous-jacente est discrète (généralement un graphe).

L'optimisation combinatoire consiste à chercher une solution optimale dans un ensemble discret et fini, en théorie c'est facile car il suffit d'essayer toutes les solutions (combinaisons), et de comparer leurs qualités pour voir la meilleure. Cependant, en pratique, l'énumération de toutes les solutions peut prendre trop de temps ; or, le temps de recherche de la solution optimale est un facteur très important et c'est à cause de lui que les problèmes d'optimisation combinatoire sont réputés si difficiles **[33]**.

1.9.2 La complexité d'un COP

La complexité d'un problème est la quantité de ressources (en nombre d'instructions et en espace mémoire) dont a besoin un algorithme exact pour trouver une solution optimale

. Un problème est de complexité polynomiale (resp. Exponentiel) s'il existe un algorithme permettant de trouver une solution optimale pour toutes ses instances en un temps polynomial (resp. Exponentielle) par rapport à la taille de l'instance **[19]**.

Un algorithme dont le temps d'exécution est de l'ordre de n^K, où n est la taille de l'instance, serait considéré comme ayant une complexité polynomiale.

Un exemple de problème polynomial est celui qui consiste à trouver tous les plus courts chemins entre paires de sommets dans un graphe. L'algorithme exact trouvé par Floyd-Warshall **[34]** permet de le résoudre en temps polynomial $O(n^3)$, avec n le nombre de nœuds du graphe **[19]**.

Ci-après, nous citons les différentes classes de complexité pour les COP **[19]**.

1. **La classe** P **:** un COP ramené à un problème de décision est de classe P s'il peut être résolu, de manière exacte, par un algorithme de complexité polynomiale. On peut citer par exemple le problème de calcul de plus court chemin dans un graphe.
2. **La classe** NP **:** un COP ramené à un problème de décision est dit de classe NP si et seulement si, on peut vérifier que n'importe quelle proposition est bien une solution du problème, en un temps polynomial.
3. **La classe** NP − complet **:** un COP ramené à un problème de décision est dit de classe NP − complet s'il vérifie les deux propriétés suivantes :

- Il est possible de vérifier une solution en temps polynomial,
- Tous les problèmes de la classe NP se ramènent à celui-ci via une réduction polynomiale**.**

4. **La classe** NP −difficile **:** un problème NP −difficile est un problème qui remplit simplement la seconde condition, et donc peut être dans une classe de problème plus large et plus difficile que la classe NP.

1.10 Méthodes de résolution des problèmes d'optimisation

Un grand nombre de méthodes de résolution existe pour résoudre les problèmes d'optimisation combinatoire, Ces méthodes sont classées en deux catégories : les méthodes exactes qui garantissent l'optimalité de la solution trouvée, et les méthodes approchées qui relâchent le critère d'optimalité de la solution pour gagner en temps de résolution **[19,33]**.

1.10.1 Méthodes exactes

Ce sont des méthodes efficaces qui garantissent l'obtention de la solution optimale (optimum global) du problème traité de petite taille. Elles consistent à effectuer une énumération explicite de toutes les solutions possibles pour assurer l'obtention de toutes les meilleures solutions optimales potentielles que la solution optimale trouvée au cours de la recherche. Parmi Les méthodes exactes, on peut citer : la programmation linéaire en nombres entiers, les procédures de recherche arborescente et la programmation dynamique, elles offrent en théorie la garantie de trouver une solution optimale mais sont coûteux en temps de calcul et d'espace mémoire, la figure 1.10 **[33]** montre une taxonomie des méthodes de résolution d'un problème d'optimisation.

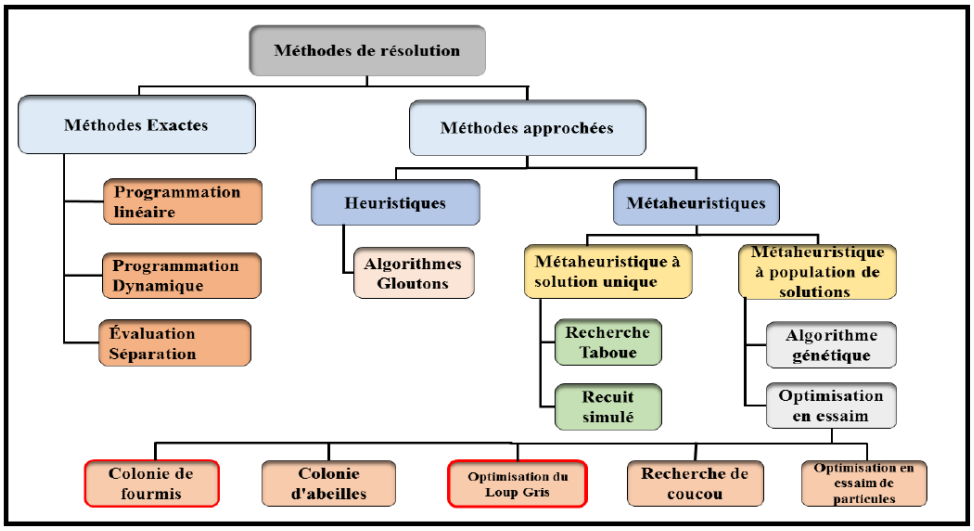


Figure 1.10 : Taxonomie des méthodes de résolution d'un problème d'optimisation

1.10.2 Méthodes approchées

L'utilisation de méthodes exactes n'est pas toujours possible pour un problème donné à cause d'un certain nombre de contraintes, telles que le temps de calcul souvent important ou bien la difficulté, voire l'impossibilité dans certains cas, d'une définition séparée du problème.

Pour faire face à ces contraintes, nous utilisons des méthodes approchées qui sont souvent utilisées pour accélérer le processus et trouver une bonne solution rapidement dans les cas où une recherche exhaustive est peu pratique. Les méthodes approchées offrent une solution de bonnes qualités en un temps raisonnable. Ces méthodes peuvent être réparties en deux classes : Heuristiques et Métaheuristiques **[33]**.

1.10.2.1 Heuristiques

Le terme heuristique dérive du grec ancien *heuriskêin* et signifie « trouver ». Une heuristique est un algorithme d'approximation qui permet de trouver souvent en temps polynomial, au moins une solution réalisable, tenant compte de la fonction objectif, mais sans assurance sur l'optimalité de la solution trouvée. En général, une heuristique est conçue pour un problème particulier, en étudiant sa structure (heuristique adhoc). Il est à souligner ici qu'une méthode heuristique peut être déterministe ou stochastique **[35,19]**.

Les heuristiques peuvent être classées en deux catégories **[19]** :

• **Méthodes constructives :** qui construisent la solution par une suite de choix partiels et définitifs. Ces méthodes sont connues également sous le nom de méthodes gloutonnes (Greedy en Anglais)

• **Méthodes de fouille locale :** qui partent d'une solution initialement complète et, de manière répétitive, essaient d'améliorer cette solution en explorant son voisinage dans l'espace de recherche. Par exemple, les méthodes de descente substituent à la solution courante une solution voisine si et seulement si la seconde est meilleure.

1.10.2.2 Métaheuristiques

Le terme métaheuristique a été inventé par Fred Glover en 1986 **[35]**, lors de la conception de la recherche taboue. Ce sont des algorithmes d'optimisation basés sur des populations et inspirés de phénomènes naturels. Ils peuvent gérer des espaces de recherche complexes et explorer efficacement des solutions diverses **[36]**. Les méthodes heuristiques précédentes peuvent être piégées dans un optimum local. Les méta-heuristiques essaient d'échapper à ces optima locaux en acceptant temporairement des solutions qui n'améliorent pas la fonction objective. Les métaheuristiques fournissent une méthode générale d'exploitation de l'espace des solutions et recherchent un compromis entre l'exploitation de cet espace et l'exploration par amélioration locale des solutions trouvées **[19]**.

Intensification (exploitation) : Dans l'intensification, les zones prometteuses sont explorées plus en profondeur afin de trouver de meilleures solutions.

Diversification (exploration) : Dans la diversification, les zones non explorées doivent être visitées pour s'assurer que toutes les zones de l'espace de recherche sont explorées, et cela dans le but de trouver de nouvelles meilleures solutions

Les métaheuristiques peuvent être réparties en deux classes métaheuristiques à solution unique et métaheuristiques à population de solution

a. Métaheuristiques à solution unique

Ces méthodes consistent à traiter et améliorer une seule solution afin de trouver la solution optimale, elles se basent généralement sur une recherche locale (recherche par voisinage),

Parmi les métaheuristiques les plus populaires, on retrouve la méthode de Descente (DM : Descent Method) aussi appelée Hill climbing , le Recuit Simulé (SA : Simulated Annealing) , la recherche Tabou (Tabu Search TS), et la Procédure de Recherche Gloutonne Aléatoire Adaptative (GRASP : Greedy Randomized Adaptive Search Procedure) **[33]**.

b. Métaheuristiques à population de solution

Contrairement aux métaheuristiques à solution unique, elles font évoluer simultanément un ensemble de solutions dans l'espace de recherche, On distingue deux grandes classes, les algorithmes évolutionnaires inspirées de la biologie de l'évolution des espèces (les méthodes d'héritage et de la sélection naturelle), et les algorithmes d'intelligence en essaim inspirés du comportement collectif des insectes sociaux ou d'autres animaux **[33]**.

❖ **Algorithmes génétiques**

Sont des algorithmes inspirés de la théorie de l'évolution. Un algorithme évolutif typique est composé de trois éléments essentiels :

(1) une population constituée de plusieurs individus représentant des solutions potentielles au problème en question

(2) un mécanisme d'évaluation de l'adaptation de chaque individu de la population à l'égard de son environnement extérieur

(3) un mécanisme d'évolution composé d'opérateurs permettant d'éliminer certains individus et de produire de nouveaux individus à partir des individus sélectionnés **[19].**

❖ **Les algorithmes basé essaim**

Les algorithmes d'intelligence par essaim (SI, Swarm Intelligence) basés sur l'intelligence des animaux ou leur capacité individuelle est très limitée, peuvent conjointement effectuer de nombreuses tâches complexes. L'intelligence collective observée notamment chez les insectes sociaux (les fourmis et les abeilles), les animaux évoluant en groupe comme les oiseaux migrateurs, les loups, les lions, etc. D'autres intelligences collectives sont observées dans certains phénomènes biologiques comme les systèmes immunitaires. Ce sont des systèmes composes d'agents simples qui peuvent **[33]** :

- Partager des informations entre eux;
- S'auto-organiser et évoluer ensemble;
- Être très efficace pour le co-apprentissage;
- Faire du parallélisme pour des problèmes complexes et en temps réel.

Plusieurs algorithmes ont été proposés ces dernières décennies, on donne ci-après, de façon non exhaustive, les différentes métaheuristiques selon l'ordre chronologique : Optimisation d'essaims de particules (PSO: Particle Swarm Optimization), Optimisation par colonie de fourmis (ACO: Ant Colony Optimization), optimisation par essaim de poissons (FSA: Fish Swarm Algorithm), algorithme d'optimisation bactérienne de la recherche de nourriture (BFOA: Bacterial Foraging Optimization Algorithm), qui s'inspire du comportement de recherche de nourriture et de reproduction des bactéries, optimisation par sauts de grenouille partitionnes (SFLA: Shuffled Frog Leaping Algorithm), optimisation par colonie d'abeilles(ABC: Artificial Bee Colony optimization), l'algorithme d'optimisation basée sur la biogéographie insulaire (BBO : Biogeography-Based Optimization , Recherche de coucou (CS: Cuckoo Search) , algorithme de luciole (FA: Firey Algorithm , optimisation des loup gris (GWO : Grey Wolf Optimizer) , l'algorithme d'optimisation des fourmilions (ALO: Ant Lion Optimizer) **[32]**.

Algorithme d'optimisation d'Ant lion (ALO)

Dans cette section, nous allons définir l'algorithme inspiré de la nature, appelé Ant Lion Optimizer (ALO). L'algorithme ALO imite le mécanisme de chasse des fourmis dans la nature. Cinq étapes principales de la chasse aux proies, telles que la marche aléatoire des fourmis, la construction de pièges, le piégeage des fourmis dans les pièges, la capture des proies et la reconstruction des pièges, sont mises en œuvre.

Inspiration

Les fourmilions appartiennent à la famille des Myrmeleontidae et à l'ordre des Neuroptères (insectes à ailes en filet). Le cycle de vie des antlions comprend deux phases principales : les larves et les adultes. La durée de vie totale naturelle peut atteindre 3 ans, principalement à l'état larvaire (trois à cinq semaines seulement à l'âge adulte). Les fourmilions se métamorphosent dans un cocon pour devenir adultes. Ils chassent principalement à l'état

larvaire et la période adulte est consacrée à la reproduction, leur nom provient de leur comportement de chasse unique et de leur proie préférée **[37]**.

Une larve de fourmilion creuse une fosse en forme de cône dans le sable en se déplaçant le long d'une trajectoire circulaire et en projetant du sable avec sa mâchoire massive. **[37]**.

Après avoir creusé le piège, la larve se cache sous le fond du cône (comme un prédateur qui attend) et attend que des insectes (de préférence des fourmis) soient piégés dans la fosse, comme le montre la figure 1.11 **[37]**.

Le bord du cône est suffisamment tranchant pour que les insectes tombent facilement au fond du piège. Lorsque le fourmilion se rend compte qu'une proie se trouve dans le piège, il tente de l'attraper. Cependant, les insectes ne sont généralement pas attrapés immédiatement et tentent de s'échapper du piège. Dans ce cas, les fourmilions lancent intelligemment des sables vers le bord de la fosse pour faire glisser la proie vers le fond de la fosse **[37]**.

Lorsqu'une proie est prise dans la mâchoire, elle est entraînée sous le sol et consommée.

Après avoir consommé la proie, les fourmilions jettent les restes à l'extérieur de la fosse et amendent la fosse pour la prochaine chasse.

