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' Électronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Télécommunications

Spécialité : Réseaux et télécommunications

Thème

Ordonnancement de tâches dans un environnement de Fog Computing

Préparé par :

Ouhammou Megdouda

Chebrouk Melissa

Dirigé par :

M^{me} Mammeri Karima

M^{me} Azizou Zahia

Examiné par :

M Diboune Abdel Hani

M Boualem Mohamed

Année universitaire : 2023/2024

Remerciements

Nous tenons à remercier « Dieu » tout puissant qui nous a armés de courage, de volonté et surtout de patience.

Nous exprimons notre sincère gratitude à toutes les personnes qui ont rendu ce mémoire possible par leurs aides et leurs contributions.

Nos premiers remerciements sont adressés tout d'abord à notre promotrice

Mme Mammeri et Co-promotrice Mme Azizou pour nous avoir proposé ce sujet et de l'avoir

dirigé. Leurs conseils et leurs encouragements nous ont été précieux, nous apprécions aussi leur regard critique sur ce travail.

Nous tenons également à remercier les membres de jury qui ont eu l'amabilité d'examiner et

de juger notre mémoire.

Toutes nos reconnaissances sont adressées à tous les enseignants qui nous ont suivi

infatigablement durant tout notre cursus universitaire.

Nous gardons une place toute particulière à nos familles, nous leurs exprimons toute notre

profonde reconnaissance car ils nous ont constamment aidé, par leur soutien moral et leurs

encouragements pour achever ce travail.

Dédicace

À mes très chers parents,

Véritables piliers de ma vie, sources inépuisables, d'amour et d'affection,

À mes chers frères,

Compagnons de toujours, sources constantes de joie et de bonheur,

À mes belles-sœurs,

Qui enrichissent notre famille par leur présence chaleureuse et leur amour sincère,

À mes chères cousines et mes copines,

Pour leur soutien indéfectible et leur amitié précieuse,

À toute ma famille,

Source d'espoir et de motivation,

À mon binôme,

Pour les efforts déployés avec assiduité et persévérance tout au long de ce projet,

À tous ceux qui ont cru en moi et m'ont accompagné dans cette aventure, cette réussite est Aussi la vôtre.

Melissa.Ch

Dédicace

À la mémoire de mon grand-père, dont l'héritage de persévérance et de sagesse continue de m'inspirer chaque jour,

À mes très chers parents, sources de vie, d'amour et d'affection,

À mes chers frères et ma chère sœur, sources de joie et de bonheur,

À ma chère belle-sœur, qui enrichit notre famille de sa gentillesse et de son soutien précieux,

À toute ma famille, source d'espoir et de motivation,

À mon binôme, pour les efforts déployés avec assiduité et persévérance tout au long de ce projet,

À tous ceux qui ont cru en moi et m'ont accompagné dans cette aventure, cette réussite est aussi la vôtre.

Megdouda.Oh

Table des matières

Liste des figures

Liste des tableaux

Liste des Abréviations

ACO : Ant Colony Optimization

ADGTS : Adaptive Double fitness Genetic Task Scheduling

ALO : Ant Lion Optimizer

ARPANET : Advanced Research Projects Agency Network

AWS : Amazon Web Services

CT : Completion Time

ECT : expected complete time

EDF: Earliest Deadline First

FCAS : Fog Computing Adaptive Scheduling Algorithm

FCFS : First Come First Served

FIFO : First In First Out

GA : Genetic Algorithm

IaaS : Infrastructure as a Service

IBM : International Business Machines

IMMPA : Improved Modified Marine Predators Algorithm

IOT : Internet Of Things

MCSO : Modified Cat Swarm Optimization

MIT : Massachusetts Institute of Technology

MIPS : Millions d'Instructions Par Seconde

MMPA : Modified Marine Predators Algorithm

MPA : Marine Predators Algorithm

MPSO : Modified Particle Swarm Optimization

NBIHA : Hybrid Algorithm Inspired by Bioinformatics

NIST : National Institute of Standards and Technology

PaaS : Platform as a Service

PSO : Particle Swarm Optimization

PWOA : Particle Whale Optimization Algorithm

RAM : Random Access Memory

RFID : Radio Frequency IDentification

SaaS : Software as a Service

TCaS : Time-Cost aware Scheduling

TS-ALO : Task Scheduling-ALO

TS-PSO : Task Scheduling-PSO

UIT : International Telecommunication Union

USB : Universal Serial Bus

WOA: Whale Optimization Algorithm

Introduction générale

L'Internet des Objets (IoT, Internet of Things) est un réseau qui permet à divers objets physiques et virtuels d'être connectés, facilitant ainsi l'échange de données et la communication entre eux [1].

Le Cloud Computing est une technologie permettant le stockage et le traitement des données générées par l'IoT. Il offre une gestion efficace des ressources informatiques et permet la fourniture de services à grande échelle, intégrant des capacités avancées de traitement et de gestion des données avec une accessibilité mondiale [1].

L'Internet des Objets et le Cloud Computing sont deux technologies distinctes mais essentielles, offrant de nombreux avantages pour la recherche scientifique et l'amélioration de la vie quotidienne. Les applications IoT, présentes dans divers domaines, génèrent d'énormes quantités de données, nécessitant souvent des solutions spécifiques de stockage et de traitement. L'intégration de l'IoT avec le Cloud Computing est une approche efficace pour simplifier le développement des applications IoT. Cependant, malgré les nombreux avantages de cette intégration, plusieurs défis persistent, ce qui nécessite la création de nouveaux mécanismes pour résoudre ces problèmes.

C'est pourquoi le Fog computing a été introduit comme une solution intermédiaire entre le Cloud et les dispositifs IoT. Dans l'architecture du Fog computing, les dispositifs Fog situés près des utilisateurs assurent le calcul et le stockage locaux. L'un des principaux défis est l'ordonnancement efficace de tâches dans cet environnement.

L'ordonnancement est un aspect fondamental dans le cadre du Fog computing. Son objectif est d'assigner les tâches aux ressources disponibles de manière à optimiser divers critères tels que la minimisation du temps d'exécution de tâches, temps de complétion et le makespan. Cependant, le problème d'ordonnancement reste complexe en raison de la diversité et de la distribution des ressources, ainsi que des contraintes variées à prendre en compte.

Dans ce travail, nous proposons une étude approfondie sur l'ordonnancement de tâches dans le Fog Computing. Notre étude se concentre sur l'utilisation de deux métaheuristiques avancée, l'algorithme ALO (Ant Lion Optimizer) et l'algorithme PSO (Particle Swarm Optimization), pour résoudre le problème de l'ordonnancement de tâches. Ces approches sont mises en œuvre pour optimiser l'affectation de tâches aux ressources disponibles.

Notre mémoire est structuré en trois chapitres, détaillés comme suit :

Chapitre I « IoT, Cloud Computing et Fog Computing » Ce chapitre introduit les concepts de base de l'Internet des Objets (IoT), du Cloud Computing et du Fog Computing.

Chapitre II « Ordonnancement de tâches dans le Fog Computing » Ce chapitre explore la notion d'ordonnancement ainsi que divers algorithmes classiques et ceux spécifiques à l'ordonnancement dans le Fog Computing.

Chapitre III « Proposition et évaluation de performances » Ce chapitre présente notre proposition pour le problème d'ordonnancement de tâches dans le Fog Computing.

Chapitre I : IoT, Cloud Computing et Fog Computing

1. Introduction

L'Internet des Objets (IoT) connaît une croissance rapide, permettant les échanges de données entre les objets qui sont connectés à internet. Cette expansion a donné la naissance au Cloud Computing, offrant aux entreprises la possibilité de réduire les coûts en accédant à des logiciels en ligne. De plus, la technologie Fog Computing émerge avec ses propres avantages, offrant un traitement de données plus rapide.

Dans ce chapitre nous présenterons les fondements de certains concepts relatifs à ces domaines. Nous débuterons par une analyse approfondie du concept et de l'évolution de l'IoT. Ensuite, nous aborderons les principes du Cloud Computing. Enfin nous passerons au Fog Computing pour étudier son architecture, ses domaines d'application ainsi que ses avantages et inconvénients.

2. Internet des Objets (IoT)

De nos jours, tant les individus que les organisations dépendent de plus en plus des ordinateurs et des appareils intelligents pour accomplir leurs tâches quotidiennes. Ils utilisent tous les terminaux connectés tels que les Smartphones, les ordinateurs et les tablettes pour collecter et transmettre des données. Cette technologie, appelée l'Internet des Objets (IoT) qui a permis de fournir plusieurs services et son utilisation ne se limite pas uniquement aux besoins des individus, mais elle est également exploitée par divers établissements dans le but de générer des revenus.