Un autre comportement intéressant qui a été observé dans le style de vie des fourmilions est la pertinence de la taille du piège et de deux choses : le niveau de faim et la forme de la lune. Les fourmilions ont tendance à creuser des pièges plus grands lorsqu'elles ont plus faim et/ou lorsque la lune est pleine. Ils ont évolué et se sont adaptés de cette manière pour améliorer leurs chances de survie. On a également découvert qu'un fourmilion n'observe pas directement la forme de la lune pour décider de la taille du piège, mais qu'il dispose d'une horloge lunaire interne pour prendre de telles décisions **[37]**.

L'algorithme ALO s'inspire principalement du comportement de recherche de nourriture des larves de fourmilions.

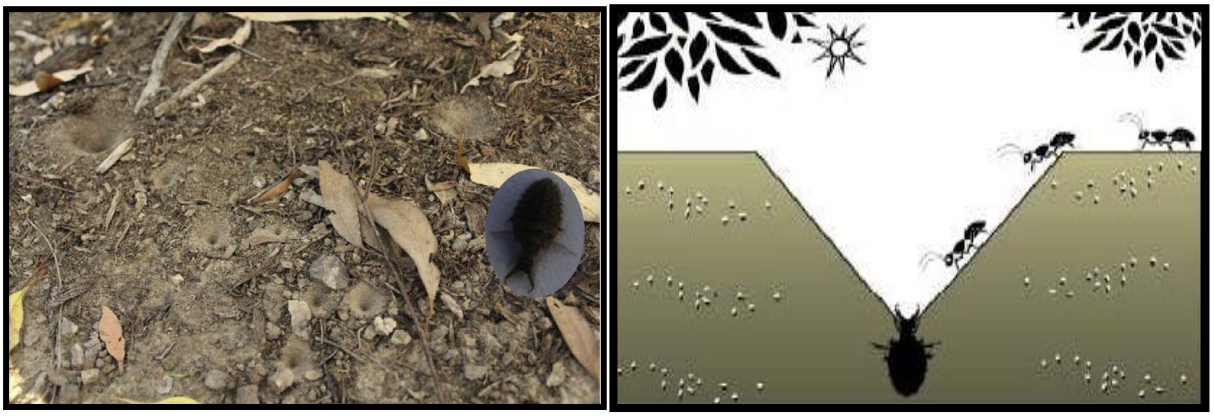


Figure 1.11 : Pièges en forme de cône et le comportement de chasse des antlions.

- **Les opérateurs de l'algorithme ALO**

L'algorithme d'ALO simule cinq étapes principales de la chasse :

a. La marche aléatoire de proie

La marche aléatoire principale dans cet algorithme est la suivante :

$$X(t) = [0, cumsum(2r(t_1) - 1), cumsum(2r(t_2) - 1), \ldots. \ cumsum(2r(t_n)] \qquad (1.3)$$

Où ***cumsum*** calcule la somme cumulative, ***n*** est le nombre maximum d'itérations, ***t*** montre l'étape de marche arbitraire et ***r(t)*** est une fonction stochastique définie comme suit :

$$r(t) = \begin{cases} 1 \rightarrow rand > 0{,}5 \\ 0 \rightarrow rand \leq 0{,}5 \end{cases} \qquad (1.4)$$

La marche aléatoire discutée ci-dessus est utilisée dans l'équation suivante pour mettre à jour la position des fourmis :

$$X_i^t = \frac{(X_i^t - a_i) \times (d_i - c_i^t)}{(d_i^t - a_i)} + c_i \qquad (1.5)$$

Où a_i est le minimum de la marche aléatoire de la variable i, d_i est le maximum de la marche aléatoire dans la variable i.

b. Le piégeage des fourmis dans les pièges

L'impact des fourmilions sur le mouvement des fourmis est modélisé comme suit :

$$c_i^t = Antlion_j^t + c^t \qquad (1.6)$$

$$d_i^t = Antlion_j^t + d^t \qquad (1.7)$$

Où c^t et d^t sont le minimum et le maximum de toutes les variables à l'itération *t*, et $Antlion_j^t$ est une position de l'antlion sélectionnée à l'itération *t*.

c. La construction des pièges

La construction des pièges se fait avec une roue de roulette utilisée. L'opérateur de roulette sélectionne les antlions en fonction de leur fitness pendant l'optimisation. Ce mécanisme donne de grandes chances aux antlions (fourmilions) plus aptes à attraper les fourmis.

d. Glissement des fourmis vers le fourmilions

Les fourmis glissantes vers le fourmilion "Ant Lion" calculées par les équations (1.8) et (1.9)

$$c^t = \frac{c^t}{I} \qquad (1.8)$$

$$d^t = \frac{d^t}{I} \qquad (1.9)$$

Où I = rapport, c^t est le minimum des variables totales à l'itération *t*, et d^t est le vecteur contenant le maximum des variables totales à l'itération *t*.

e. La capture des proies et la reconstruction des pièges

La proie de chasse et le réarrangement des fosses calculées par l'équation (1.10)

$$Antlion_j^t = Ant_i^t \quad \text{si} \quad f(Ant_i^t) > f\left(Antlion_j^t\right) \qquad (1.10)$$

Où *t* montre l'itération courante (actuelle), $Antlion_j^t$ montre la position du fourmilions *j* sélectionnée à l'itération *t*, et Ant_i^t indique la position de la fourmi *i* à l'itération *t*.

f. Élitisme

L'élitisme est une caractéristique importante des algorithmes évolutionnaires qui leur permet de maintenir la meilleure solution obtenue à n'importe quelle étape du processus chaque itération est sauvegardé et sa position est supposée affecter chaque fourmi dans un espace de recherche courant. Ceci est défini par les prochaines marches aléatoires de proies, comme suit :

$$\boldsymbol{Ant_i^t = \frac{R_A^t + R_E^t}{2}} \qquad (1.11)$$

Où R_A^t est la marche aléatoire autour de l'antlion sélectionnée par la roue de roulette à l'itération, R_E^t est la marche aléatoire autour de l'élite à l'itération, et Ant_i^t indique la position de la fourmi *i* à l'itération *t.*

Nous détaillerons son algorithme lors de son implémentation dans le chapitre 3.

Méthode d'essaim particulaire (PSO)

L'optimisation par essaim particulaire (en anglais : Particle Swarm Optimization) est une méta-heuristique d'optimisation stochastique à base de population de solutions, proposée en 1995 par James Kennedy (socio-psychologue) et Russel Eberhart (ingénieur électricien) pour la résolution des problèmes d'optimisation, plus particulièrement les problèmes à variable, continue. Elle est basée sur le comportement social des individus évoluant en essaim, c'est à dire, les « interactions sociales » entre des « agents » appelés « particules » représentant un « essaim », dans le but d'atteindre un objectif donné dans un espace de recherche commun où chaque particule a une certaine capacité de mémorisation et de traitement de l'information. Dans PSO le comportement social est modélisé par une équation mathématique permettant de guider les particules durant leur processus de déplacement. Le déplacement d'une particule est influencé par trois composantes : la composante d'inertie, la composante cognitive et la composante sociale. Chacune de ces composantes reflète une partie de l'équation, Figure 1.12 **[94]** :

1. La composante d'inertie : la particule tend à suivre sa direction courante de déplacement

2. La composante cognitive : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée

3. La composante sociale : la particule tend à se diriger vers le meilleur site atteint par ses voisines.

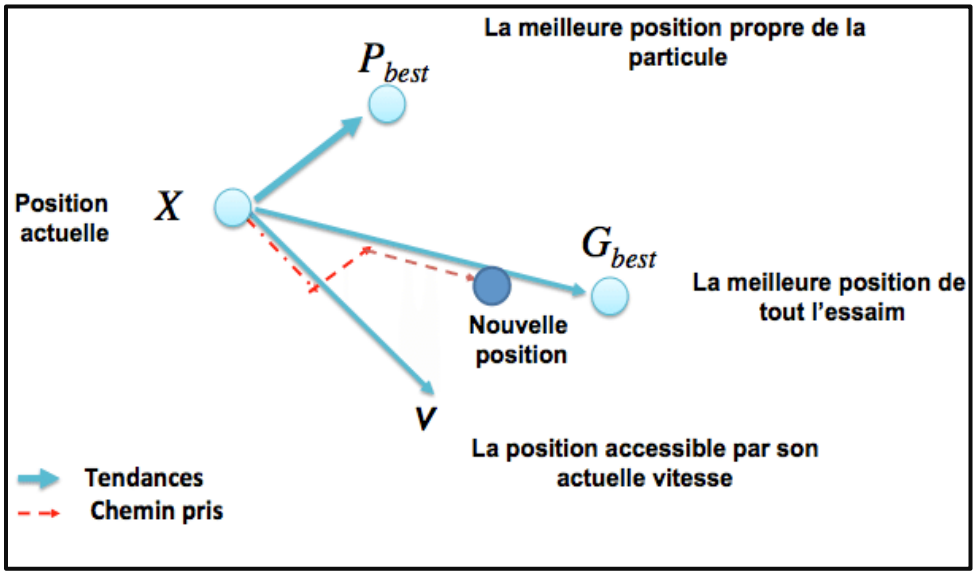


Figure 1.12 : Déplacement d'une particule**.**

- **Formalisation et algorithme**

Une particule *i* de l'essaim dans un espace de recherche de dimension *D* est caractérisée, à l'instant *t*, par **[94]** :

• X : sa position dans l'espace de recherche,

• V : sa vitesse,

• Pb : la position de la meilleure solution par laquelle elle est passée,

• Pg : la position de la meilleure solution connue de tout l'essaim,

• f (Pb) : la valeur de fitness de sa meilleure solution,

• f (Pg) : la valeur de fitness de la meilleure solution connue de tout l'essaim, Le déplacement de la particule *i* entre les itérations *t* et *t+1* se fait selon les deux équations (1.13) et (1.14) :

$$\begin{cases} V_{iD}(\mathrm{t}+1) = V_{iD}(\mathrm{t}) + c_1\, r_1\big(Pb_{iD}(\mathrm{t}) - X_{iD}(\mathrm{t})\big) + c_2\, r_2\big(Pg_{iD}(t) - X_{iD}(\mathrm{t})\big) & (\mathbf{1.12}) \\ X_{iD}(\mathrm{t}) = X_{iD}(\mathrm{t}) + V_{iD}(\mathrm{t}) & (\mathbf{1.13}) \end{cases}$$

Où :

• $c_1$et c_2 : deux constantes qui représentent les coefficients d'accélération.

• r1 et r2 : deux nombres aléatoires tirés de l'intervalle [0,1].

- **Optimisation standard des essaims de particules (S-PSO)**

Les auteurs Shi et Elberhart **[97]** ont introduit un nouveau paramètre ω, appelé poids d'inertie d'une particule. Avec ce nouveau paramètre, l'équation de mise à jour de la vitesse (1 .13) est devenue l'équation (1.15) :

$$V_{iD}(\mathrm{t}+1) = w\, V_{iD}(\mathrm{t}) + c_1\, r_1\big(Pb_{iD}(\mathrm{t}) - X_{iD}(\mathrm{t})\big) + c_2\, r_2\big(Pg_{iD}(t) - X_{iD}(\mathrm{t})\big) \qquad (1.14)$$

L'algorithme d'optimisation par essaim de particules (PSO) avec cette amélioration est communément appelé PSO standard.

L'algorithme de base de la méthode PSO proposé par **[96]**, commence par une initialisation aléatoire des particules dans leur espace de recherche, en leurs attribuant une position et une vitesse initiales. À chaque itération de l'algorithme les particules se déplacent selon les équations (1.12) et (1.13) et les fonctions objectif (fitness) des particules sont calculées afin de pouvoir calculer la meilleure position de toutes *Pg.* La mise à jour des *Pb* et *Pg* est faite à chaque itération suivant l'algorithme. Le processus est répété jusqu'à satisfaction du critère d'arrêt **[95].**

Algorithme 1.1: Algorithme PSO (Particle Swarm Optimization)

Algorithme PSO

Initialiser les paramètres et la taille S de l'essaim;

Initialiser la vitesse et les positions aléatoires des particules dans chaque dimension de l'espace de recherche;

Pour chaque particule, Pb=X ;

Calculer f(x) pour chaque particule ;

Calculer Pg // la meilleure Pb

Tant que (la condition d'arrêt n'est pas vérifiée) **faire**

Pour (i allant de 1 à S) **faire**

Calculer la nouvelle vitesse à l'aide de l'équation (1.12) ;

Trouver la nouvelle position à l'aide de l'équation (1.13) ;

Calculer f(x) de chaque particule ;

Si f(x) est meilleure que f(Pb) **alors**

Pb=X ;

Si f(Pb) est meilleure que f(Pg) **alors**

Pg=Pb ;

Fin pour

Fin tant que

Fin

Afficher la meilleure solution trouvée Pg

1.11 Conclusion

Dans ce chapitre, nous avons exploré les concepts fondamentaux de l'Internet des objets (IoT), du Fog Computing et du Cloud Computing, en mettant en lumière leurs caractéristiques distinctes, leurs complémentarités, et leur rôle central dans le paysage technologique actuel. L'optimisation, en tant que processus clé, permet de maximiser la performance et l'efficacité de ces systèmes interconnectés, assurant une utilisation optimale des ressources et une réponse rapide aux besoins des utilisateurs. Ensemble, ces technologies et techniques d'optimisation créent une base solide pour le développement de solutions intelligentes et évolutives.

Dans le chapitre 2, nous allons détailler les différentes solutions d'allocation de services proposées dans le Fog Computing.

2 Chapitre 2 : Allocation de services dans le Fog Computing

2.1 Introduction

En tant que nouveau paradigme, le Fog Computing pose des défis anciens et nouveaux, et l'un des principaux problèmes ouverts est la gestion des services et plus précisément le problème de placement des services. En effet, l'un des principaux obstacles à l'adoption du Fog est de savoir "comment déployer efficacement des services sur les nœuds Fog disponibles".

Contrairement aux centres de données Cloud, les dispositifs Fog sont géographiquement dispersés, limités en ressources et très dynamiques, ce qui rend le problème assez difficile. Par conséquent, la définition d'un provisionnement efficace, effectif et équitable pour les applications IoT sera importante pour fournir et garantir ainsi des services garantis de bout en bout aux utilisateurs finaux. De plus, selon le contexte et l'intérêt, différents aspects peuvent être mis en avant : l'utilisation des ressources **[78]**, la qualité de service **[50, 76, 77]**, la qualité de l'expérience (QoE) **[63]**, etc.

Ces dernières années, plusieurs recherches ont été menées et tentent de résoudre ce problème. Différents hypothèses, caractéristiques et stratégies ont été envisagées pour proposer un placement de service efficace.

Après avoir présenté notre contexte de recherche, nous nous intéressons au fonctionnement de différentes solutions récentes et pertinentes d'allocation de services qui ont été proposées pour le Fog Computing. Dans un premier temps, nous définissons le service dans le Fog Computing. Dans un second temps, nous détaillerons les méthodes pour l'allocation de services dans le Fog**.**

2.2 Les services dans le Fog Computing

Un service est une unité indépendante virtuelle avec une capacité de calcul, de communication et de stockage qui interagit avec d'autres services et/ou avec les équipements utilisateurs pour effectuer une tâche donnée **[23]**.

Le consortium OASIS (Organization for the Advancement of Structured Information Standards) définit un service comme étant un mécanisme permettant d'accéder à une ou

plusieurs fonctionnalités. La description de ces fonctionnalités sont décrites dans une spécification, appelée interface de service, Il n'y a pas de détails sur la manière dont les fonctionnalités du service sont réalisées, un service est accessible au travers d'une interface conforme à un ensemble de contraintes et de politiques d'accès **[33]**.

Du point de vue de l'utilité, les services fournis par différentes applications IoT peuvent être classés en deux groupes principaux, à savoir les services élastiques traditionnels et les services inélastiques en temps réel **[42]**.

Les premiers font généralement référence à des services tels que le transfert de fichiers, l'analyse de données et la navigation sur le web, où chaque service atteint une fonction d'utilité strictement croissante et concave pour refléter ses performances en matière de qualité de service. En comparaison, les services inélastiques en temps réel sont généralement fournis par des applications en temps réel telles que les applications de diffusion audio, vidéo et multimédia. Ces services ont un seuil de bande passante intrinsèque par nature et adoptent des fonctions de type sigmoïde pour décrire la QoS particulière **[43,41]**.

2.3 Placement de service dans le Fog Computing

Le problème de placement de service a été largement discuté dans la littérature et plusieurs propositions ont émergé. Basées sur différentes descriptions d'applications, des hypothèses de réseau et des résultats attendus, ces solutions sont généralement difficiles à comparer entre elles. Dans cette section, nous allons voir la méthodologie généralement utilisée pour aborder le SPP (Service Placement Problem) nous donnons un aperçu des aspects suivants : énoncé du problème, taxonomie du placement, stratégies d'optimisation. Ces éléments nous permettront de décrire clairement le problème.

2.4 Énoncé du problème

L'énoncé du problème SPP passe par la description des trois parties suivantes **[92]** :

- Le modèle d'infrastructure ;
- Le modèle d'application ;
- Le modèle de déploiement et ses contraintes associées.

2.4.1 Modèle d'infrastructure

L'infrastructure physique du Fog (voir Fig 1.9) se compose respectivement d'un ensemble d'appareils sans capacités de calcul (capteurs et actionneurs), et d'un ensemble de ressources qui possèdent une puissance de calcul et/ou une capacité de stockage (nœuds de brouillard et centres de données en nuage). En raison de leur structure physique **[64]**, les nœuds de brouillard ont des ressources limitées et sont hétérogènes les uns par rapport aux autres. Tout appareil équipé de ressources de calcul (CPU, mémoire, stockage, bande passante) peut être considéré comme un FN potentiel, comme les routeurs, les petits serveurs, les points d'accès, les ordinateurs portables ou les passerelles.

Le réseau d'infrastructure est généralement abstrait en tant que graphe connecté où les sommets désignent l'ensemble des dispositifs IoT, des nœuds de brouillard et des serveurs cloud, et les arêtes désignent les liens entre les nœuds. Nous mentionnons ci-après les types de ressources et les caractéristiques les plus courants décrits dans la littérature pour décrire l'infrastructure du brouillard.

2.4.1.1 Type de ressources

- Informatique : serveurs, PC, cloudlets **[45, 72]**, etc.
- Réseau : passerelles, routeurs, commutateurs, stations de base, etc.
- Stockage : tout nœud pouvant fournir du stockage.
- Mobile : véhicules, smartphones, etc.
- Abstraction de point de terminaison : capteurs, actionneurs (par exemple, appareils GPS, capteurs sans fil, caméras, collecteur de voix, radar, etc.).

2.4.1.2 Caractéristiques

- Informatique : puissance du CPU, nombre de cœurs, RAM, autonomie de la batterie, etc.
- Réseau : Type : sans fil, filaire ; Capacités : latence, bande passante, taux d'erreur, etc.
- Stockage : Disque, etc.
- Virtualisation : machines virtuelles, conteneurs, etc.
- Matériel : GPU, etc.

2.4.2 Modèles d'application

Plusieurs abstractions et définitions de modèles sont utilisées dans la littérature pour caractériser les applications générées par les appareils IoT et traitées sur les ressources Fog et les serveurs Cloud. D'après les articles examinés, nous identifions les principales descriptions suivantes : a) un service monolithique, b) un ensemble de composants interdépendants, etc.) un graphe connecté **[92]**.

a) **Service monolithique :** L'application envoyée par les utilisateurs finaux ou les appareils IoT est représentée sous la forme d'un composant unique (service monolithique). A titre d'exemple, on peut citer le cas d'une application de traitement d'images ou d'une instance de données qui doit être traitée ou stockée dans un seul nœud physique. L'application peut être définie dans ce cas comme un service monolithique.

b) **Ensemble de services interdépendants :** Ce cas suppose que l'application est prédécoupée en un ensemble de composants (services), chacun effectuant une opération spécifique (fonctionnalité) de l'application. Dans ce cas, les dépendances entre les composants de l'application ne sont pas prises en compte.

c) Graphe connecté : L'application, dans ce cas, est composée d'un ensemble de composants interdépendants représentés sous la forme d'un graphe connecté. Les sommets représentent les composants de traitement/calcul de l'application, et les arêtes représentent les interdépendances et les besoins de communication entre les nœuds **[55]**.

On peut identifier différentes topologies de graphe, parmi lesquelles on trouve respectivement le graphe linéaire, le graphe d'application arborescente et le DAG (Directed Acyclic Graph). La topologie d'application DAG est la plus utilisée car elle modélise un large éventail d'applications IoT réalistes telles que le traitement vidéo **[47, 49, 74]**, les jeux **[90]** ou les applications de santé **[53]**. La figure 2.1 **[56]** illustre un exemple d'application d'assistance cognitive.

En ce qui concerne les exigences de l'application, nous pouvons en résumer quelques-unes **[92]** :

Calcul : Puissance CPU, nombre de cœurs, RAM, etc.

Orienté réseau : Bande passante, latence, taux d'erreur, gigue (par liaison, de bout en bout).

Orienté tâche : Délai d'exécution.

Orienté localisation : l'application doit s'exécuter dans un lieu géographique spécifique (par exemple à Paris) ; l'application ne peut s'exécuter que sur un certain nœud Fog, etc.

2.4.3 Schéma de déploiement

Le problème de placement d'application définit un schéma de mappage par lequel les composants et les liens des applications sont mappés sur un graphe d'infrastructure (c'est-à-dire des dispositifs informatiques et des liens physiques). La figure 2.1 **[56]** montre un exemple de mappage d'une application modélisée sous forme de DAG (Fig. 2.1.a) sur des nœuds Fog disponibles (Fig. 2.1.b).

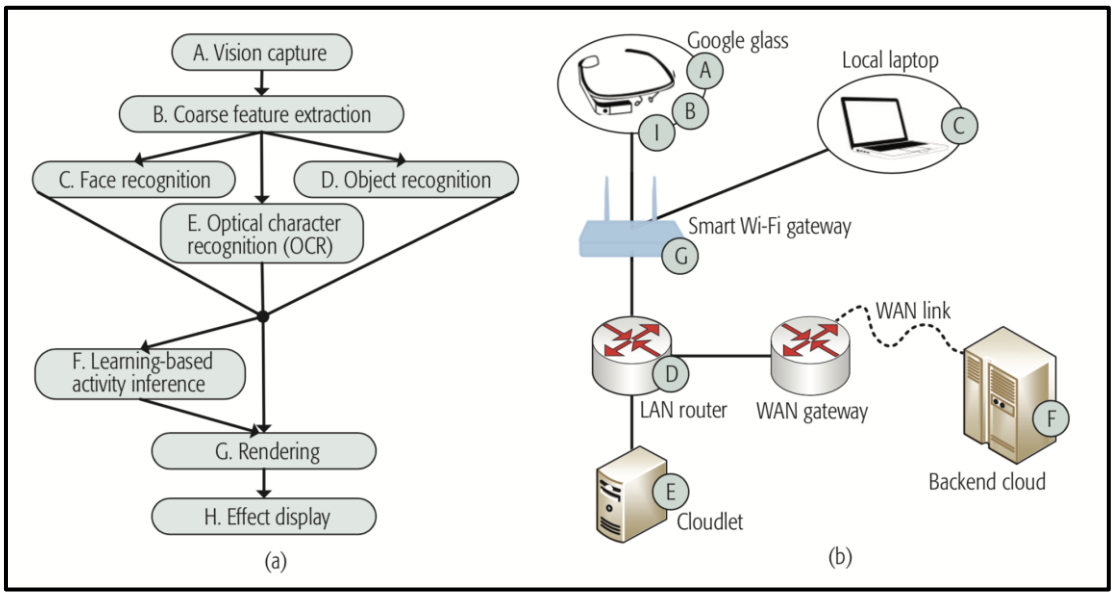


Figure 2.1 : Application d'assistance cognitive, illustrée dans (a), et déployée sur un réseau Fog, tel que présenté dans (b).

En général, le placement d'application consiste à trouver les ressources disponibles dans le réseau (nœuds et liens) qui satisfont aux exigences des applications, respectent les contraintes et optimisent l'objectif (s'il en existe un). Par exemple, respecter les exigences des applications

(services), ne pas dépasser les capacités des ressources, satisfaire les contraintes de localité, minimiser l'énergie consommée, etc. Les fournisseurs de services doivent tenir compte de ces contraintes pour, d'abord, limiter l'espace de recherche et ensuite, fournir un placement optimal ou quasi-optimal. Nous proposons ci-après de décrire certaines des contraintes les plus souvent prises en compte dans la littérature **[92]**.

- **Contraintes de ressources (Ressource Constraints CR)** Un nœud d'infrastructure est limité par des capacités finies en termes de CPU, RAM, stockage, bande passante, etc. Par conséquent, lors du placement d'applications (composants de service), nous devons respecter les besoins en ressources, c'est-à-dire garantir que les ressources des composants déployés sur les nœuds d'infrastructure ne dépassent pas leurs capacités.
- **Contraintes de réseau (Network Constraints CN)**. Une liaison réseau peut également être limitée par des contraintes telles que la latence, la bande passante, etc., et ces contraintes doivent être satisfaites lors du déploiement des applications.
- **Contraintes applicatives** : Nous soulignons ici deux types de contraintes applicatives :
    - **Exigence de localité (Locality requirement CL)**. L'exigence de localité restreint généralement l'exécution de certains services à des emplacements spécifiques. En raison de matériel spécifique, d'exigences de confidentialité ou d'une politique donnée, certains composants peuvent être limités à être déployés sur des zones ou des périphériques spécifiques. Les contraintes de localité peuvent être basées sur : un ensemble de nœuds Fog **[85, 89]** ; un emplacement géographique spécifique utilisant le GPS par exemple **[71]** ; imposer une co-localisation des composants **[44]**.
    - **Sensibilité au délai (Delay Sensitivity CD)**. Certaines applications peuvent spécifier un délai pour l'opération de traitement ou le déploiement de l'ensemble de l'application dans le réseau. Cette contrainte est généralement spécifiée en définissant un seuil à ne pas dépasser.

2.5 Taxonomie de placement de service

Pour aborder SPP, il faut tenir compte de certaines spécificités et critères lors de la conception des stratégies de déploiement. C'est ce que l'on appelle la taxonomie de placement

de service montré dans la figure 2.2 **[92]**. Dans ce mémoire, nous proposons de prêter attention aux quatre aspects principaux suivants :

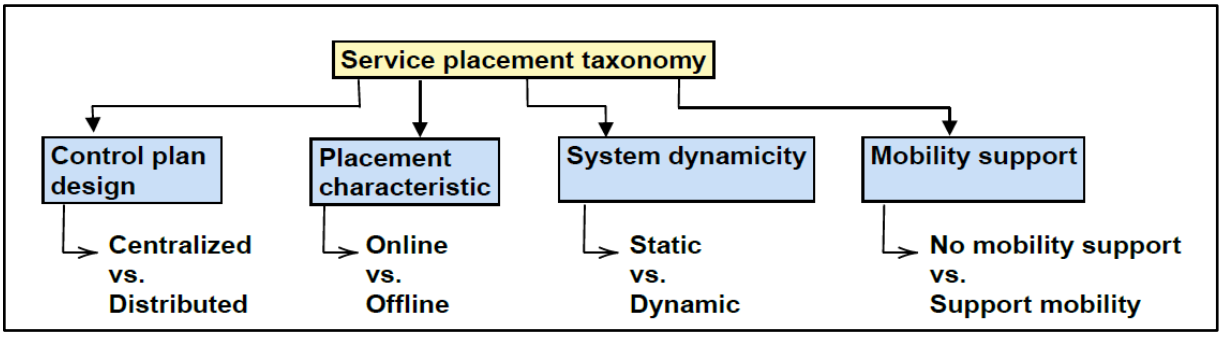


Figure 2.2 : Taxonomie de placement de services.