2.1. Définition d'IoT

IoT est plus qu'une simple technologie ; c'est plutôt un concept ou un ensemble de technologies qui sont liées. Plusieurs définitions ont été données pour mieux comprendre le concept de l'Internet des Objets, parmi ces définitions nous citons :

L'Internet des Objets est : «une infrastructure dynamique d'un réseau global. Ce réseau global a des capacités d'auto-configuration basée sur des standards et des protocoles de communication interopérables. Dans ce réseau, les objets physiques et virtuels ont des identités, des attributs physiques, des personnalités virtuelles et des interfaces intelligentes, et ils sont intégrés au réseau d'une façon transparente. » [2]

Selon P.Guillemin et P. Friess [2] « L'Internet des Objets permet aux personnes et aux objets d'être connectés à tout moment, en tout lieu, avec n'importe quel autre objet et n'importe quel utilisateur, en utilisant idéalement n'importe quel chemin/réseau et n'importe quel service. »

Selon T.Lu et W. Neng [3] « Les objets ont des identités et des représentations virtuelles qui fonctionnent dans des espaces intelligents utilisant des interfaces intelligentes pour se connecter et communiquer dans des contextes sociaux, environnementaux et d'utilisateurs. »

Grâce à des technologies telles que des systèmes d'identification normalisés et des dispositifs mobiles sans fil, l'IoT facilite la récupération, le stockage, le transfert et le traitement des données, en garantissant une continuité entre les mondes physique et virtuel [4].

2.2. Les Objets Connectés

Les objets connectés, également appelés objets intelligents car permettent une communication machine à machine sans nécessiter d'intervention humaine en tant qu'intermédiaire, et ils peuvent même prendre des décisions.

Les Objets connectés peuvent fonctionner de manière statique ou dynamique.

- Dans le mode statique, l'objet est programmé à l'avance. Par exemple, régler la cloche de l'école pour sonner chaque heure.
- Dans le mode dynamique, l'objet réagit en temps réel aux événements détectés. Par exemple, déclencher une alerte en cas de détection d'incendie.

2.3. Évolution d'IoT

L'Internet des Objets a émergé dans les années 90 avec le concept de contrôle à distance des équipements électriques et électroniques. Bill Gates a introduit ce concept en 1995 pour la première fois dans son livre "The Road Ahead". En 1998, Kevin Ashton du MIT (Massachusetts Institute of Technology) a proposé la notion d'Internet des Objets. Le laboratoire Auto-ID du MIT s'est spécialisé dans la création d'objets connectés dès 1999, en utilisant l'identification par radiofréquence et les réseaux de capteurs sans fil. En 2005, l'UIT (International Telecommunication Union) a publié un rapport traitant de la connexion entre les mondes réel et virtuel grâce aux technologies comme la RFID (Radio Frequency IDentification) et les capteurs sans fil. Enfin, en 2009, la Commission européenne a publié un plan d'action détaillant les perspectives et les enjeux du développement de l'Internet des Objets [5].

Trois Caractéristiques principales d'un objet connecté sont désignées :

1. Communication : La capacité de l'objet à échanger des données avec d'autres objets connectés via des réseaux sans fil comme le Wifi, le Bluetooth, ou des réseaux cellulaires.

2. Connexion : La capacité de se connecter à un réseau, internet, pour transmettre des données.

3. Intelligence : La capacité de comprendre les événements et même de prendre des décisions.

2.4. Domaines d'applications

L'IoT est largement utilisé dans différents domaines, notamment :

- Maison intelligente : liée à l'environnement domestique, comprenant par exemple des caméras de surveillance, des ampoules intelligentes, des garages intelligents, des réfrigérateurs intelligents, des fenêtres ...
- Transport intelligent : implique l'utilisation de plusieurs modes de transport, la surveillance pour prévenir les accidents, l'optimisation de la signalisation …
- Sécurité : Alarmes domestiques connectées à des applications mobiles pour la surveillance à distance, vidéosurveillance…
- Éducation, agriculture, sport, énergie, environnement…

3. Cloud Computing

Avant l'émergence du Cloud, nous avions l'habitude de stocker nos données sur nos appareils personnels tels que les téléphones, les clés USB (Universal Serial Bus) et les ordinateurs. Cependant, en cas de panne de l'un de ces appareils, il était souvent impossible de récupérer les données stockées. Les entreprises devaient alors investir dans leurs propres centres de données pour gérer et sécuriser leurs informations, ce qui nécessitait des ressources considérables en termes de matériel et de personnel. Heureusement, l'avènement des services du Cloud a révolutionné cette pratique, offrant aux utilisateurs la possibilité de louer de l'espace serveur plutôt que d'investir dans leur propre infrastructure. Ces services sont gérés par des entreprises spécialisées qui disposent de centres de données équipés pour garantir la sécurité et la disponibilité des données stockées.

3.1. Définition du Cloud Computing

Le Cloud Computing permet de stocker et d'accéder à distance à des données et des programmes via internet, en utilisant des serveurs distants. Cette approche offre une grande flexibilité, permettant l'utilisation depuis n'importe quel endroit et appareil connecté. En

comparaison avec le stockage local, le Cloud offre des avantages comme l'évolutivité et l'élasticité, ce qui révolutionne notre manière d'interagir avec l'informatique et les données [6].

D'après P.Mell et T.Grance du NIST(National Institute of Standards and Technology) [7], le Cloud Computing est décrit comme : « un modèle de traitement distribué qui permet un accès réseau 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.»

D'après R.Buyya, J.Broberg et A.M. Goscinski [8], le Cloud Computing est définie comme : « la livraison de services informatiques à la demande, via Internet, avec des ressources telles que des serveurs, des stockages, des bases de données, des réseaux, des logiciels et des services, permettant une tarification à l'utilisation. »

3.2. Evolution du Cloud Computing

L'idée du Cloud Computing existe depuis quelques décennies, remontant même aux années 1950 avec le traitement par serveur centralisé. À cette époque, les ordinateurs centraux étaient de vastes machines coûteuses, rendant impossible pour chaque employé d'en posséder un. Au lieu de cela, des systèmes de "partage de temps" ont été développés, permettant à plusieurs utilisateurs d'accéder à un ordinateur central depuis des terminaux déconnectés. Ce partage de puissance de calcul est à l'origine du Cloud Computing tel que nous le connaissons aujourd'hui [9].

Dans les années 1960, J.C.R. Licklider a conceptualisé un réseau interconnecté d'ordinateurs, posant ainsi les bases de ce qui allait devenir internet. Le réseau ARPANET (Advanced Research Projects Agency Network), développé en 1969, a été le premier à permettre le partage de ressources entre des ordinateurs situés à des endroits différents. Cette vision de connectivité universelle est le fondement même du Cloud Computing.

Au fil des décennies, de nombreuses technologiques avancées ont contribué à l'évolution du Cloud. Par exemple, en 1972, IBM (International Business Machines) a lancé le système d'exploitation, machine virtuelle (VM), introduisant la virtualisation. Dans les années 1990, les fournisseurs de télécommunications ont commencé à offrir des réseaux privés virtuels, facilitant l'accès partagé aux infrastructures physiques [10].

Dans les années 2000, Services Web Amazon (AWS, Amazon Web Services) est apparu, lançant Elastic Compute Cloud (EC2) en 2006, permettant aux utilisateurs de louer des ressources informatiques virtuelles. La même année, Google a lancé ses services Google Docs, permettant le stockage et la manipulation de documents dans le Cloud.

En 2007, IBM, Google et plusieurs universités ont collaboré pour créer une ferme de serveurs destinée à la recherche. C'est également l'année où des services comme Netflix ont commencé à utiliser le Cloud pour diffuser du contenu vidéo à travers le monde.

Cette évolution illustre la transition progressive vers le Cloud Computing moderne, offrant aux utilisateurs la possibilité d'accéder à des ressources informatiques à la demande, de manière flexible et économique [11].

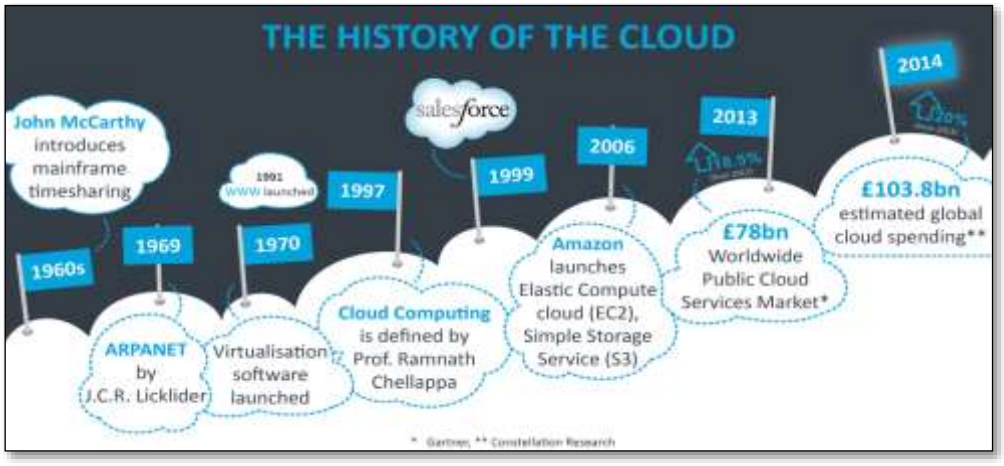

Figure I. 1.L'historique du Cloud Computing

3.3. Les caractéristiques du Cloud Computing

Le Cloud Computing présente cinq caractéristiques essentielles, telles qu'illustrées dans la figure I.2 [6].

- **Service libre à la demande :** Un utilisateur peut provisionner de manière unilatérale des capacités informatiques, telles que le stockage réseau, selon les besoins automatiquement sans nécessiter d'interaction humaine avec chaque fournisseur de service.
- **Large couverture réseau :** Les capacités sont disponibles sur le réseau, accessibles via internet et utilisent des méthodes standard qui permettent aux utilisateurs d'accéder aux services depuis divers appareils, tels que les téléphones mobiles, les ordinateurs portables et les assistants personnels numériques.

- **Partage dynamique des ressources :** Le fournisseur regroupe ses ressources informatiques pour les partager entre plusieurs utilisateurs selon un modèle multiutilisateurs. Ces ressources, qu'elles soient physiques ou virtuelles, sont assignées et réassignées automatiquement en fonction des besoins des utilisateurs. Habituellement, les utilisateurs n'ont pas de contrôle sur l'emplacement exact de ces ressources, mais peuvent souvent spécifier une préférence générale, comme le pays ou le centre de données. Ces ressources peuvent inclure le stockage, le traitement, la mémoire et la bande passante réseau.
- **Élasticité rapide :** Les capacités peuvent être rapidement adaptées et ajustées pour répondre à la demande, parfois de manière automatique. Pour les utilisateurs, cela signifie que les ressources sont facilement disponibles et peuvent être acquises en quantité souhaitée à tout moment.
- **Surveillance des ressources :** Les systèmes Cloud surveillent et optimisent automatiquement l'utilisation des ressources en utilisant des mécanismes de mesure adaptés à chaque type de service, tels que le stockage, le traitement, la bande passante et les comptes utilisateur actifs. Cette surveillance permet un contrôle précis de l'utilisation des ressources, offrant ainsi une transparence tant pour les fournisseurs que pour les utilisateurs.

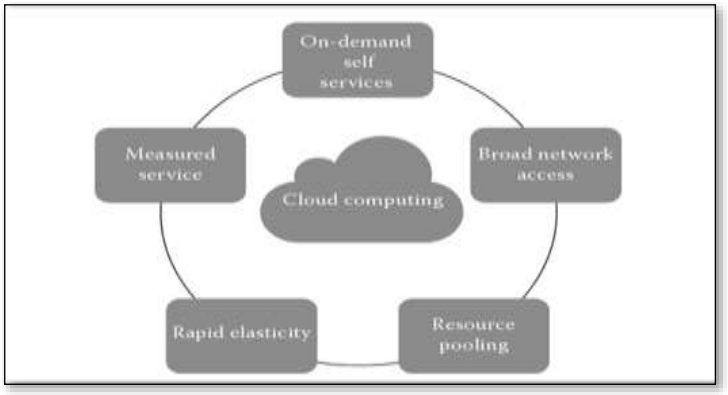

Figure I. 2.Les caractéristiques du Cloud Computing

3.4. Les types de Cloud Computing

3.4.1. Cloud public

Le Cloud public est une infrastructure flexible et ouverte, gérée par un fournisseur tiers, et accessible via internet. Dans ce modèle, les ressources informatiques sont partagées entre plusieurs entités, qu'elles soient individuelles ou des entreprises, toutes utilisant les services

fournis par ce fournisseur externe [12]. L'infrastructure du Cloud public d'un hébergeur peut être répartie entre plusieurs centres de données pour garantir la disponibilité et la sécurité. Plusieurs entreprises peuvent utiliser ces services, ce qui peut attirer des milliers de clients en fonction de la popularité, de la qualité du service et de la réputation du fournisseur Cloud. Comme on peut le constater dans la figure I.3, que toutes les entreprises ont un accès direct au Cloud.

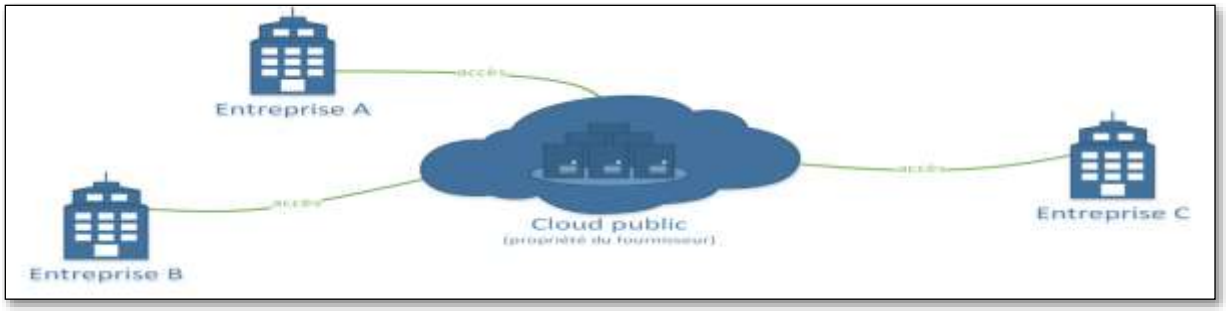

Figure I. 3. Le Cloud public

3.4.2. Cloud privé

Le Cloud privé représente un modèle informatique où toutes les ressources essentielles, telles que les serveurs, le stockage, le réseau et les licences logicielles, sont réservées à l'usage exclusif d'une seule entité, qu'il s'agisse d'une entreprise, d'une organisation gouvernementale ou d'une institution [12].

Dans la figure I.4, nous observons trois entreprises (A, B, C), où une entreprise spécifique (entreprise B) paie pour accéder à une infrastructure dédiée, comprenant un ensemble de serveurs. Dans ce scénario, seule l'entreprise B bénéficie de l'ensemble des ressources disponibles, y compris le processeur, la RAM (Random Access Memory), le stockage et le réseau, tandis que les entreprises A et C n'ont pas accès à cette infrastructure en raison du manque d'autorisations nécessaires. Chaque entreprise dispose de son propre Cloud privé et ne peut pas accéder au Cloud privé d'une autre entreprise. Cela garantit la sécurité du réseau et de l'infrastructure physique.

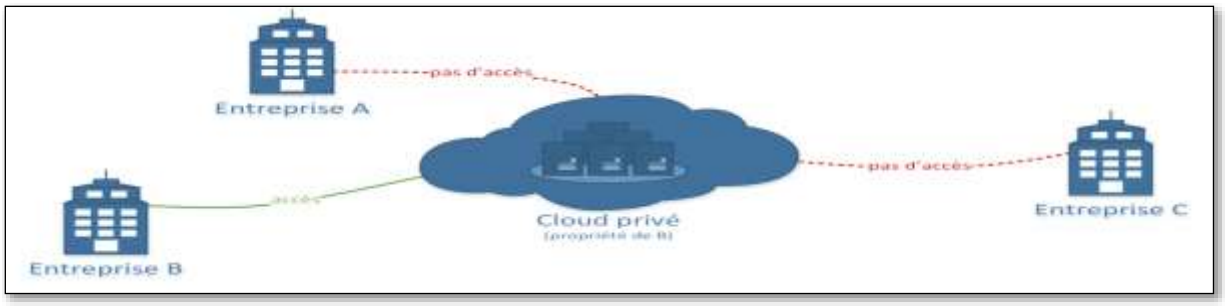

Figure I. 4. Le Cloud privé

3.4.3. Cloud communautaire

Le Cloud communautaire est conçu pour être utilisé par une communauté spécifique d'organisations partageant des intérêts communs tels que la mission, les exigences de sécurité, les politiques et les normes de conformité. Il peut être possédé, géré et exploité par une ou plusieurs des organisations de la communauté, par un tiers, ou une combinaison des deux, et peut exister sur Fog Computing ou à distance [7].

Dans la figure I.5, on constate que les entreprises A et B ont un accès direct au Cloud, étant membres de la même communauté, contrairement à l'entreprise C.

Figure I. 5. Le Cloud communautaires

3.4.4. Cloud hybride

L'infrastructure de Cloud hybride se compose d'au moins deux infrastructures de Cloud, telles que privé, communautaire ou public, qui restent des entités séparées mais sont interconnectées par une technologie standardisée ou propriétaire. Cette interconnexion permet la portabilité des données et des applications [7].

La figure I.6 illustre trois entreprises A, B et C, composé à la fois d'un Cloud public et d'un Cloud privé. Toutes ces entreprises ont un accès direct à la partie du Cloud public.

Cependant, la partie du Cloud privé est accessible uniquement à l'entreprise C. Ainsi, cette dernière peut bénéficier d'un niveau de sécurité et de contrôle supplémentaire pour ses données sensibles.