En tant que tel, la première spécificité considère si la coordination du mappage se fait de manière centralisée ou distribuée. La deuxième se base sur le fait que le problème soit traité comme un déploiement hors ligne (statique) ou en ligne (dynamique). La troisième propose d'observer si la question de la dynamicité du système est prise en charge ou non (c'est-à-dire gérer les changements dans le système). Enfin, la quatrième caractéristique décrit si la mobilité des utilisateurs finaux et/ou des appareils Fog est prise en charge par la solution proposée **[92]**.

2.5.1 Conception du plan de contrôle : Évaluation d'approches centralisées et distribuées

Le développement d'une stratégie de placement et de gestion des services commence par la sélection de la stratégie de coordination à adopter. Deux modèles de plan de contrôle courants sont présentés dans cette section : la coordination centralisée et la coordination distribuée. Pertinentes pour les systèmes multicouches et géographiquement distribués, ces approches sont relativement différentes et chacune a ses propres avantages et inconvénients, comme présentés ci-dessous.

a) **Centralisé**

Un plan de contrôle centralisé nécessite des informations globales sur les demandes des applications et les ressources de l'infrastructure pour prendre et diffuser des décisions de déploiement globales. L'avantage des algorithmes de placement centralisés est qu'ils peuvent

potentiellement trouver une solution globalement optimale. Cependant, ils sont vulnérables en matière d'évolutivité et de complexité de calcul.

Un grand nombre de travaux considèrent un plan de contrôle centralisé pour aborder le problème de placement de service (SPP). Parmi ces travaux, on cite celui de Hong et al. **[58]** qui envisage un coordinateur central pour prendre les décisions de déploiement des applications IoT sur l'infrastructure Fog.

b) **Distribué**

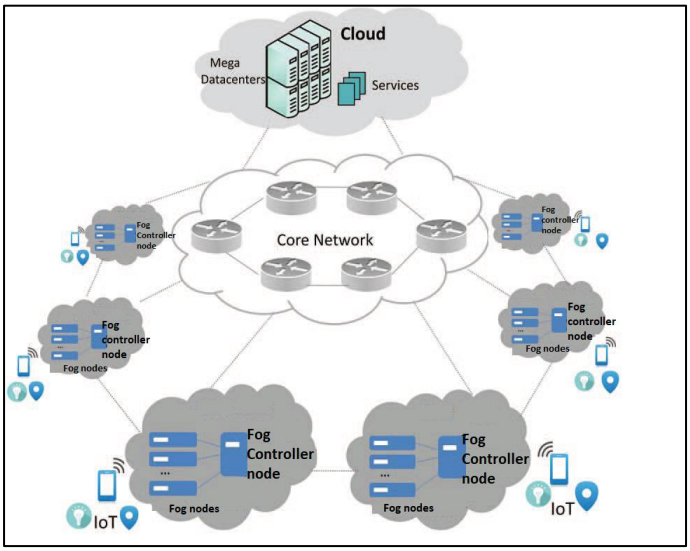


Figure 2.3 : Exemple d'un plan de contrôle distribué.

Contrairement à la solution centralisée, une approche distribuée utilise plusieurs nœuds d'autorité et d'orchestration pour contrôler le mappage des services. Généralement distribués dans le réseau (comme illustré sur la figure 2.3 **[92]**), les éléments de gestion calculent les décisions de placement en fonction des ressources et des informations locales. Ce plan de contrôle est plus flexible et peut être plus efficace pour gérer les changements dynamiques de l'infrastructure comme le Fog Computing, sans avoir recours à des calculs à l'échelle du réseau. L'approche distribuée permet d'aborder les problèmes d'évolutivité et de connaissance de la localité et permet de fournir des services qui s'adaptent le mieux au contexte local. Cependant, aucune garantie n'est fournie quant à l'optimalité globale des solutions calculées.

Parmi les travaux qui ont envisagé un plan de contrôle distribué, on peut citer celui de Wang et al. **[83]** qui propose une architecture basée sur le Fog Computing composée de nœuds

Fog et d'une coordination entre ces nœuds. La sous-couche des FN (Fog Nodes) gère les tâches nécessitant un traitement en temps réel. Les tâches complexes qui nécessitent plus de capacités de calcul sont transmises à la sous-couche de coordination des FN ou acheminées vers le Cloud.

2.5.2 Placement hors ligne par rapport au placement en ligne

Le problème de placement de service peut être abordé de manière hors ligne (statique) ou en ligne (dynamique). Plus précisément, on dit que le placement est hors ligne s'il prend une décision de déploiement au moment de la compilation, où toutes les informations requises sont disponibles. Il nécessite des informations complètes sur les activités du système et fournit des solutions qui répondent aux exigences données. Pour le placement en ligne, les décisions de déploiement sont prises pendant l'exécution du système. Les décisions se basent à la fois sur les caractéristiques du processus et sur l'état actuel du système **[92]**.

Dans la plupart des cas d'utilisation réels, le SPP doit être abordé comme un problème en ligne. Cela signifie que les algorithmes associés doivent prendre en compte les services à leur arrivée, plutôt que de calculer leur placement à l'avance avant leur exécution. On remarque qu'un placement en ligne peut s'adapter au comportement dynamique (aux changements) du système, mais il ne peut pas exploiter au mieux les ressources du système (fournir une décision de placement optimale) **[92]**.

À titre d'exemple d'algorithmes de placement hors ligne fournis dans la littérature, on peut citer **[67, 77]** qui supposent une connaissance complète des informations pour le réseau Fog. Pour le placement en ligne, on peut citer par exemple le travail de Lee et al. **[62]** qui propose une stratégie de placement en ligne qui minimise la latence de calcul pour un système Fog en cas d'incertitude sur la disponibilité des FN. Le cadre d'optimisation en ligne proposée suggère d'observer séquentiellement les informations sur le réseau.

2.5.3 Placement statique par rapport au placement dynamique

Ce critère aborde le fait que les propositions gèrent ou non la dynamicité du système. On peut identifier respectivement deux aspects : la dynamicité de l'infrastructure Fog et la dynamicité des applications. Le réseau Fog est très dynamique, des entités rejoignant et quittant le réseau en raison de l'instabilité des liaisons réseau ou de pannes. Les capacités des ressources peuvent également varier dans le temps.

Du point de vue de l'application, la structure du graphe d'application peut évoluer au fil du temps en réponse aux changements des conditions réelles. De nouvelles sources ou de nouveaux dispositifs peuvent apparaître, ou des dispositifs existants peuvent disparaître (ajout de nouvelles caméras, panne de certains composants, les utilisateurs peuvent décider à tout moment de commencer ou d'arrêter l'envoi de leurs données pour un service). De plus, des changements dans les informations de l'application peuvent également être observés (par exemple, sur la quantité de données, les exigences des composants).

Pour faire face au caractère dynamique de l'infrastructure Fog et/ou des applications, il est nécessaire de définir des stratégies réactives capables de déterminer quand une adaptation est nécessaire, de fournir un mécanisme transparent et d'offrir la qualité de service satisfaisante. Ainsi, une approche est dite dynamique si la stratégie de placement fournie est capable de déployer de nouveaux services, de remplacer ou de libérer des services déjà déployés, afin de respecter les contraintes de QoS et d'optimiser un objectif donné (s'il en existe un).

Parmi les travaux qui proposent de gérer la nature dynamique de l'environnement Fog, parmi lesquels on peut citer le travail de Yousefpour et al. **[88]** qui propose un provisionnement dynamique des services dans l'infrastructure Fog qui satisfait aux exigences de QoS et aux accords de niveau de service (SLA), et minimise le coût des ressources. Pour gérer la dynamicité des applications IoT, Mahmud et al. **[64]** proposent une politique qui détermine dynamiquement les nœuds hôtes pour les composants déployés et gère les changements soudains de fréquences des services reçus.

2.5.4 Prise en charge de la mobilité

La gestion de la mobilité est un défi majeur dans le Fog Computing. Fournir une solution qui prend en charge la mobilité des utilisateurs finaux et/ou des appareils Fog et garantit que les utilisateurs reçoivent toujours les services associés et les performances souhaitées sans interruption est un problème complexe dans le Fog Computing.

Les changements fréquents d'emplacement des nœuds finaux (ou des nœuds Fog comme les smartphones, voiture connectée, etc) peuvent entraîner des retards excessifs ou des pertes de paquets. Dans une telle situation, le gestionnaire du système (coordinateur) devrait être capable de déplacer le service de manière transparente des anciens appareils vers les nouveaux afin d'en assurer la continuité.

Dans **[48, 59, 68, 69, 73, 80, 81, 82, 83]**, par exemple, les auteurs tentent d'aborder le problème de la mobilité des utilisateurs finaux en proposant des approches de placement dynamique. Dans **[73]**, Saurez et al proposent de gérer la mobilité des utilisateurs finaux en proposant une stratégie basée sur les décisions suivantes : "Quand migrer" en fonction du paramètre de latence ; et "Où migrer" en fonction de la proximité d'un FN par rapport aux appareils mobiles et du nœud de traitement actuel.

2.6 Stratégies d'optimisation

Le problème de placement de service dans une infrastructure Fog a été abordé en fonction de plusieurs objectifs différents, avec des formulations différentes et des propositions d'algorithmes variés. Cette section traite des objectifs possibles que l'on peut poursuivre dans de tels systèmes, des métriques envisagées pour évaluer le déploiement fourni, la formulation du problème utilisée par les propositions existantes, les stratégies de résolution et les algorithmes utilisés pour résoudre le SPP.

2.6.1 Objectifs et métriques d'optimisation

Nous proposons de présenter d'abord une classification globale des stratégies d'optimisation proposées dans la littérature. D'une part, nous distinguons les deux catégories suivantes : l'optimisation mono-objectif et l'optimisation multi-objectif. D'autre part, nous présentons les métriques les plus souvent prises en compte lors de l'optimisation **[92]**.

2.6.1.1 Optimisation mono-objectif par rapport multi-objectif

L'optimisation mono-objectif propose d'optimiser une seule fonction objectif. L'optimisation multi-objectif, quant à elle, propose d'optimiser simultanément un ensemble de fonctions objectifs **[65]**.

2.6.1.2 Métriques d'optimisation

L'optimisation peut être réalisée pour maximiser ou minimiser les valeurs des métriques suivantes **[92].**

- **Latence**

Faible latence pour les applications sensibles aux retards. La réduction de la latence a attiré l'attention de plusieurs articles examinés. En effet, plusieurs travaux visent à minimiser la latence des services déployés sur les ressources disponibles tout en respectant l'ensemble des exigences et des contraintes. Par exemple, dans **[86, 87]**, les auteurs proposent de minimiser le délai de service du déploiement d'applications IoT sur le framework IoT-Fog-Cloud.

2.6.1.2.1 Utilisation des ressources

Un enjeu important du Fog Computing est d'optimiser l'utilisation des ressources tout en déployant le maximum de services sur les nœuds de Fog appropriés. Parmi les travaux trouvés dans la littérature qui étudient cet objectif, on peut citer celui de Hong et al**. [58]** qui propose des décisions de déploiement tout en maximisant le nombre de requêtes d'analyse IoT satisfaites. Skarlat et al. dans **[75]** proposent également de maximiser le nombre de requêtes d'application satisfaites en priorisant les applications ayant la deadline la plus proche.

2.6.1.2.2 Coût

Les facteurs liés aux coûts deviennent très influents dans la gestion des services Fog, du point de vue du fournisseur de services ou des utilisateurs. On peut identifier deux principaux types de coûts :

- **Coût de réseau** : lié aux frais de transmission de données et aux dépenses associées.
- **Coût d'exécution** : lié aux dépenses de calcul des nœuds Fog.

D'autres dépenses peuvent également être identifiées : coûts liés au stockage, au déploiement, aux mesures de sécurité, à la migration, etc.

Dans **[88]**, les auteurs proposent de minimiser le coût total qui comprend le coût de traitement et de stockage dans le Cloud et le Fog, le coût de communication entre le Fog et le Cloud et entre les nœuds Fog, et le coût de communication du déploiement de service depuis le contrôleur de service Fog vers les FN.

2.6.1.2.3 Consommation d'énergie

L'efficacité énergétique est l'une des principales préoccupations des systèmes IoT et un critère de performance important que plusieurs travaux tentent d'étudier dans le contexte du Fog Computing. La consommation d'énergie englobe principalement deux aspects :

- **Le type de service à traiter** : la consommation d'énergie dépend du type de service exécuté.
- **La consommation d'énergie au niveau du service** : elle-même composée de trois éléments principaux :

  - L'énergie consommée lors de l'envoi du service par l'utilisateur final vers le nœud Fog.
  - L'énergie consommée lors du traitement du service par le nœud Fog.
  - L'énergie consommée lorsque le Fog a besoin d'interagir avec le Cloud.

Par exemple, les travaux de Sarkar et al. **[70, 71]** et de Nishio et al. **[67]** étudient la question de la consommation d'énergie dans l'environnement Fog en tenant compte de la consommation d'énergie à la fois au niveau du réseau et du dispositif.

2.6.1.2.4 Autres métriques

D'autres métriques peuvent être prises en compte lors de la résolution du problème de placement de service. Certaines de ces métriques sont :

- **Qualité d'expérience (QoE)** : Considérée comme une mesure centrée sur l'utilisateur, la QoE englobe les exigences, les perceptions et les intentions de l'utilisateur lors du déploiement de services **[61]**. À titre d'exemple, on peut citer le travail de Mahmud et al. **[63]** qui propose une stratégie de placement d'applications basée sur la QoE, où le déploiement tient compte en priorité des attentes des utilisateurs.
- **Taux de congestion** : Concernant le taux de congestion, Yu et al**. [89]** proposent de considérer le rapport minimum entre le flux et la capacité du lien pour aborder le placement de service et le routage des données des applications de traitement en temps réel dans l'IoT.