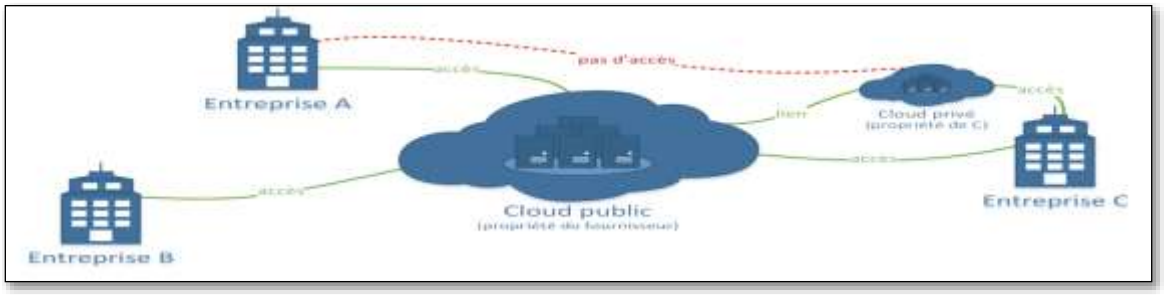

Figure I. 6. Le Cloud hybride

3.5. Différents modèles de services Cloud

Dans le cadre des recherches menées par le NIST, Mell et Grance élaborent trois modèles de services pour le Cloud Computing. La représentation graphique suivante illustre ces modèles, qui incluent IaaS : Infrastructure as a Service, le SaaS : Software as a Service, PaaS : Platform as a Service [13].

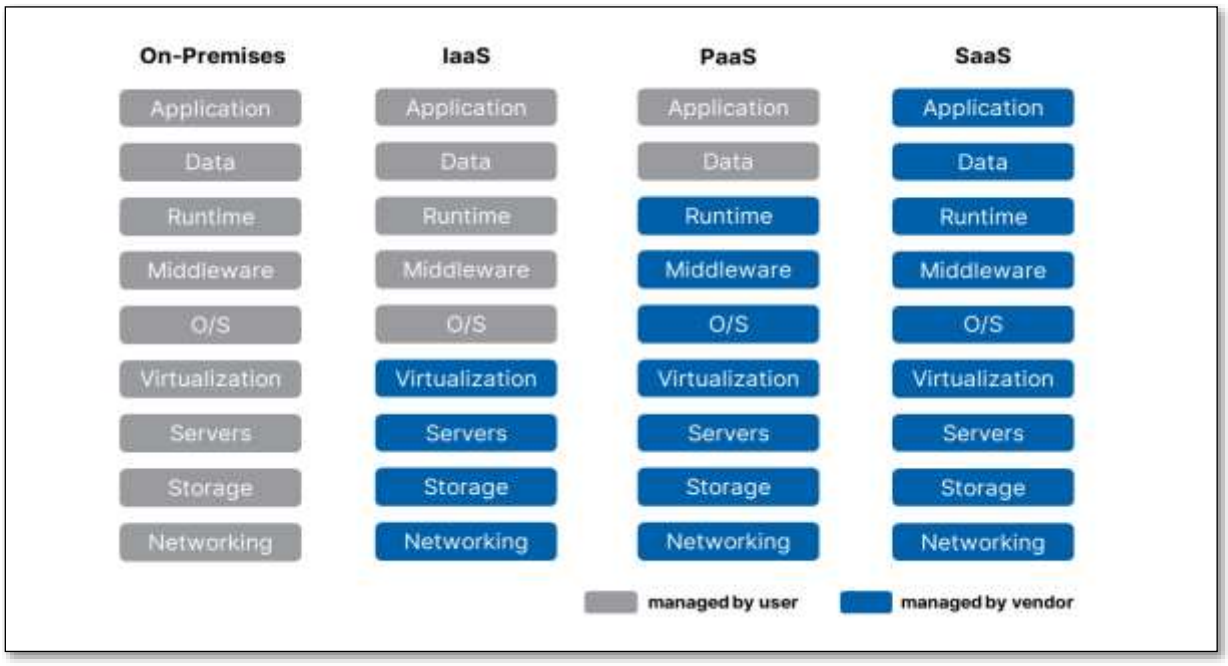

Figure I. 5. Les modèles de services Cloud

3.5.1. Infrastructure as a Service (IaaS)

L'IaaS, consiste à fournir des ressources informatiques à la demande, principalement stockées à distance dans des data center. Cette offre permet aux administrateurs d'entreprise d'accéder aux serveurs et à leurs configurations. Les clients ont la possibilité de louer des clusters, de la mémoire ou du stockage de données, avec des coûts directement liés à l'utilisation. On peut comparer ce modèle à celui des services publics comme l'électricité ou l'eau, où l'utilisation est facturée en fonction de la consommation [14].

Lors de la transition vers le Cloud (voir la figure I.7), les responsabilités sont réparties entre l'organisation (client) et le fournisseur Cloud dont l'organisation installe son propre système d'exploitation, configure et déploie ses applications, tandis que le fournisseur Cloud gère l'infrastructure sous-jacente telle que les serveurs, le réseau et le stockage.

3.5.2. Platform as a Service (PaaS)

La plateforme en tant que service est un modèle où le fournisseur prend en charge le système d'exploitation et les outils d'infrastructure, laissant au client le contrôle des applications et la possibilité d'ajouter ses propres outils. C'est similaire à l'hébergement web où le client loue l'utilisation de serveurs préconfigurés et gérés par le fournisseur, mais avec des systèmes mutualisés et une grande élasticité, permettant une adaptation automatique à la demande. En hébergement web traditionnel, l'adaptation nécessite généralement une demande formelle du consommateur [12].

En regardant la répartition des responsabilités entre le client et le fournisseur Cloud dans la figure I.7 a évolué. Le client donc peut utiliser des services PaaS pour offrir un accès rapide à un environnement complet prêt à l'emploi. Cela permet de réaliser des tests de développement, de déployer des applications sans se soucier de toute l'infrastructure sous-jacente.

3.5.3. Software as a Service (SaaS)

Un modèle qui offre aux utilisateurs une application prête à l'emploi accessible via internet dont les utilisateurs n'ont pas à s'occuper de l'installation, de la gestion ou de la maintenance de l'application ou de l'infrastructure associée [12].

Dans ce modèle, l'organisation laisse au fournisseur Cloud la responsabilité de gérer toutes les opérations. Les utilisateurs accèdent facilement aux applications via une connexion internet sur leurs appareils comme téléphone, tablette ou navigateur web. Par exemple, l'utilisation d'applications de messagerie en ligne. L'avantage pour le client est de ne plus avoir

à se préoccuper des mises à jour de l'application ou de l'infrastructure sous-jacente, puisque celles-ci sont prises en charge et hébergées par le fournisseur Cloud.

3.6. Les avantages du Cloud

- ✓ Accessibilité étendue : Les services, données et applications sont accessibles de n'importe où et depuis divers équipements.
- ✓ Optimisation des ressources : Les ressources sont ajustées aux besoins spécifiques de l'application et de l'utilisateur, évitant ainsi les pertes de performance.
- ✓ Facturation basée sur la consommation : Permet de réaliser des économies en évitant l'acquisition de matériel coûteux.
- ✓ Scalabilité automatique : Permet d'augmenter automatiquement les ressources selon les besoins, assurant une flexibilité optimale.
- ✓ Décharge de la maintenance : La maintenance de l'infrastructure est prise en charge par le fournisseur de services Cloud [15].

3.7. Les limites du Cloud

- ❖ Dépendance à la connexion internet : En cas de perte de connexion, l'accès au Cloud n'est plus possible.
- ❖ Confidentialité des données : Les données sont accessibles au fournisseur de Cloud, posant ainsi des problèmes de confidentialité.
- ❖ Partage d'infrastructure: Les données sont stockées avec celles d'autres utilisateurs, ce qui peut compromettre la sécurité.
- ❖ Contrôle limité sur les ressources physiques : L'utilisateur n'a pas de contrôle direct sur les infrastructures physiques déployées par le fournisseur.
- ❖ Gestion des versions : L'utilisateur n'a pas de contrôle sur les versions des systèmes et des logiciels déployés par le fournisseur [15].

Maintenant que nous avons discuté des différents services de Cloud Computing et de leurs avantages, il est temps d'explorer l'une des technologies clés qui les rendent possibles : la virtualisation. En effet, la virtualisation est au cœur de nombreuses infrastructures du Cloud Computing.

4. La virtualisation

La virtualisation consiste à créer une représentation virtuelle d'un dispositif physique ou d'une ressource, tel qu'un serveur, un dispositif de stockage, un système d'exploitation ou un

composant réseau. Cette technologie permet aux organisations de diviser un ordinateur ou un serveur physique en plusieurs machines virtuelles. Chaque machine virtuelle peut fonctionner de manière indépendante, exécuter différents systèmes d'exploitation ou applications, tout en partageant les ressources disponibles sur l'ordinateur hôte. En consolidant plusieurs ressources virtuelles sur un seul matériel physique (ordinateur,serveur), la virtualisation améliore la flexibilité, optimise les charges de travail et réduit les besoins en serveurs, la consommation d'énergie, les coûts d'infrastructure et la maintenance [16].

5. Fog Computing

5.1. Définition

Le Fog Computing ou “l’informatique en brouillard” est un paradigme proposé par Cisco en 2012 essentiellement pour faire face aux problématiques des latences élevées et du trafic réseau important causés par l’utilisation du Cloud [17].Son idée principale consiste à utiliser les ressources disponibles dans les équipements du réseau, positionnés entre les centres de données du Cloud et les utilisateurs finaux (et éventuellement les objets connectés), pour effectuer une partie du traitement et du stockage des données avant de les transmettre vers le Cloud.

D'aprés Qianyu Liu et al [18]: «L'informatique en brouillard, en tant que complément puissant de l'informatique en nuage, fournit des capacités supplémentaires de calcul et de stockage de données aux utilisateurs finaux dans l'Internet des objets (IoT) [19]. Dans l'informatique en brouillard, certains nœuds plus puissants (considérés comme des nœuds de brouillard en informatique) ayant de meilleures capacités de calcul doivent aider d'autres nœuds plus faibles dans le traitement de tâches. Par conséquent, un nouveau défi est apparu : comment planifier les tâches parmi les différents nœuds de brouillard afin d'obtenir un temps de réalisation des tâches et un coût de communication minimums.»

5.2. L’architecture du Fog Computing

L'architecture de l'informatique en brouillard pour l'IoT peut être divisée en trois couches distinctes : la couche d'infrastructure IoT, la couche d'informatique en brouillard et la couche d'informatique en nuage, comme illustré dans la Figure I.8 [18].

5.2.1. La couche d'infrastructure IoT :

Se compose de dispositifs terminaux intelligents tels que des nœuds capteurs. Ces dispositifs collectent des données IoT géographiquement distribuées et les transmettent à la couche d'informatique en brouillard.

5.2.2. La couche d'informatique en brouillard :

Est constituée de dispositifs interconnectés du réseau de brouillard périphérique. Ces dispositifs disposent d'une meilleure capacité de calcul et de stockage. Les nœuds de brouillard incluent à la fois des équipements matériels traditionnels tels que des commutateurs et des routeurs, ainsi que des fonctions réseau virtuelles fonctionnant sur un serveur commun à l'aide de la technologie de virtualisation des fonctions réseau. Cette couche traite les données IoT, alloue les ressources de calcul en fonction des demandes de tâches, prétraite les données IoT et renvoie les résultats de calcul à la couche d'infrastructure IoT. De plus, elle transmet les données IoT et les demandes de tâches vers la couche d'informatique en nuage.

5.2.3. La couche d'informatique en nuage :

Est équipée de grappes de serveurs haute performance offrant une capacité de calcul et de stockage beaucoup plus puissante. Elle reçoit les données IoT et les demandes de tâches téléchargées par la couche d'informatique en brouillard et traite les données selon les demandes de tâches.

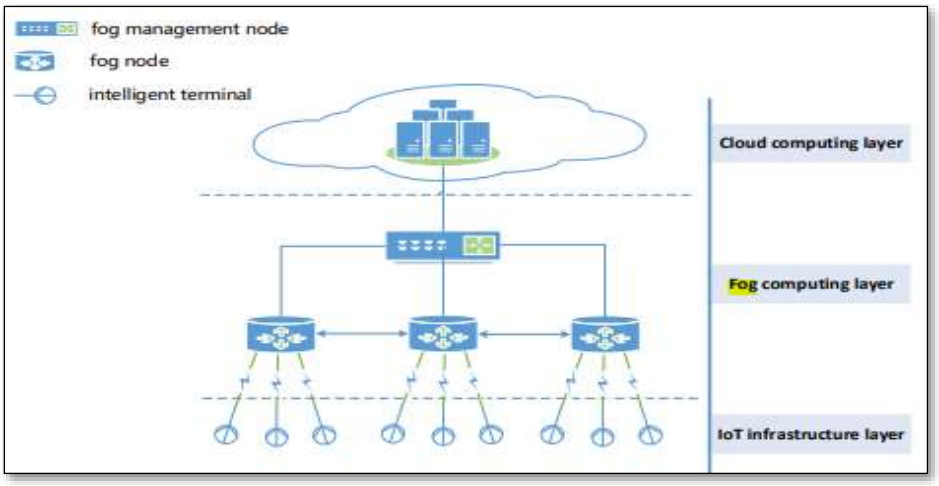

Figure I. 6. Architecteure du Fog Computing

5.3. Les caractéristiques du Fog Computing

L'informatique en brouillard est considérée comme les éléments constitutifs du nuage. Selon Ai et al. [20] et Yi et al. [21], les caractéristiques de l'informatique en brouillard peuvent être résumées comme suit :

- Conscience de l'emplacement et faible latence : L'informatique en brouillard prend en charge la conscience de l'emplacement, où les nœuds de brouillard peuvent être déployés à différents endroits. De plus, étant donné que le brouillard est plus proche des appareils finaux, il offre une latence plus faible lors du traitement des données des appareils finaux.
- Distribution géographique : Contrairement au nuage centralisé, les services et applications fournis par le brouillard sont distribués et peuvent être déployés n'importe où.
- Passage à l'échelle: Il existe des réseaux de capteurs à grande échelle qui surveillent l'environnement. Le brouillard fournit des ressources informatiques et de stockage distribuées qui peuvent fonctionner avec de tels appareils finaux à grande échelle.
- Prise en charge de la mobilité : Un des aspects importants des applications de brouillard est la capacité à se connecter directement à des appareils mobiles et donc à activer des méthodes de mobilité, telles que le protocole de séparation des identifiants de localisation qui nécessite un système de répertoire distribué.
- Interopérabilité : Les composants du brouillard peuvent interopérer et fonctionner avec différents domaines et différents fournisseurs de services.
- Prise en charge des analyses en ligne et de l'interaction avec le nuage : Le brouillard est placé entre le nuage et les appareils finaux pour jouer un rôle important dans l'absorption et le traitement des données proches des appareils finaux.

5.4. Les avantage du Fog Computing

- ✓ Faible latence: Grâce à sa proximité avec les utilisateurs finaux, le Fog peut considérablement réduire les retards, offrant ainsi la capacité de supporter des services en temps réel tels que les jeux et la diffusion vidéo [22].
- ✓ Distribution géographique étendue : Le Fog Computing peut fournir des ressources informatiques et de stockages distribués à des applications de grande envergure répartie sur une large zone géographique [22].
- ✓ Réduction des coûts d'exploitation : Économie de la bande passante réseau en traitant localement les données sélectionnées au lieu de les envoyer vers le Cloud pour analyse [23].

- ✓ Flexibilité et hétérogénéité : Le Fog Computing permet la collaboration de différents environnements physiques et infrastructures parmi plusieurs services [24].
- ✓ Scalabilité : La proximité du Fog Computing avec les appareils finaux permet d'adapter le nombre d'appareils et de services connectés [22].

5.5. Les domaines d'utilisation du Fog Computing

- Santé connectée : la santé connectée présente de nombreuses opportunités pour le Fog Computing. En permettant le traitement et l'analyse des données de santé directement au niveau des capteurs et des appareils portables, le Fog Computing offre une solution rapide pour le suivi médical en temps réel. Par exemple,un bracelet connecté mesure la fréquence cardiaque et l'activité physique. Plutôt que d'envoyer ces données à un grand centre de données sur internet, le Fog Computing permet au bracelet de les traiter et d'analyser localement, à proximité. Cela garantit également que les informations restent sécurisées car elles ne sont pas envoyées en permanence à un endroit centralisé.
- Réseaux de capteurs sans fil: En plaçant des capacités de traitement et de stockage près des capteurs, les données peuvent être traitées localement avant d'être transférées vers un centre de données. Cette approche réduit la charge sur le réseau et diminue les délais, ce qui est crucial pour les applications qui exigent une transmission rapide et une analyse immédiate des données. Par exemple, cela est particulièrement bénéfique pour des applications telles que la surveillance environnementale, la gestion de l'énergie et la détection d'incendie.
- Véhicules autonomes: Le Fog Computing est très important pour les voitures autonomes. Cela leur permet de traiter les données instantanément, ce qui est crucial pour assurer la sécurité sur la route et améliorer les performances des voitures autonomes. Par exemple, elles peuvent partager des informations sur la circulation ou les obstacles avec d'autres voitures autour d'elles, ce qui aide à prendre des décisions plus intelligentes en temps réel.
- Internet des objets (IoT): Le Fog Computing est un outil puissant dans le domaine de l'IoT. Il aide à gérer les grandes quantités de données produites par les appareils connectés en les traitants localement. Cela permet des décisions rapides et en temps réel. Par exemple, dans les maisons intelligentes, le Fog Computing permet de contrôler et d'automatiser les appareils directement depuis les capteurs, sans avoir besoin d'une connexion constante au Cloud.
- Applications industrielles: Le Fog Computing est utile pour les entreprises industrielles aussi. En rapprochant le traitement des données des machines, il permet une analyse rapide et des décisions immédiates. Cela aide à garder les machines en bon état et à éviter les arrêts

imprévus dans la production. De plus, il rend facile l'intégration de nouvelles technologies dans les usines pour améliorer les processus de fabrication et la qualité des produits.