- **Taux de blocage** : la probabilité qu'une requête soit rejetée en raison de ressources insuffisantes.
- **Requêtes échouées** : le nombre de requêtes qui n'ont pas pu être traitées avec succès.

2.7 Stratégies de résolution

Plusieurs problèmes compliquent le calcul d'un placement efficace des services dans un tel contexte. Premièrement, la nature hétérogène et les capacités limitées de la plupart des nœuds Fog (ressources limitées). Deuxièmement, le caractère dynamique de l'environnement : les ressources peuvent apparaître et disparaître, d'autres se déplacer, les informations sur l'infrastructure et les applications peuvent changer au fil du temps (par exemple, la variation de la charge de travail). Troisièmement, la distribution géographique des dispositifs Fog sur une infrastructure à grande échelle.

Ces spécificités et contraintes font du problème de placement de service (SPP) dans un environnement Fog une tâche difficile. Dans la littérature, on identifie quatre approches principales utilisées pour résoudre ce problème : la résolution exacte, les stratégies d'approximation, les politiques heuristiques ou méta-heuristiques.

2.7.1 Approches basées sur la résolution exacte

Une solution exacte se calcule souvent à l'aide d'un solveur à programmation linéaire en nombres entiers (Integer Linear Programming ILP) ou en effectuant une recherche exhaustive (en énumérant toutes les solutions). Parmi les travaux qui ont tenté de résoudre le SPP de manière exacte, on cite les travaux **[66, 69, 75, 79, 81]** qui utilisent une formulation ILP et un solveur d'optimisation exact pour définir une solution optimale, et **[91]** qui utilise une recherche exhaustive de placement.

Cependant, il est important de noter que la réalisation d'une résolution exacte nécessite un temps de traitement important avant d'atteindre les solutions optimales et ne peut être utilisée que pour des problèmes de petite taille. En effet, trouver une solution exacte peut être extrêmement long et ne convient pas aux problèmes de grande envergure comme les environnements Fog. Ainsi, les travaux de la communauté de recherche se concentrent principalement sur la fourniture d'approximations efficaces, d'approches heuristiques ou méta-heuristiques, où des solutions sous-optimales peuvent être calculées en peu de temps.

2.7.2 Approches basées sur les stratégies d'approximation

Les techniques d'approximation permettent de calculer des solutions avec des garanties prouvables quant à leur distance par rapport à la solution optimale. Les approximations sont des algorithmes efficaces qui permettent de calculer des solutions sous-optimales pour les problèmes d'optimisation NP-complets. Par exemple, dans **[89]**, les auteurs proposent d'utiliser un algorithme d'approximation entièrement polynomial en temps pour résoudre le problème de provisionnement d'applications IoT. Cette approche permet de calculer une solution sous-optimale et des limites pour le SPP dans un temps relativement court.

2.7.2.1 Approches basées sur les heuristiques

En raison de la taille, du caractère dynamique et mobile des infrastructures Fog, qui rendent l'analyse exacte quasiment inapplicable, les heuristiques sont souvent étudiées. Conçue pour obtenir une solution dans un délai raisonnable, une heuristique est un ensemble de règles et de méthodes visant à trouver une solution réalisable pour un problème donné. Cependant, avec les solutions basées sur des heuristiques, aucune garantie de performance n'est fournie.

Comme approches heuristiques, on peut trouver par exemple les heuristiques "fail-first/fail-last" utilisées par Brogi et al. dans **[50, 51]**, ils proposent d'adopter ces deux stratégies pour déterminer en un temps relativement court un déploiement réalisable pour une application dans le Cloud de Brouillard (FC). L'heuristique "fail-first" est utilisée dans ce cas pour sélectionner le composant non déployé qui à le moins de nœuds compatibles. La politique "fail-last" trie les nœuds candidats par ordre décroissant du nombre de terminaux requis par un composant logiciel et de leurs capacités matérielles. Pour garantir la satisfaction de toutes les requêtes, ces heuristiques calculent la meilleure ressource en termes de proximité spatiale et les appareils les plus puissants qui peuvent les prendre en charge.

2.7.2.2 Approches basées sur les méta-heuristiques

Inspirées par la nature, les solutions méta-heuristiques visent à fournir la meilleure solution en améliorant de manière itérative la qualité du résultat et en aidant le processus de recherche à sortir des optima locaux dans un délai raisonnable (contrairement aux heuristiques qui peuvent rester bloquées dans un optimum local). Plusieurs algorithmes méta-heuristiques

sont proposés dans la littérature, comme les algorithmes génétiques **[57]**, l'optimisation par colonies de fourmis **[52]**, l'optimisation par essaim de particules **[60]** ou la recherche par tabou **[54]**. Ces algorithmes s'appuient sur l'évolution d'une population où, à chaque itération, la population (solution) la mieux fondée est conservée pour l'itération suivante, afin de définir au final la meilleure solution par rapport à un objectif (métrique) donné.

Nous citons le travail de Skarlat et al. **[75]** qui utilise un algorithme génétique (GA) pour résoudre le SPP dans le Fog Computing. Un GA **[84]** est un algorithme évolutionnaire qui imite le processus d'évolution naturelle des chromosomes. Dans leur travail, les auteurs supposent que chaque gène d'un chromosome représente une décision de placement de service. Le placement est amélioré de manière itérative en fonction d'une fonction d'adaptation basée sur le principe d'encouragement.

2.7.3 Méthodes d'allocation basées sur la priorité et sur le tri

Les auteurs de **[39]** ont récemment proposé des techniques d'allocation de services basée sur la priorité et d'allocation de services basée sur le tri, qui sont utilisés pour déterminer l'ordre optimal d'utilisation des appareils pour exécuter différents services. Les résultats expérimentaux ont démontré que la technique proposée réduit la communication de données sur le réseau de 88 %, ce qui est obtenu en allouant la plupart des services localement dans le Fog. Ils ont augmenté la distribution des services aux appareils du Fog de 96 %, tout en minimisant simultanément le gaspillage des ressources du Fog.

2.7.3.1 Allocation de services en fonction des priorités

Pour sélectionner le processus d'allocation de services et décider si les services doivent être gérés dans le Fog ou dans le Cloud. Nous avons besoin des exigences des services (serReq) et des capacités des appareils (devCap) en tant qu'entrée. Ensuite, les services sont alloués aux appareils de Fog en utilisant la méthode Fog Device (devCap, serReq). Ensuite, l'un des algorithmes présentés dans **[39]** (BestFit,WorstFit,FirstFit) sera exécuté. Ensuite, les capacités de Fog Computing seront évaluées afin d'allouer les services. Si aucun appareil de Fog n'est disponible pour gérer le service, nous l'assignons aux appareils Cloud en appelant Cloud Device (remainingSer) et en transférant le reste des services au Cloud **[39]**.

2.7.3.2 Allocation de services en fonction de tri

Dans **[39]** le tri sélectif à double pivot est utilisé, un algorithme de tri efficace couramment utilisé en informatique et dans les applications de traitement des données, en particulier pour le tri des types de données primitifs tels qu'entier, double et réel. Selon les conclusions des auteurs Laroui et al. **[40]**, le tri sélectif à double pivot est généralement plus rapideet plus efficace que les autres algorithmes de tri sélectif, en particulier sur les grands ensembles de données. Ils soulignent que le tri sélectif à double pivot fonctionne efficacement avec des données ordonnées de manière aléatoire et partiellement triées, et qu'il comporte un nombre minimal de comparaisons et de permutations.

Le tri peut être une technique précieuse dans le contexte de l'informatique en brouillard pour optimiser l'allocation des services et réduire le niveau de complexité de ladistribution des services pour les dispositifs en brouillard. Le tri des données avant qu'elles n'atteignent les dispositifs de brouillard facilite le déploiement des ressources et maximise les performances globales du réseau. Les besoins techniques des services classés par ordre croissant, du plus faible au plus élevé, afin de faciliter les techniques d'attribution des services pour les dispositifs de Fog. Également les capacités des dispositifs de Fog sont triées. Cela permet une allocation de services plus rapide et plus efficace **[39]**.

Des techniques d'optimisation telles que la théorie des jeux, le bin packing, la programmation linéaire et la planification peuvent être également utilisées pour modéliser et résoudre les problèmes d'allocation de services dans des environnements Fog **[39]**.

2.7.4 Bin packing

Le bin packing est un problème d'optimisation combinatoire qui consiste à emballer un ensemble d'articles dans un nombre fixe de bacs, dans le but de minimiser le nombre de bacs utilisés ou le coût global de la solution. Dans les environnements de Fog Computing et d'IoT, le bin packing peut être utilisé pour optimiser le placement des services et des appareils, en tenant compte de facteurs tels que les conditions du réseau, les exigences deservice et les caractéristiques des

appareils, afin de minimiser le coût global de la solution en réduisant le nombre de nœuds de Fog utilisés.

Il existe de nombreuses variantes du problème du bin packing, notamment le problème dubin packing multidimensionnel et le problème du bin packing multi-contraintes. Ces variantes peuvent être utilisées pour ajouter des contraintes et des exigences supplémentaires dans le cadre de l'informatique en nuage et de l'IoT environnements. Par exemple, le problème de l'emballage multidimensionnel peut être utilisé pour prendre en compte les différents besoins en ressources des services et des appareils, et le problème de l'emballage multi-contraintes peut être utilisé pour prendre en compte des contraintes supplémentaires telles que la sécurité et la protection de la vie privée **[39]**.

2.8 Conclusion

Dans ce chapitre, nous avons présenté dans un premier temps des travaux d'optimisation du placement des instances de services d'IoT dans les infrastructures de Fog. Ces travaux se concentrent principalement sur la réduction de la latence du système, mais cela peut être accompagné par d'autres objectifs comme la réduction de la consommation énergétique et le coût opérationnel ou encore l'amélioration de la QoS.

Après avoir présenté le contexte et les travaux de la littérature liés à notre étude, nous présentons dans le chapitre suivant notre contribution.

3 Chapitre 3 : Allocation de services basée sur Ant-Lions Optimizer (ALO)

3.1 Introduction

Dans ce chapitre, nous décrivons la métaheuristique utilisée dans notre mémoire pour résoudre le problème d'allocation de service dans le Fog Computing. Nous commençons par présenter la modélisation mathématique du problème de placement de services sur les nœuds Fog.

Ensuite, nous présentons l'adaptation de cet algorithme au problème d'allocation. Par la suite, Afin d'évaluer les performances d'ALO, nous avons effectué une comparaison avec la métaheuristique Particle Swarm Optimization (PSO).

Enfin, nous terminons le chapitre par une conclusion qui résume l'essentiel de notre contribution.

3.2 La modélisation mathématique de problème de placement de services dans les nœuds Fog

Dans un serveur de Fog, chaque hôte dispose de ressources informatiques variables telles que la mémoire principale, la bande passante, le stockage secondaire et un certain nombre de cœurs de processeur. Il y a p nombre de machines virtuelles qui sont exécutées sur q nombre d'hôtes. Supposons qu'il y ait n services et que chaque service soit considéré comme une charge de travail avec des exigences en ressources telles que l'unité centrale, la bande passante et la mémoire principale. Ces demandes de service peuvent être placées sur les machines virtuelles sur la base d'un à plusieurs **[93]**.

Dans ce mémoire, nous générons la matrice qui donne le temps de calcul attendu (ETC Expected Time to Compile / Complete) défini comme le temps de calcul pour les services spécifiques sur la VM. Dans la matrice ETC, les lignes de la matrice indiquent les services et les colonnes de la matrice indiquent les machines virtuelles. La matrice contient le temps d'exécution du service sur ces VM, nous avons considéré un nombre m de machines virtuelles et un nombre n de services. La valeur de i variera donc de 1 à n et celle de j de 1 à m **[93]**.

Les valeurs de la matrice ETC sont calculées comme le rapport entre la tâche et la vitesse de la VM, comme indiqué dans l'Equation suivante :

$$ETC_{ij} = \frac{la\ durée\ des\ tâches}{V_{MIPS}} \quad (3.1)$$

Les services seront ensuite attribués aux machines virtuelles à l'aide de la méthode d'attribution de services proposée. L'attribution des services aux machines virtuelles est représentée par la matrice D_{ij}. La valeur de D_{ij} sera de 1 si la tache *i* attribuée à la machine virtuelle *j,* sinon la valeur de D_{ij} est de 0, l'équation 3.2 donne la matrice d'allocation de dix services aux trois machines virtuelles.

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad (3.2)$$

Dans notre simulation, nous prenons en compte le temps de communication et l'utilisation des ressources du réseau pour calculer le coût du service. Le temps de transmission du service dans l'environnement Fog Computing est calculé à l'aide de l'équation 3.3. Ainsi, le temps de transmission est défini comme le rapport entre la taille des données et la largeur de bande (BW) pour chaque service sur chaque VM.

$$TT_{ij} = \frac{la\ \ durée\ de\ tâches}{BW} \quad (3.3)$$

Le temps de transmission est calculé pour chacun des services sur chaque VM. Nous considérons ici la latence globale du service, qui est définie comme la somme du temps de traitement et du temps de communication. Le temps de latence pour le service complet est déterminé par $2 * TT_{ij}$

La somme totale des temps d'exécution (ET_j) pour tous les services assignés à la VM_j est calculée à l'aide de l'équation 3.4.

$$ET_j = \sum_{i=1}^{n} D_{ij} \times ETC_{ij} \quad (3.4)$$

E_{vj} est la consommation d'énergie globale d'une VM_j est définie par l'équation (3.5) :

$$Ev_j = ET_j \times b_j + (M - ETj) \times a_j \times MIPS_j \quad (3.5)$$

En outre, la valeur de $b_j = 10^{-8} * (MIPS_j)^2$ et a_j = 0,6 *b_j Joules/sec sont considérées pour notre simulation.