6. La différence entre le Fog et le Cloud Computing

Les principes du Cloud Computing et du Fog Computing diffèrent sur plusieurs aspects [25] :

- Infrastructure: Le Cloud Computing utilise une infrastructure centralisée avec de vastes centres de données éloignés des appareils clients, tandis que le Fog Computing repose sur une architecture distribuée avec des nœuds placés près des appareils clients.
- Proximité aux utilisateurs: Le Cloud Computing établit une communication directe et souvent lente entre le Cloud et les appareils, tandis que le Fog Computing agit comme une interface entre les centres de données et le matériel, offrant une meilleure proximité.
- Traitement des données: Le Cloud Computing traite les données dans des centres distants, tandis que le Fog Computing les traite près des sources d'information, ce qui est crucial pour le temps réel.
- Capacités de calcul et de stockage: Le Cloud Computing offre des capacités de calcul et de stockage supérieures, tandis que le Fog Computing est composé de nombreux petits nœuds avec des capacités limitées.
- Analyses et réactivité: Le Cloud Computing effectue des analyses approfondies à long terme avec une réactivité plus lente, tandis que le Fog Computing réalise des analyses rapides à court terme avec une réactivité instantanée.
- Latence: Le Cloud Computing présente une latence plus élevée, tandis que le Fog Computing assure une faible latence.
- Dépendance à la connexion internet: Le Cloud Computing dépend fortement d'une connexion internet, tandis que le Fog Computing présente moins de risque de panne grâce à son architecture distribuée et à divers protocoles.

7. Conclusion

Le Fog Computing est un domaine vaste qui offre de nombreux avantages par rapport aux limites du Cloud. Dans le prochain chapitre, nous aborderons l'un des principaux défis du Fog : l'ordonnancement des tâches. Nous identifierons différentes techniques et algorithmes utiliser pour la résolution du ce probleme d'ordonnancement, ainsi que leur classification.

Chapitre II: Ordonnancement de tâches dans le Fog Computing

1. Introduction

L’ordonnancement présente l'un des défis majeurs dans le domaine de l'informatique en brouillard. Elle joue un rôle fondamental pour améliorer les performances de l'ensemble du système, en optimisant un ou plusieurs objectifs pour assigner efficacement les tâches aux ressources disponibles, tout en respectant les exigences de qualité de service des applications. Parallèlement, elle vise également à accroître la satisfaction de client.

Dans ce chapitre, nous commencerons par un aperçu général sur les problèmes d'ordonnancement, en abordant leurs caractéristiques et leurs différents types. Ensuite, nous décrirons en détails la classification des algorithmes d'ordonnancement dans le Fog Computing.

2. Ordonnancement : Concepts et définitions

L'ordonnancement est un concept vaste et omniprésent qui englobe de nombreux domaines de notre vie. Ainsi, la littérature propose diverses définitions de cette notion, mais elles partagent toutes un élément commun : l’assignation des tâches à des ressources dans le but d'optimiser un objectif spécifique [26].

Selon J. Carlier et P. Chrétienne, ordonnancer consiste à coordonner l'exécution d'une tâche en assignant les ressources nécessaires et en établissant leur séquence d'exécution [27].

Selon P. Esquirol et P. Lopez, le problème d'ordonnancement se définit comme le processus d'organisation temporelle des tâches, en prenant en compte les contraintes temporelles telles que les délais, ainsi que les contraintes liées à l'utilisation et à la disponibilité des ressources nécessaires [28].

L'ordonnancement est un problème NP-difficile, tel que défini par M. Pinedo dans [29] : « L'ordonnancement est un processus de prise de décision qui est utilisé régulièrement dans de nombreuses industries manufacturières et de services. Il traite de l'allocation des ressources aux tâches sur des périodes de temps données et son objectif est d'optimiser un ou plusieurs objectifs ».

3. Eléments de problème d'ordonnancement

Dans tout problème d'ordonnancement, quatre éléments essentiels se distinguent [26] :

3.1. Ressources :

Les ressources, qu'elles soient techniques ou humaines, sont des moyens limités utilisés pour accomplir des tâches.

3.2. Tâches :

Une tâche ou un job se caractérise par sa localisation temporelle, délimitée par une date de début et une date de fin, et requiert une durée prédéfinie.

3.3. Contraintes :

Les contraintes sont les limites ou les restrictions imposées par l'environnement, les ressources disponibles ou les utilisateurs.

3.4. Les objectifs :

Ce sont les critères à optimiser, c'est une ou plusieurs fonctions objectives à minimiser ou à maximiser.

4. Type d'ordonnancement

L'ordonnancement peut être classé en deux catégories principales : statique et dynamique [30]. Dans un contexte statique (hors ligne), chaque tâche reste assignée au même nœud de calcul pendant toute sa durée d'exécution. L'attribution initiale des tâches est basée sur la demande de ressources formulée par les utilisateurs, plutôt que sur les besoins réels en ressources de chaque tâche.

En revanche, dans un contexte dynamique (en ligne), les tâches ont la capacité d'être déplacées d'un nœud de calcul à un autre pendant leur exécution. Ce déplacement est effectué en tenant compte des besoins réels en ressources de chaque tâche.

5. Mesures d'évaluation des algorithmes d'ordonnancement des tâches

Les performances des algorithmes d'ordonnancement des tâches dans le Fog Computing ont été évaluées à l'aide de plusieurs mesures. Les mesures les plus utilisées sont [31] :

- Temps d'exécution : Le temps nécessaire pour terminer l'exécution d'une tâche donnée.
- Utilisation des ressources : Il s'agit de l'utilisation minimale des ressources pour exécuter le nombre maximal de tâches.

- Consommation d'énergie: Elle détermine l'énergie consommée par une ressource pour effectuer une tâche.
- Bande passante du réseau : Elle détermine le débit maximum du signal pour la transmission des données.
- Temps de réponse : C'est le temps requis pour répondre à la tâche de l'utilisateur.
- Délai d'exécution : C'est le temps acceptable pour l'achèvement d'une tâche.
- Délai : Le temps nécessaire pour transférer des données à travers le réseau.
- Coût: Le montant du budget requis pour exécuter une tâche.
- Makespan: Le temps total requis pour achever l'exécution de toutes les tâches.
- Le temps de complétion : La somme totale des temps d'exécution de toutes les tâches dans le système.

6. Les algorithmes classiques d'ordonnancement

6.1. L'algorithme Min-min:

Simplifie la planification des tâches en calculant d'abord le temps d'exécution minimal pour chaque tâche sur chaque ressource disponible, puis en choisissant itérativement la tâche avec le temps d'exécution minimal le plus court, l'assignant à la ressource correspondante, mettant à jour les temps d'exécution des autres tâches sur la même ressource, et répétant le processus jusqu'à ce que toutes les tâches soient attribuées à des ressources. En résumé, Min-min optimise l'utilisation des ressources en sélectionnant et en assignant les tâches en fonction de leur temps d'exécution minimal [32].

6.2. L'algorithme Max-min :

Fonctionne en calculant d'abord les temps d'exécution minimum pour chaque tâche, puis en sélectionnant la tâche avec la durée maximale parmi toutes celles sur les ressources. Cette tâche est ensuite ordonnancée sur la machine correspondante. Ensuite, les temps d'exécution des autres tâches sur cette machine sont mis à jour en ajoutant le temps d'exécution de la tâche assignée. Cette tâche est alors retirée de la liste des tâches, et le processus est répété jusqu'à ce que toutes les tâches soient assignées sur les ressources disponibles [33].

6.3. L'algorithme EDF (Earliest Deadline First) :

La notion d'algorithme Earliest Deadline a été introduite par Liu et Layland [34] en 1973 .fonctionne sur le principe de donner la priorité à l'exécution des tâches dont la date limite est la plus proche. Cela signifie que les tâches nécessitant une réalisation rapide sont exécutées en premier. Principalement utilisé dans les systèmes temps réel, EDF est un schéma

d'ordonnancement préemptif où la tâche la plus urgente est sélectionnée dynamiquement en fonction de son échéance. Cependant, sa mise en œuvre est complexe et il peut se comporter de manière suboptimale en cas de surcharge du système, ce qui limite sa fréquence d'utilisation [35].

6.4. L'algorithme Round Robin:

Adopte une approche simple consistant à répartir équitablement les tâches sur les machines virtuelles disponibles, garantissant ainsi que le nombre de tâches pour chaque machine virtuelle reste uniforme. Cette méthode est mise en œuvre dans le simulateur CloudSim [II.12].

6.5. L'algorithme FIFO (First In First Out):

Egalement connu sous le nom de FCFS (First Come First Served), est l'un des algorithmes les plus simples utilisés en gestion des ressources. Son fonctionnement est direct :chaque tâche et ressource disponibles sont ajoutées dans une file d'attente, puis exécutées dans l'ordre chronologique de leur arrivée. En d'autres termes, la première tâche ou ressource arrivée est la première à être traitée. Cet algorithme est mis en œuvre dans le simulateur CloudSim [36].

7. Classification des algorithmes d'ordonnancement dans le Fog Computig

Plusieurs chercheurs ont proposé diverses approches pour aborder le problème d'ordonnancement des tâches dans l'informatique en brouillard. Nous proposons la classification suivante :

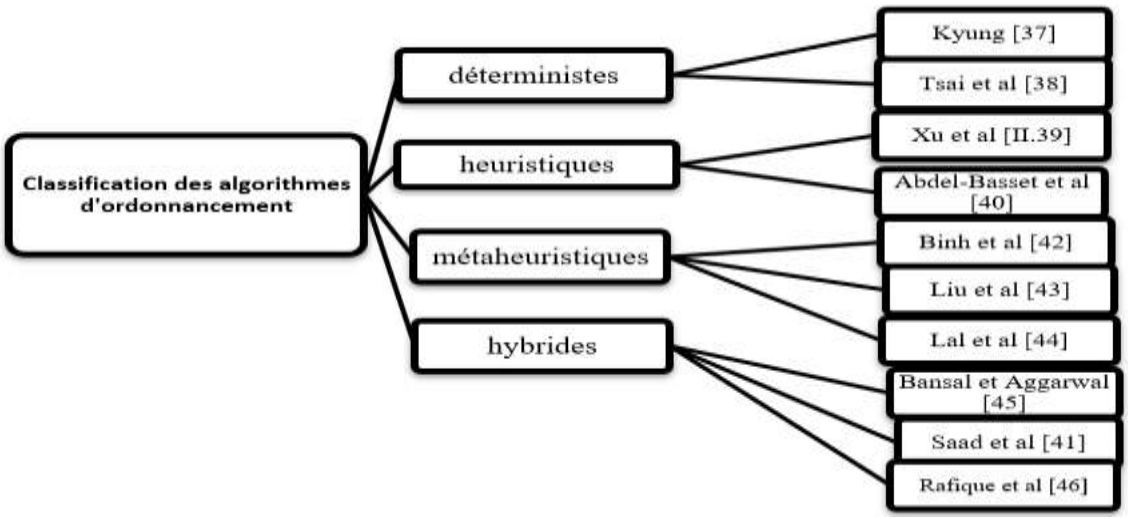

Figure II. 1. Classification des algorithmes d'ordonnancement dans le Fog Computing

7.1. Les algorithmes déterministes

Les mécanismes déterministes d'ordonnancement des tâches en brouillard utilisent des règles et des algorithmes prédéfinis pour attribuer les tâches aux nœuds en brouillard, produisant toujours le même résultat pour une entrée donnée. Bien que simples à mettre en œuvre, ils ne garantissent pas toujours une utilisation optimale des ressources ou une réduction de la latence. La recherche exhaustive est un exemple de cette approche, énumérant tout l'espace de recherche pour trouver le plan optimal. Ces méthodes excluent toute composante aléatoire ou probabiliste.

Kyung [37] a présenté un modèle d'ordonnancement des tâches basé sur la priorité des tâches tenant compte des nœuds en brouillard statiques et opportunistes par rapport à leur mobilité. Dans ce modèle, il est possible de traiter les tâches sensibles aux retards dans les nœuds en brouillard statiques et les tâches insensibles aux retards dans les nœuds en brouillard opportunistes. Sur la base des résultats présentés dans cet article, le modèle proposé est plus performant que les autres modèles traditionnels en termes de délai de réponse. Cependant, cette approche présente certains inconvénients potentiels. Le classement des nœuds en brouillard ne reflète pas toujours avec précision leurs capacités de calcul réelles ou leur charge de travail. De plus, la priorité des tâches peut entraîner un retard ou une négligence des tâches de moindre priorité, ce qui pourrait affecter les performances globales du système.

Tsai et al. [38] ont proposé un modèle de transformation linéaire pour résoudre le problème d'attribution de tâches non linéaires dans les systèmes Cloud-Fog. Divers critères, notamment le temps d'exécution et les coûts d'exploitation, sont pris en compte pour l'attribution optimale des tâches. Le modèle proposé peut trouver la solution optimale en fonction des exigences des tâches, de la vitesse de traitement des nœuds et du coût d'utilisation des ressources des nœuds. Bien que l'approche déterministe proposée puisse identifier une solution optimale, la complexité de calcul de l'approche augmente rapidement avec la taille du problème.

7.2. Les algorithmes Heuristique

Les méthodes basées sur l'heuristique consistent à utiliser des règles ou des heuristiques pour attribuer des tâches aux nœuds Fog en fonction de leurs caractéristiques, telles que la proximité, la disponibilité et la charge de travail. Ces méthodes sont simples et efficaces mais ne fournissent pas toujours des solutions optimales. De nombreux chercheurs ont amélioré les méthodes heuristiques pour obtenir de meilleures performances en matière de planification [31].

Xu et al. [39] ont introduit l'algorithme FCAS (Fog Computing Adaptive Scheduling Algorithm), basé sur la programmation dynamique adaptative, pour optimiser le chemin de traitement des données avec différentes priorités aux nœuds de brouillard. L'objectif principal de cet algorithme est de minimiser le temps de latence et la consommation d'énergie lors de l'exécution des tâches prioritaires. Les résultats des simulations ont confirmé son efficacité accrue et sa fiabilité améliorée, en réduisant également de manière significative la consommation d'énergie associée.

Abdel-Basset et al. [40] ont proposé trois algorithmes basés sur des prédateurs marins heuristiques pour résoudre le problème d'ordonnancement des tâches dans un environnement Fog-Cloud. En plus de l'algorithme MPA standard (MPA,Marine Predators Algorithm), les auteurs ont introduit deux autres versions : le MPA modifié (MMPA Modified Marine Predators Algorithm) et le MMPA amélioré (IMMPA, Improved Modified Marine Predators Algorithm). Dans le MMPA, la capacité d'exploitation de l'algorithme MPA est améliorée en se basant sur les positions les plus récemment mises à jour plutôt que sur la meilleure position précédente. Le but de l'algorithme proposé, le MPA et ses versions améliorées (MMPA et IMMPA), est d'optimiser l'ordonnancement des tâches dans le Fog Computing afin d'améliorer la qualité de service pour les applications IoT. Cela inclut la réduction de la consommation d'énergie, du temps de réponse, du makespan (durée totale pour exécuter toutes les tâches), du temps d'écoulement (temps entre la soumission et la fin de toutes les tâches).

7.3. Les algorithmes métaheuristiques

Il s'agit d'algorithmes d'optimisation, inspirés par des phénomènes naturels. Ils sont capables de gérer des espaces de recherche complexes et d'explorer efficacement des solutions diverses [41].

Binh et al. [42] ont introduit un algorithme d'ordonnancement génétique appelé Time-Cost aware Scheduling (TCaS) dans IoT-Cloud-Fog, basé sur le temps d'exécution et le coût opérationnel. Le TCaS proposé vise à atteindre un bon compromis entre différents critères, notamment le temps, le coût et la satisfaction de l'utilisateur. Dans cette méthode, un modèle à trois couches comprenant la couche client, brouillard et nuage est utilisé pour l'attribution des ressources. L'algorithme tente d'attribuer des tâches aux couches client et brouillard, et le reste des exigences sont satisfaites par la couche nuage. Les performances de TCaS sont évaluées avec différents ensembles de tâches. Plusieurs mesures sont utilisées pour évaluer ces

performances, y compris le temps de réponse total, le temps de traitement et le coût du centre de données pour l'exécution de ces tâches.

Liu et al. [43] ont proposé un algorithme d'ordonnancement de tâches génétiques à double fitness adaptatif (ADGTS, Adaptive Double fitness Genetic Task Scheduling). Grâce à la planification collaborative des tâches et des ressources Fog, l'ADGTS peut optimiser simultanément le makespan des tâches et le coût de communication, ainsi que s'adapter de manière flexible à l'importance différente du makespan et du coût de communication dans les applications pratiques. Les résultats de simulation montrent que cet algorithme proposé a de meilleures performances que l'algorithme traditionnel Min-Min en termes de makespan, et peut en outre équilibrer les performances de makespan et du coût de communication.

Lal et al. [44] ont proposé une optimisation par colonie de fourmis (ACO, Ant Colony Optimization) comme méthode méta-heuristique pour minimiser le coût total d'exécution et le temps total d'exécution. Divers ensembles de données de flux de travail ont été utilisés pour l'évaluation de l'ACO et les résultats obtenus ont prouvé l'efficacité de l'ACO par rapport à GA. Cependant, l'ACO a montré un temps de réponse plus long.

7.4. Les algorithmes hybrides

Les algorithmes hybrides combinent les avantages de deux ou plusieurs méthodes d'optimisation pour résoudre des problèmes complexes. Cette approche vise à exploiter les forces de chaque méthode tout en minimisant leurs faiblesses, pour obtenir de meilleures performances.