La consommation totale d'énergie dans l'ensemble du système est définie par l'équation (3.6).

$$E = \sum_{j=1}^{m} Ev_j \quad (3.6)$$

La somme des durées de transmission (TV_j) pour tous les services qui ont été utilisés dans le système est définit par l'équation (3.7).

$$Tv_j = \sum_{j=1}^{m} D_{ij} \times TT_{ij} \quad (3.7)$$

Le temps de transmission total des VM est calculé à l'aide de l'équation (3.8)

$$TL = \sum_{j=1}^{m} TV_j \quad (3.8)$$

La valeur de la fonction objectif sera calculée en ajoutant le Makespan, la consommation totale d'énergie du système et le temps de transmission total comme indiqué dans l'équation (3.9) avec les facteurs de pondération *p*, *q* et *r* dont la valeur est égale à 0.33.

$$minimize \quad F = M \times p + E \times q + TL \times r \quad (3.9)$$

3.3 Motivation

Ces dernières années, la plupart des métaheuristiques présentées dans la littérature pour résoudre des problèmes combinatoires sont inspirées des phénomènes naturels comme l'intelligence des animaux ou l'intelligence collective observée notamment chez les insectes sociaux.

ALO est capable de fournir des résultats très compétitifs en termes d'amélioration de l'exploration, d'évitement des optima locaux, d'exploitation et de convergence. L'algorithme ALO trouve également des conceptions optimales supérieures pour la majorité des problèmes d'ingénierie classiques utilisés (conception de fermes à trois barres, conception de poutres en porte-à-faux et conception de trains d'engrenages), ce qui montre que cet algorithme a des mérites dans la résolution de problèmes contraints avec des espaces de recherche diversifiés.

3.4 Contribution

Dans cette partie, nous présentons une adaptation de l'algorithme ALO nommé SA-DALO (Services Allocation based on Discret Ant Lion Optimizer) pour résoudre le problème d'allocation de services aux nœuds Fog en transformant le problème en un problème d'optimisation avec une fonction objectif.

Nous allons optimiser à la fois le Makespan, la consommation d'énergie et le temps de transmission.

Nous avons modifié l'algorithme ALO, habituellement utilisé pour résoudre des problèmes continus, afin de l'appliquer à notre problème qui est de nature discrète. L'ALO est efficace pour explorer de vastes espaces de recherche continus, mais notre problème nécessite des choix spécifiques et limités pour chaque variable. Pour ce faire, nous avons adapté l'algorithme en remplaçant les mouvements continus par des pas discrets, conformes aux contraintes de notre problème en utilisant la fonction Matlab « *round* ».

Algorithme 3.1 : Allocation de services basée sur Ant Lion Optimizer

Services allocation based on descret Ant Lion Optimizer (SA-DALO)
Entrés : les tailles de services, MIPS des VMs, les bandes passantes des VMs , taille de la population ,nombre des itérations (Max-iter) **Sortie :** l'élite (la matrice d'allocation) Initialiser la première population de fourmis et d'antlions de manière aléatoire Calculer le fitness des fourmis et des antlions Trouver la meilleure fourmi et la considérer comme l'élite (déterminé optimale) **Tant que** i ≤ Max-iter **Pour** chaque fourmi Sélectionner un fourmilion à l'aide de la roulette Mettre à jour c et d à l'aide des équations (1.8) et (1.9) Créer une marche aléatoire et la normaliser en utilisant les équations (1.3) et (1.5) Mettre à jour la position de la fourmi en utilisant (1.11) **fin pour** Calculer le fitness de toutes les fourmis à l'aide de l'équation (3.9) Remplacer un fourmilion par la fourmi correspondante si elle devient plus apte (Equation (1.10)) Mettre à jour l'élite si un antlion devient plus apte que l'élite i=i+1 **fin tant que** Retourner l'élite (la matrice d'allocation)

Les paramètres utilisés dans cette étude sont représentés dans le tableau 3.1

Taille de la population	50
Nombre des itérations	100
Nombre de VMs	3
Nombre de services	[10,40]
Les tailles de services (MI)	[1000,20000]
VMs MIPS	[500,2000]
BW	[1000,3000]
Coefficient d'inertie *w*	1
Coefficients de correction *c1, c2*	2

Tableau 3.1 : Paramètres de simulation

3.5 Tests et simulation

Dans le tableau 3.2, nous présentons les résultats de la simulation pour plusieurs indicateurs clés : le Makespan, l'énergie consommée, le temps de transmission et la fonction de fitness, en fonction du nombre de services. Pour cette simulation, nous avons utilisé l'algorithme Ant Lion Optimizer (ALO), que nous avons adapté pour optimiser ces paramètres dans le cadre de notre problème.

Nous avons également adapté l'algorithme S-PSO présenté dans le chapitre 1 nommé SA-DSPSO (Services Allocation based on Descret Standard Particle Swarms Optimization) afin d'évaluer la performance de la métaheuristique SA-DALO.

Les essaies ont été effectués sur une machine doté d'un processeur Intel(R) Core(TM) i5-6600U CPU avec une fréquence de 2,60 Ghz / 2,80 Ghz et une RAM de 8Go , exécutant Matlab R2021a sous système Windows 10 professionnel 64 bits .

Algorithmes / No services	SA-DALO				SA-DSPSO			
	M	E	TL	F	M	E	TL	F
10	7,9	18,2	11,7	14,2	8,1	33	13	24,7
20	30,2	20,9	44,7	44,7	30,6	118,2	48	61,7
30	58,7	64	89,6	83,4	59,5	129,7	98	117,3
40	67,2	87	105,2	91,9	67,6	212,5	112,6	126,4

Tableau 3.2 : Résultats d'exécution de SA_DALO et SA-DSPSO avec les différents scénarios

3.6 Etude comparative avec la métaheuristique SA-DSPSO

Sur la base des mesures présentées dans le tableau 3.2, nous comparons les résultats obtenus par l'algorithme SA-DALO et ceux obtenus par l'algorithme d'optimisation SA-DSPSO

Nous présentons quatre histogrammes illustrant les principaux résultats obtenus. Chaque histogramme représente des données essentielles liées à un de nos paramètres de performance tels que le Makespan, l'énergie consommée, le temps de transmission et la fonction de fitness, en fonction de différentes configurations de services et de méthodes d'optimisation. L'objectif est de comparer et d'analyser l'efficacité de l'algorithme SA-DALO avec d'autres méthodes, en examinant comment ces paramètres varient sous différentes conditions expérimentales.

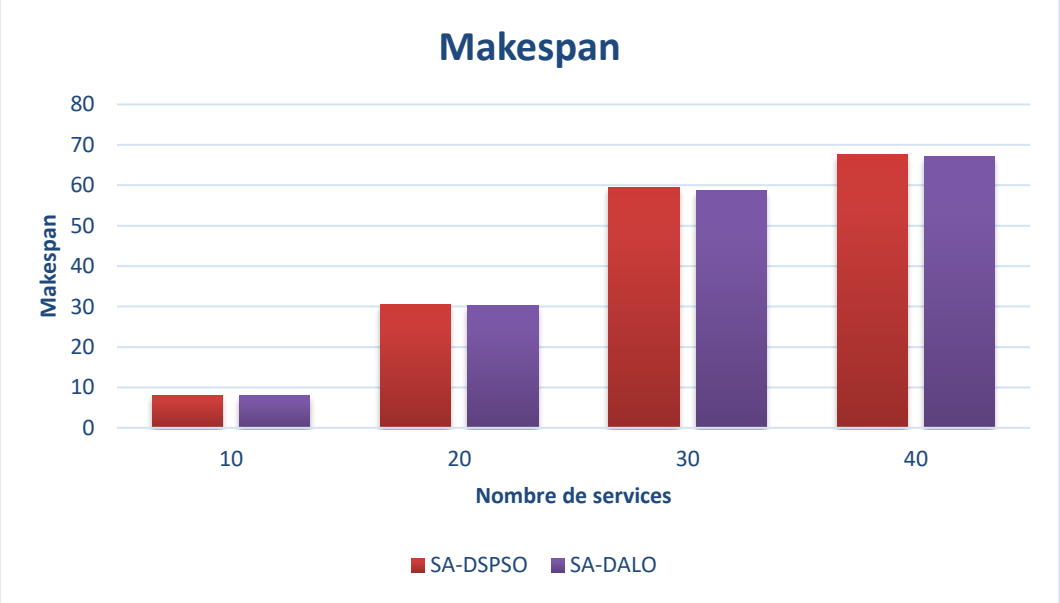


Figure 3.1 : Résultats de Makespan selon les deux Algorithmes SA-DALO et SA-DSPSO avec différents nombres de services

Analyse 1 : À partir des résultats obtenus, nous remarquons que les valeurs du Makespan obtenues avec SA-DSPSO sont 2 % plus grandes que celles obtenues par SA-DALO

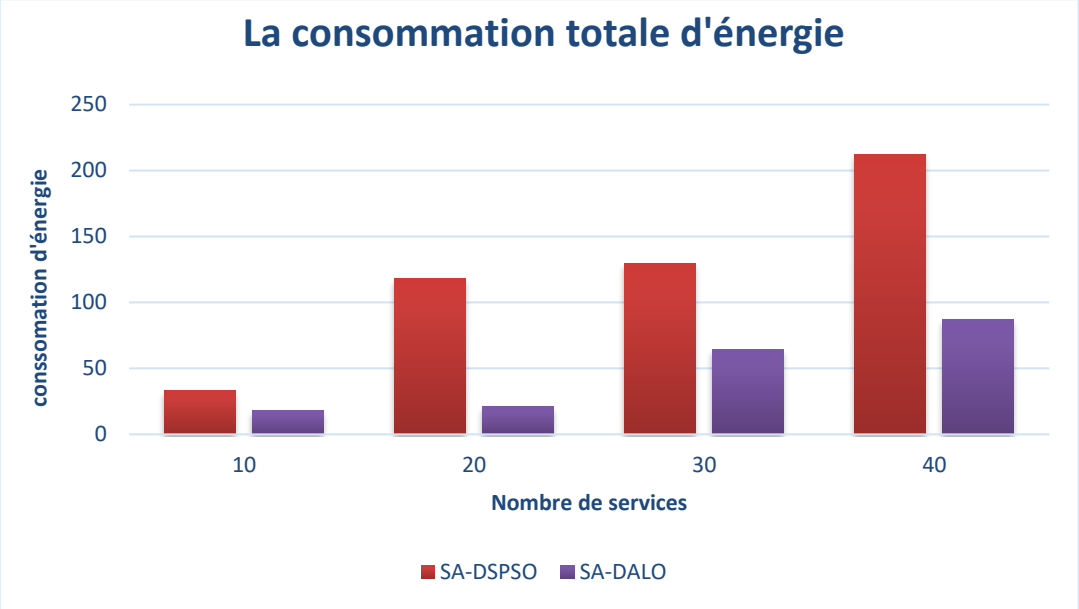


Figure 3.2 : Valeurs d'énergie totale consommée selon les deux Algorithmes SA-DALO et SA-DSPSO avec différents nombres de services

Analyse 2 : nous observons, d'après les résultats, que les valeurs de l'énergie totale consommée issues de SA-DALO sont 59% plus faibles que celles obtenues par le SA-DSPSO.

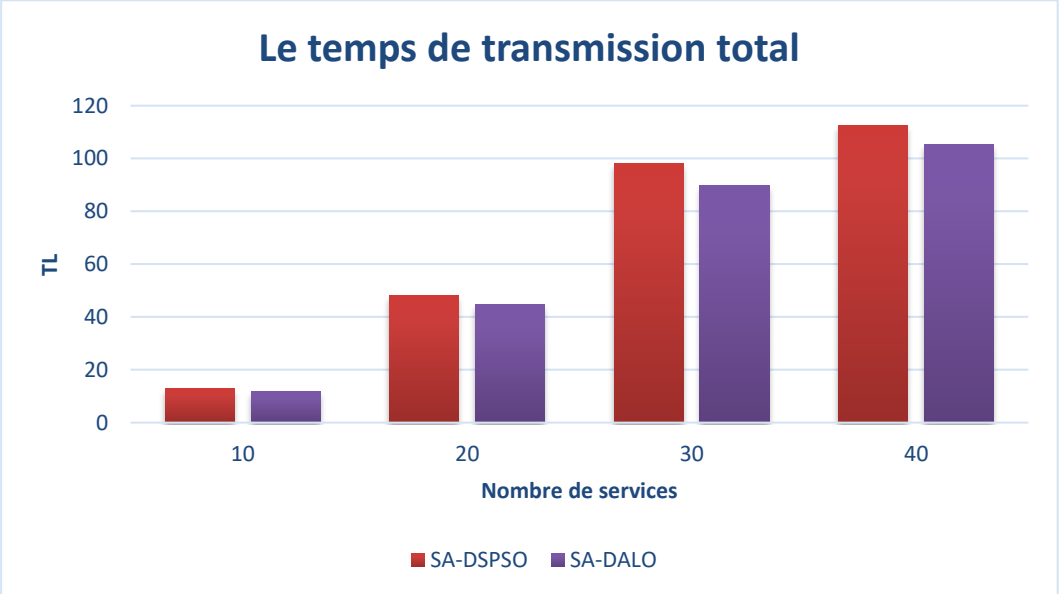


Figure 3.3 : les valeurs de temps total de transmission selon les deux Algorithmes SA-DALO et SA-DSPSO avec différents nombres de services

Analyse 3 : Les résultats révèlent que le temps total de transmission est de 8 % plus court avec l'utilisation de SA-DALO comparé au SA-DSPSO.

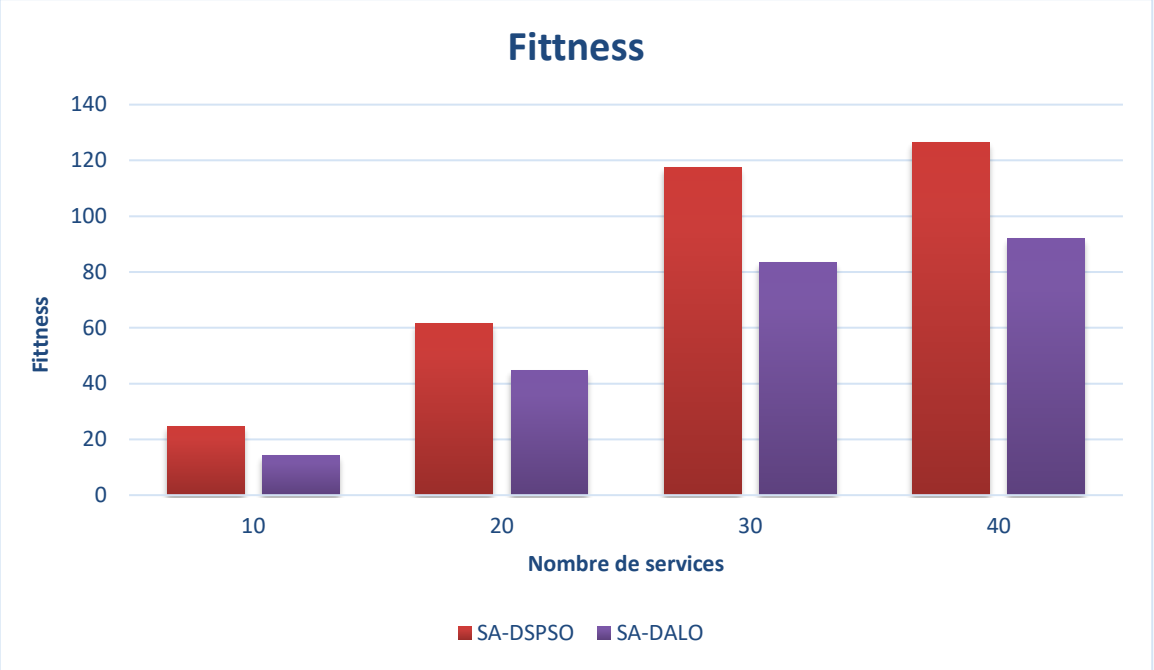


Figure 3.4 : les résultats de la fittness selon les deux Algorithmes SA-DALO et SA-DSPSO avec différents nombres de services

Analyse 4 : Comme nous pouvons le voir dans la figure 3.4 ci-dessus, en variant le nombre de services, les valeurs de fitness obtenues par SA-DALO sont 31 % plus faibles que celles obtenues par SA-DSPSO.

3.7 Synthèse

Pour évaluer la performance de l'algorithme SA-DALO, nous avons comparé ses résultats avec ceux de l'algorithme SA-DSPSO, une méthode évolutive dans le domaine des algorithmes inspirés par la nature. Nous avons constaté que la différence de makespan entre SA-DALO et SA-DSPSO était minime, à seulement 2%. En revanche, l'économie d'énergie réalisée par SA-DALO était significativement meilleure, atteignant 59%. De plus, bien que la différence pour le temps de transmission total soit de seulement 8%, SA-DALO a surpassé SA-DSPSO de 31% en termes de fitness. Ces résultats démontrent clairement que l'approche ALO proposée surpasse l'utilisation de S-PSO pour minimiser la fonction objectif, en particulier pour la résolution des problèmes d'allocation de services dans le Fog Computing.

3.8 Conclusion

Dans ce chapitre, nous avons abordé le problème de l'allocation de services dans un environement de Fog Computing. Nous avons modélisé ce problème en définissant les contraintes et les objectifs spécifiques, offrant ainsi une base solide pour l'application des techniques de résolution.

Pour résoudre ce problème, nous avons proposé l'utilisation de la métaheuristique Ant Lion Optimizer (ALO). Nous avons montré comment cette méthode peut être efficacement appliquée à notre problème.

Afin d'évaluer les performances d'ALO, nous avons effectué une comparaison avec la métaheuristique PSO. Nos expérimentations ont révélé que, dans la majorité des cas, les valeurs obtenues avec SA-DALO surpassent celles obtenues avec SA-DSPSO en termes de qualité de solution et de convergence. SA-DALO a démontré une meilleure capacité à échapper aux minima locaux et à exploiter efficacement l'espace de recherche, ce qui s'est traduit par des solutions optimisées et robustes.

La métaheuristique ALO s'est avérée être une méthode prometteuse et performante pour résoudre le problème d'allocation de services. Cette étude ouvre la voie à des recherches futures visant à affiner davantage ALO et à explorer son application dans d'autres domaines d'optimisation complexes.

Conclusion générale

En conclusion, ce mémoire a exploré en profondeur l'allocation de services dans un environnement de Fog Computing. Nous avons commencé par une présentation détaillée des concepts de base de l'Internet des objets (IoT), du Cloud Computing et du Fog Computing, soulignant les avantages et les défis de ces technologies émergentes.

L'étude des approches existantes pour l'allocation des services dans le Fog Computing a révélé la complexité de gérer efficacement les ressources distribuées tout en répondant aux exigences de performance, telles que la latence et la qualité de service (QoS). Pour répondre à ces défis, nous avons proposé une méthode d'optimisation innovante basée sur la métaheuristique Ant Lion Optimizer (ALO).

Afin de tester la performance de notre algorithme, nous avons effectué une série de tests de simulation, une étude comparative a été réalisée avec l'algorithme SA-DALO et l'algorithme SA-DSPSO pour une étude de performance.

Les résultats expérimentaux de notre approche montrent des améliorations significatives par rapport aux méthodes traditionnelles, en optimisant l'allocation de services et en réduisant la consommation d'énergie, le Makespan et le temps de transmission, ce qui confirme l'efficacité de l'SA-DALO dans ce contexte. Cette étude ouvre la voie à de futures recherches et développements dans le domaine du Fog Computing, en proposant des pistes pour améliorer encore davantage l'allocation des services et l'efficacité des systèmes IoT.

En somme, ce mémoire contribue à la compréhension et à l'amélioration des mécanismes d'allocation de services dans le Fog Computing, offrant ainsi des solutions pratiques et optimisées pour les défis posés par l'IoT et les environnements de calcul distribués.

4 Bibliographie

[1] IEEE INTERNET OF THINGS JOURNAL, VOL. X, NO. X, Fog/Edge Computing-based IoT (FECIoT), Architecture, Applications, and Research Issues ,SEPTEMBER 2018, DOI: 10.1109/JIOT.2018.2875544.

[2] K. Dolui and S. K. Datta, "Comparison of edge computing implementations:Fog computing, cloudlet and mobile edge computing," IEEE Globalinternet of Things Summit (GIoTS), Geneva, 2017, pp. 1-6.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016.

[4] M. Aazam, and E. N. Huh, "Fog computing: The cloud-IOT/IoEmiddleware paradigm," IEEE Potentials, vol. 35, no. 3, pp. 40–44, May 2016.

[5] Smart eye. https ://iatranshumanisme.com/2018/01/23/smart-farmsvers- nouvelle-forme-dagriculture/. Visitée : 2024-03-17.

[6] Google glass. https ://www.futura-sciences.com/tech/definitions/smartphonegoogle-glass-15803/. Visitée : 2024-03-14.

[7] Drone de santé. https ://club-digital-sante.info/2016/11/drones-santesecours.
Visitée : 2024-03-18.

[8] Skydrone. https ://www.skydrone.fr/transport-objet-par-drone. Visitée :2024-03-20.

[9] Waze. https ://www.waze.com/fr, . Visitée : 2024-03-18.

[10] Waze stats. https ://searchengineland.com/waze-launches-local-adsprimarily-aimed-at-smbs-and-franchises-295285, . Visitée : 2024-03-18.

[11] National Institute of Standards and Technology Special Publication 800-145
7 pages (September 2011)

[12] H. Atlam, R. Walters, and G. Wills. Fog computing and the internet of things: A review. Big Data and Cognitive Computing, 2(2), 2018. ISSN 2504-2289. doi: 10.3390/bdcc2020010. URL http://www.mdpi.com/2504-2289/2/2/10.

[13] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran. EmuFog : Extensible and scalable emulation of large-scale Fog computing infrastructures.2017.

[14] NIST Special Publication (SP) 500-325, Fog Computing Conceptual Model, March 2018

[15] QUBAHAN Academic journal,A Study of Moving from Cloud Computing to Fog Computing, Doi: 10.48161/issn.2709-8206

[16] M. J. Baucas and P. Spachos, "Using cloud and Fog computing for large scale iot-based urban sound classification," *Simulation Modelling Practice and Theory,* vol. 101, p. 102013, 2020.

[17] NIST Special Publication 800-191 (Draft), The NIST Definition of Fog Computing, March 2018.

[18] Guillaume Pierre. Le " Fog Computing " est l'avenir du cloud – en plus frugal et plus efficace. The Conversation France, 2021, pp.1-4. Hal-03336292

[19] Mohammed Islam NAAS, placement des données de l'internet des objets dans une infrastructure de Fog, thèse présentée et soutenue à l'université de Bretagne occidentale, le 19/02/2019,

[20] P. Hu, S. Dhelim, H. Ning, and T. Qiu. Survey on Fog Computing: architecture, key technologies, applications and open issues. Journal of Network and Computer Applications, 98 :27 – 42, 2017. ISSN 1084-8045. doi : https:// doi.org/10.1016/j.jnca.2017.09.002. URL http://www.sciencedirect.com/ science/article/pii/S1084804517302953

[21] T.H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun. Fog Computing : Focusing on mobile users at the edge. arXiv preprint arXiv :*1502.01815*,2015

[22] M. Armbrust et al. *A view of cloud computing. communications of the ACM*". 2010 ,Vol.53, N4, P 50-58 DOI 10.1145/1721654.1721672.

[23] Tanissia Djemai. Placement optimisé de services dans les architectures fog computing et internet of things sous contraintes d'énergie, de QoS et de mobilité. Réseaux et télécommunications [cs.NI]. Université Paul Sabatier - Toulouse III, 2021. Français. ffNNT : 2021TOU30019ff. fftel-03280438f

[24] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys Tutorials, 17(4) :2347–2376, Fourthquarter 2015. ISSN 1553-877X.

[25] M. Abu-Elkheir, M. Hayajneh, and N. Ali. Data management for the internet of things: Design primitives and solution. Sensors, 13(11) :15582–15612, 2013.

[26] Adila MEBREK. *Fog Computing* pour l'Internet des objets, thèse de doctorat, UNIVERSITE DE TECHNOLOGIE DE TROYES, 2020.

[27] Fabien Douchet. Optimisation énergétique de data centers par utilisation de liquides pour le refroidissement des baies informatiques. Energie électrique. Université de Bretagne Sud, 2015. Français. ffNNT : 2015LORIS386ff. fftel-01325363ff

[28] W. Yu *et al* ., « Une enquête sur l'informatique de pointe pour l'Internet des objets », dans *IEEE Access* , vol. 6, pp. 6900-6919, 2018, doi : 10.1109/ACCESS.2017.2778504.

[29] Big Data Cogn. Comput, Fog Computing and the Internet of Things: A Review,2018, 2, 10; doi:10.3390/bdcc2020010

[30] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. Comput. Netw., 54(15) :2787–2805, October 2010. ISSN 1389-1286.doi : 10.1016/j.comnet.2010.05.010.

[31] P**.** Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on Fog Computing:Architecture, Key Technologies, Applications and Open Issues," Journal of Network and Computer Applications, pp. 27-42, Nov. 2017

[32] C. H. Papadimitriou and K. Steiglitz. Combinatorial optimization: algorithms and complexity. Courier Corporation, 1998.

[33] AZIZOU Zahia. Découverte et localisation de services dans l'Internet des Objets, thèse de doctorat, Université Akli Mohand Oulhadj de Bouira, 2022.

[34] R.W. Floyd. Algorithm 97: shortest path. Communications of the ACM, 5(6) :345, 1962.

[35] Mohammed BENHAMMOUDA. Contribution à l'optimisation d'ordonnancement de workflows dans un environnement cloud, informatique, UNIVERSITE DJILLALI LIABES - SIDI BEL ABBES, 2021

[36] Saad M, Enam RN and Qureshi R (2024) Optimizing multi-objective task scheduling in Fog Computing with GA-PSO algorithm for big data application. *Front. Big Data* 7:1358486. doi: 10.3389/fdata.2024.1358486

[37] MIRJALILI, Seyedali. The ant lion optimizer. *Advances in engineering software*, 2015, vol. 83, p. 80-98.

[38] Puliafito, C.; Mingozzi, E.; Longo, F.; Puliafito, A.; Rana, O. Fog Computing for the internet of things: A survey. ACM Trans. Internet Technol. (TOIT) **2019**, 19, 1–41. [CrossRef]

[39] Alsemmeari, R.A.; Dahab, M.Y.; Alturki, B.; Alsulami, A.A.; Alsini, R. Towards an Effective Service Allocation in Fog Computing. Sensors **2023**, 23, 7327. https://doi.org/10.3390/s23177327

[40] Laroui, M.; Nour, B.; Moungla, H.; Cherif, M.A.; Afifi, H.; Guizani, M. Edge and fog computing for IoT: A survey on current research activities & future directions. Comput. Commun. **2021**, 180, 210–231.

[41] T. Zhang, J. Jin, X. Zheng and Y. Yang, "Rate-Adaptive Fog Service Platform for Heterogeneous IoT Applications," in *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 176-188, Jan. 2020, doi: 10.1109/JIOT.2019.2945328.

[42] E. E. Tsiropoulou, P. Vamvakas, and S. Papavassiliou, "Joint customized price and power control for energy-efficient multi-service wireless networks via s-modular theory," IEEE Transactions on Green Communications and Networking, vol. 1, no. 1, pp. 17–28, 2017.

[43] J. Jin, "Flow control and performance optimization for multi-service networks," Ph.D. dissertation, University of Melbourne, 2010.

[44] M. Abderrahim, M. Ouzzif, K. Guillouard, J. Fran_cois, A. Lebre, C. Prud'homme, and X. Lorca. 2019. "Efficient Resource Allocation for Multi-Tenant Monitoring of Edge Infrastructures". In PDP. 158{165. https://doi.org/10.1109/EMPDP.2019.8671621

[45] Swati Agarwal, Shashank Yadav, and Arun Yadav. "An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing". IJIEEB 8 (01 2016), 48-61, 2016. https://doi.org/10.5815/ijieeb.2016.01.06

[46] Y. Ai, M. Peng, and K. Zhang, "Edge Computing Technologies for Internet of Things:A Primer". Digital Communications and Networks (2017). http://www.sciencedirect,2017. com/science/article/pii/S2352864817301335

[47] H. Reza Arkian, A. Diyanat, and A. Pourkhalili. 2017. "MIST: Fog-Based Data Analytics Scheme with Cost-Efficient Resource Provisioning for IoT Crowdsensing Applications". Journal of Network and Computer Applications 82 (2017), 152{165. https://doi.org/ 10.1016/j.jnca.2017.01.012

[48] T. Bahreini and D. Grosu."Efficient Placement of Multi-component Applications in Edge Computing Systems". In ACM/IEEE Symposium on Edge Computing .ACM, New York, NY, USA, Article 5, 11 pages. 2017 https://doi.org/10.1145/3132211.3134454

[49] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar. 2017. "Mobility-Aware Application Scheduling in Fog Computing". IEEE Cloud Computing 4, 2 (March 2017), 26_35. https://doi.org/10.1109/MCC.2017.27

[50] A. Brogi and S. Forti. "QoS-Aware Deployment of IoT Applications Through the Fog". IEEE Internet of Things Journal 4, 5 (Oct 2017), 1185-1192.

[51] A. Brogi, S. Forti, and A. Ibrahim,"How to Best Deploy Your Fog Applications, 2017. Probably". In IEEE ICFEC. 105-114.

[52] M. Dorigo. "Optimization, Learning and Natural Algorithms". Ph.D. Dissertation,1992.

Politecnico di Milano.

[53] T. N. Gia, M. Jiang, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. 2015. "Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction". In IEEE CIT/IUCC/DASC/PICom. 356{363. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.51

[54] F. Glover. "Future Paths for Integer Programming and Links to Artificial Intelligence".Comput. Oper. Res. 13, 5 (May 1986), 533-549. 1986. https://doi.org/10.1016/0305-0548(86)90048-1

[55] H. Gupta, A. Vahid Dastjerdi, S.-K. Ghosh, and R. Buyya. 2016. "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments". CoRR abs/1606.02007 (2016).

[56] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. "Towards Wearable Cognitive Assistance". In MobiSys. ACM, New York, NY, USA, 68-81. 2014. https://doi.org/10.1145/2594368.2594383

[57] J.H. Holland. "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence". MIT Press, Cambridge, MA, USA,1992.

[58] H. J. Hong, P. H. Tsai, A. C. Cheng, M. Y. S. Uddin, N. Venkatasubramanian, and C. H. Hsu. 2017. "Supporting Internet-of-Things Analytics in a Fog Computing Platform". In IEEE CloudCom. 138-145. https://doi.org/10.1109/CloudCom.2017.45

[59] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwalder, and B. Koldehofe. "Mobile Fog: A Programming Model for Large-scale Applications on the Internet of Things".2013. In ACM MCC. ACM, New York, NY, USA, 15-20.

[60] J. Kennedy and R.C. Eberhart. "Particle Swarm Optimization". In IEEE International Conference on Neural Networks 1995. 1942-1948.

[61] K. U. R. Laghari, N. Crespi, B. Molina, and C. E. Palau. "QoE Aware Service Delivery in Distributed Environment" 2011. In IEEE AINA. 837{842. https://doi.org/10.1109/WAINA.2011.58

[62] G. Lee, W. Saad, and M. Bennis. "An Online Secretary Framework for Fog Network Formation with Minimal Latency" 2017. In IEEE ICC. 1-6.

[63] R. Mahmud, K. Ramamohanarao, and R. Buyya. "Quality of Experience (QoE)-Aware Placement of Applications in Fog Computing Environments ". ACM Transactions

on Internet Technology (2018)

[64] R. Mahmud, S.N. Srirama, K. Ramamohanarao, and R. Buyya. "Latency-aware Application Module Management for Fog Computing Environments". J. Parallel and Distrib. Comput. (2018)

[65] R.T. Marler and J.S. Arora. "Survey of Multi-Objective Optimization Methods for Engineering". Structural and Multidisciplinary Optimization 26, 6 (01 Apr 2004), 369-395.

[66] SQ. T. Minh, D. T. Nguyen, A. Van Le, H. D. Nguyen, and A. Truong. "Toward Service Placement on Fog Computing Landscape" 2017. In 4th NAFOSTED Conference on Information and Computer Science. 291-296.

[67] T. Nishio, R. Shinkuma, T. Takahashi, and N.B. Mandayam. "Service-Oriented Heterogeneous Resource Sharing for Optimizing Service Latency in Mobile Cloud" 2013. ACM, New York, NY, USA, 19-26.

[68] B. Ottenwalder, B. Koldehofe, K. Rothermel, and U. Ramachandran. "MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing".2013. In ACM
DEBS. ACM, New York, NY, USA, 183-194. https://doi.org/10.1145/2488222.2488265

[69] J. Santos, T. Wauters, B. Volckaert, and F. De Turck. "Resource Provisioning for IoT Application Services in Smart Cities".2017. In CNSM. 1-9. https://doi.org/10.23919/CNSM.2017.8255974

[70] S. Sarkar, S. Chatterjee, and S. Misra. "Assessment of the Suitability of Fog Computing in the Context of Internet of Things". IEEE Transactions on Cloud Computing 6,1 (Jan 2018), 46-59.

[71] S. Sarkar and S. Misra. "Theoretical Modelling of Fog Computing: A Green Computing Paradigm to Support IoT Applications". IET Networks 5, 2 (2016), 23{29.

[72] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. "The Case for VM-Based Cloudlets in Mobile Computing". IEEE Pervasive Computing 8, 4 (Oct 2009), 14-23. https://doi.org/10.1109/MPRV.2009.82

[73] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenw□alder. "Incremental Deployment and Migration of Geo-distributed Situation Awareness Applications in the Fog". 2016. In ACM DEBS. ACM, New York, NY, USA, 258-269.

[74] V. Scoca, A. Aral, I. Brandic, R. De Nicola, and R. Brundo Uriarte. "Scheduling Latency-Sensitive Applications in Edge Computing".2018. In CLOSER.

[75] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner. "Optimized IoT Service Placement in the Fog".2017. Service Oriented Computing and Applications 11, 4 (01 Dec 2017), 427-443.

[76] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar. "Towards QoS-Aware Fog Service Placement".2017. In ICFEC. IEEE Computer Society, 89-96.

[77] V. B. C. Souza, W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, G. Ren, and G. Tashakor. "Handling Service Allocation in Combined Fog-Cloud Scenarios".2016. In IEEE ICC. 1-5.

[78] M. Taneja and A. Davy. "Resource Aware Placement of IoT Application Modules in Fog-Cloud Computing Paradigm". 2017. In IFIP/IEEE IM. 1222-1228.

[79] R. I. Tinini, L. C. M. Reis, D. M. Batista, G. B. Figueiredo, M. Tornatore, and B. Mukherjee. "Optimal Placement of Virtualized BBU Processing in Hybrid Cloud-Fog RANover TWDM-PON". 2017. In GLOBECOM. 1-6.

[80] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K.K. Leung."Dynamic Service Migration and Workload Scheduling in Edge-Clouds". Performance Evaluation 91 (2015), 205-228. Special Issue: Performance 2015.

[81] K. Velasquez, D.P. Abreu, M. Curado, and E. Monteiro. "Service Placement for Latency Reduction in the Internet of Things". 2017. Annals of Telecommunications 72, 1 (01 Feb 2017), 105{115. https://doi.org/10.1007/s12243-016-0524-9

[82] J. Wang, J. Pan, and F. Esposito. "Elastic Urban Video Surveillance System Using Edge Computing".2017. In SmartIoT. ACM, New York, NY, USA, Article 7, 6 pages. http://doi.acm.org/10.1145/3132479.3132490

[83] P. Wang, S. Liu, F. Ye, and X. Chen. "A Fog-Based Architecture and Programming Model for IoT Applications in the Smart Grid". CoRR abs/1804.01239 (2018). arXiv:1804.01239 http://arxiv.org/abs/1804.01239

[84] D. Whitley. "A Genetic Algorithm Tutorial". Statistics and Computing 4, 2 (01 Jun 1994), 65{85. https://doi.org/10.1007/BF00175354

[85] Y. Xia, X. Etchevers, L. Letondeur, T. Coupaye, and F. Desprez. "Combining Hardware Nodes and Software Components Ordering-based Heuristics for Optimizing the Placement of Distributed IoT Applications in the Fog".2018. In SAC. ACM, New York, NY, USA, 751{760. https://doi.org/10.1145/3167132.3167215

[86] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue. "On Reducing IoT Service Delay via Fog O_oading". IEEE Internet of Things Journal 5, 2 (April 2018), 998{1010.

https://doi.org/10.1109/JIOT.2017.2788802

[87] A. Yousefpour, G. Ishigaki, and J. P. Jue. "Fog Computing: Towards Minimizing Delay in the Internet of Things". In IEEE EDGE. 17{24. https://doi.org/10.1109/IEEE.EDGE.2017.12

[88] A. Yousefpour, A. Patil, G. Ishigaki, J.P. Jue, I. Kim, X. Wang, H.C. Cankaya, Q. Zhang, and W. Xie. 2017. "QoS-aware Dynamic Fog Service Provisioning".

[89] R. Yu, G. Xue, and X. Zhang. 2018. "Application Provisioning in Fog Computing-enabled Internet-of-Things: A Network Perspective". In IEEE INFOCOM.

[90] J. K. Zao, T. T. Gan, C. K. You, S. J. R. M~A©ndez, C. E. Chung, Y. T. Wang, T. Mullen, and T. P. Jung. "Augmented Brain Computer Interaction Based on Fog Computing and Linked Data". In International Conference on Intelligent Environments. 2014. 374_377. https://doi.org/10.1109/IE.2014.54

[91] L. Zhao, J. Liu, Y. Shi, W. Sun, and H. Guo. "Optimal Placement of Virtual Machines in Mobile Edge Computing". In GLOBECOM. 2017. 1-6. https://doi.org/10.1109/GLOCOM.2017.8254084

[92] Farah Ait Salaht, Frédéric Desprez, Adrien Lebre. An overview of service placement problem in Fog and Edge Computing. [Research Report] RR-9295, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, LYON, France. 2019, pp.1-43.

[93] V. Yadav, BV Natesha et RMR Guddeti, « GA-PSO : Allocation de services dans un environnement de Fog Computing utilisant un algorithme hybride bio-inspiré », *TENCON 2019 - Conférence IEEE Région 10 2019 (TENCON)* , Kochi, Inde, 2019, pp. 1280 -1285, : 10.1109/TENCON.2019.8929234.

[94] Maria Zemzami. Variations sur PSO : approches parallèles, jeux de voisinages et applications. Complexité [cs.CC]. Normandie Université; Ecole nationale des sciences appliquées (Kénitra, Maroc), 2019.

[95] Maria Zemzami , Norelislam Elhami , Abderahman Makhloufi , Mhamed Itmi , Nabil Hmina. Application d'un modèle parallèle de la méthode PSO au problème de transport d'électricité, February 2017

[96] KENNEDY J., EBERHART R., "Particle Swarm Optimization," Proceedings of the IEEE International Joint Conference on Neural Networks, IEEE Press, vol. 8, no. 3, pp. 1943–1948. 1995.

[97] Shi, Y.; Eberhart, R.C. A Modified Particle Swarm Optimizer. In Proceedings of the IEEE International Conference on Evolutionary Computation, Anchorage, AK, USA, 4–9 May 1998; IEEE Press: Piscataway, NJ, USA, 1998; pp. 69–73.

[98] Younes Abbassi, Habib Benlahmer. Un aperçu sur la sécurité de l'internet des objets (IOT). Colloque sur les Objets et systèmes Connectés - COC'2021, IUT d'Aix-Marseille, Mar 2021, MARSEILLE, France.

Résumé

Dans ce mémoire, nous explorons l'allocation de services dans un environnement de Fog Computing, en abordant d'abord les concepts fondamentaux de l'Internet des objets (IoT) et du Cloud Computing, et en introduisant le Fog Computing comme une solution pour rapprocher les ressources de calcul des dispositifs IoT, réduisant ainsi la latence. Nous explorons ensuite les différents défis et approches pour l'allocation des services dans cet environnement distribué, en se focalisant sur les critères de performance comme la latence et la qualité de service (QoS). Enfin, nous proposons une méthode d'optimisation basée sur la métaheuristique Ant Lion Optimizer (ALO) en mode discret pour l'allocation des services dans le Fog Computing. Nos résultats expérimentaux montrent que l'approche SA-DALO offre des améliorations notables par rapport à d'autres techniques, optimisant l'utilisation des ressources et la performance globale du système.

Mots clés : Internet des objets, Cloud computing, Fog Computing, Ant lion optimizer, SA-DALO

ABSTRACT

In this dissertation, we explore service allocation in a Fog Computing environment, first addressing the fundamental concepts of the Internet of Things (IoT) and Cloud Computing, and introducing Fog Computing as a solution for bringing compute resources closer to IoT devices, thereby reducing latency. We then explore the different challenges and approaches for allocating services in this distributed environment, focusing on performance criteria such as latency and Quality of Service (QoS). Finally, we propose an optimization method based on the Ant Lion Optimizer (ALO) metaheuristic in discrete mode, describing its application to improve service allocation in fog computing. Our experimental results show that the SA-DALO approach offers significant improvements over other techniques, optimizing resource utilization and overall system performance.

Keywords: Internet of Things, Cloud computing, Fog Computing, Ant lion optimizer, SA-DALO