Bansal et Aggarwal. [45] ont développé une nouvelle approche hybride appelée algorithme d'optimisation des particules baleines (PWOA, Particle Whale Optimization Algorithm) pour surmonter les limitations du l'algorithme d'optimisation par essaim de particules (PSO,Particle Swarm Optimization) dans les espaces de recherche de haute dimension, caractérisées par une convergence lente vers l'optimum global, ainsi que l'exploration limitée des espaces de recherche par l'algorithme d'optimisation des baleines (WOA,Whale Optimization Algorithm) basé sur une population. En combinant les meilleures caractéristiques du PSO et du WOA, le PWOA améliore les capacités d'exploration et d'exploitation. L'objectif principal de l'algorithme PWOA est de minimiser le temps total d'exécution (TET) et le coût total d'exécution (TEC) associés aux tâches dépendantes dans un environnement de Cloud-Fog Computing. Les résultats de simulations démontrent que le

PWOA minimise à la fois le temps total d'exécution et les coûts plus efficacement que le PSO et le WOA.Haut du formulaireBas du formulaire

Saad et al. [41] ont proposé un algorithme hybride d'optimisation multi-objectif de l'ordonnancement des tâches dans des environnements d'informatique en brouillard, combinant les forces de l'algorithme génétique (GA, Genetic Algorithm) et de l'optimisation par essaims de particules (PSO). L'approche hybride combine les avantages de GA et PSO, réalisant une exploration et une exploitation efficaces de l'espace de recherche, ce qui conduit à des performances améliorées par rapport aux approches traditionnelles à un seul algorithme. Les résultats de l'algorithme hybride proposé ont amélioré le temps d'exécution par rapport à l'algorithme GA, Hybrid PWOA et l'algorithme PSO, ainsi que le temps de réponse et le temps de complétion.

Rafique et al. [46], ont proposé un nouvel algorithme hybride inspiré par la biologie (NBIHA, A Novel Bio-Inspired Hybrid Algorithm), qui combine une optimisation par essaim de particules modifiée (MPSO, Modified Particle Swarm Optimization) et une optimisation par essaim de chats modifiée (MCSO, Modified Cat Swarm Optimization). Dans cette approche proposée, le MPSO est utilisé pour ordonnancer les tâches parmi les dispositifs Fog, et l'hybride MPSO-MCSO est utilisé pour gérer les ressources au niveau des dispositifs Fog. Dans cette approche, les ressources sont attribuées et gérées en fonction de la demande des requêtes entrantes. L'objectif principal de ce travail proposé est de réduire le temps de réponse et d'optimiser l'utilisation des ressources en planifiant efficacement les tâches et en gérant les ressources Fog disponibles.

Une comparaison des méthodes d'ordonnancement étudiées présentée dans le tableau II.1

Catégorie	Travaux	Techniques	Temps de réponse	coût	Makespan	Temps d' exécution	Utilisation des ressources	Temps de complétion	Consommation d' énergie
Déterministes	Kyung [37]	Modèle basé sur la propriété des taches	X						
	Tsai et al[38]	Modèle de transformation linéaire		X		X	X		
Heuristiques	Xu et al [39]	Programmation dynamique adaptative					X		X
	Abdel – basset [40]	Basé sur des prédateur marins (MPA ,MMPA,IMMPA)	X		X				X
Métaheruistiques	Binh et al [42]	Ordonnancement génétique	X	X		X			
	Liu et al [43]	Ordonnancement des taches double fitness adaptative (ADGTS)		X	X				
	Lal et al [44]	Colonie de fourmis (ACO)		X	X				
Hybrides	Bensal et Aggarwal[45]	Algorithme d'optimisation des particules baleines (PWOA)		X	X				
	Saad et al [41]	Algorithme hybride multi-objectif	X			X		X	
	Rafique et al [46]	Algorithme inspiré par la biologie	X				X		

Tableau. II. 1. Comparaison des méthodes d'ordonnancement des travaux étudiés

8. L'optimisation

8.1. Définition de l'optimisation

Nous présenterons quelques définitions de l'optimisation proposées par différents auteurs :

Selon Talbi, E. G [47] : "L'optimisation est l'art de trouver les meilleures solutions pour des problèmes complexes dans divers domaines en utilisant des méthodes algorithmiques."

Selon Pardalos, P. M.et Resende, M. G. C [48]: "L'optimisation est le processus de recherche de la meilleure solution, souvent définie comme la solution qui minimise ou maximise une fonction objectif tout en satisfaisant un ensemble de contraintes."

8.2. Problème d'optimisation

Le problème d'optimisation implique la recherche du minimum ou du maximum (optimum) d'une fonction spécifiée, souvent appelée fonction objectif [49], dans Le but de trouver la (ou les) solution(s) optimale(s) parmi un ensemble de solution, appelé espace de recherche ou espace de solutions [50].Cette fonction peut représenter un objectif à minimiser (comme les coûts) ou à maximiser (comme les bénéfices). La définition du problème d'optimisation comprend généralement des contraintes, qui sont des conditions que les paramètres ou variables de décision doivent respecter [49].

Dans un problème de minimisation, l'objectif est de trouver une solution x* telle que la fonction objectif f(x*) soit la plus petite possible parmi toutes les solutions x dans l'ensemble S. Dans ce cas, la fonction objectif f est souvent appelée une fonction de coût [50].

$$f(x^*) \leq f(x), \forall x \in S, soit \min f(x) pour\ tout\ x \in S \qquad \text{(II.1)}$$

Dans un problème de maximisation, l'objectif est de trouver une solution x* telle que la fonction objectif f(x*) soit la plus grande possible parmi toutes les solutions x dans l'ensemble S. Dans ce cas, la fonction objectif f est souvent appelée une fonction d'utilité, de profit ou de fitness.

$$f(x^*) \geq f(x), \forall x \in S, soit \max f(x)\ pour\ tout\ x \in S \qquad \text{(II.2)}$$

Les problèmes d'optimisation sont généralement classés en deux catégories distinctes : ceux impliquant des variables continues et ceux impliquant des variables discrètes, également connus sous le nom de problèmes combinatoires. Pour les problèmes avec des variables continues, la solution est généralement exprimée par un ensemble de nombres réels, tandis que

pour les problèmes combinatoires, la solution est souvent représentée par un ensemble fini de nombres entiers [50].

8.3. Méthodes de résolution de problèmes d'optimisation

Il existe de nombreuses méthodes de résolution des problèmes d'optimisation varient en fonction de la nature du problème, des contraintes et des objectifs.

8.3.1. Problème d'optimisation combinatoire

L'optimisation combinatoire (OC) joue un rôle central à la fois en recherche opérationnelle et en informatique, étant donné sa capacité à résoudre un large éventail de problèmes. De nombreuses applications pratiques peuvent être formulées sous forme de problèmes d'OC, tels que le voyageur de commerce, le problème du sac à dos multiple et l'ordonnancement de tâches [51].

Un problème d'optimisation combinatoire (POC) est caractérisé par un ensemble fini et discret de solutions, ainsi qu'une fonction objectif pour évaluer chaque solution [II.28]. Ces problèmes visent à trouver des solutions optimales dans des contextes où la structure sous-jacente est discrète. En théorie, résoudre ces problèmes semble simple car il suffit d'explorer toutes les solutions possibles et de comparer leurs qualités pour déterminer la meilleure.

Cependant, dans la pratique, énumérer toutes les solutions peut prendre énormément de temps, or, le temps nécessaire pour trouver la solution optimale est souvent un facteur important, ce qui rend les problèmes d'optimisation combinatoire réputés pour leur complexité [50].

Il existe plusieurs approches pour résoudre les problèmes d'optimisation combinatoire (POC). On peut les classer généralement en deux catégories : les méthodes exactes et les méthodes approchées [52].

8.3.2. Les méthodes exactes

Les méthodes exactes sont efficaces pour trouver la meilleure solution à tous les types de problèmes d'optimisation combinatoire. Cependant, cette recherche peut être chronophage, en particulier pour les problèmes complexes. Elles sont généralement adaptées aux problèmes de taille moyenne ou petite en raison de leur utilisation intensive de la mémoire et du risque d'interruption prématurée [53].Parmi ces méthodes, on compte la programmation linéaire en

nombres entiers, les algorithmes de recherche arborescente et la programmation dynamique. Bien qu'elles offrent théoriquement une solution optimale, leur coût en termes de temps de calcul et de mémoire peut être élevé [50].

8.3.3. Les méthodes approchées

La recherche d'une solution optimale est fréquemment un défi complexe et ardu, surtout lorsqu'il s'agit de trouver rapidement des solutions satisfaisantes. C'est pourquoi les méthodes approchées sont souvent privilégiées pour accélérer ce processus et obtenir rapidement des solutions acceptables dans des situations où une exploration exhaustive est peu pratique. Ces méthodes offrent des solutions de qualité raisonnable dans des délais raisonnables. Elles peuvent être catégorisées en deux classes principales : les heuristiques et les métaheuristiques [50].

8.3.3.1. Heuristiques

Une heuristique est une approche de résolution adaptée à un problème spécifique, visant à trouver une solution réalisable dans un délai raisonnable, même si elle n'est pas forcément optimale. Son utilisation est pertinente pour obtenir une solution approximative à un problème complexe, accélérant ainsi le processus de résolution exacte [54].

Les heuristiques se divisent en deux catégories principales [53]:

- **Méthodes constructives:** Ces méthodes élaborent des solutions en commençant par une solution de départ, puis en ajoutant progressivement des éléments jusqu'à ce qu'une solution complète soit obtenue.
- **Méthodes de fouilles locales :** qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière répétitive essaient d'améliore cette solution en explorant son voisinage.

8.3.3.2. Métaheuristiques

Devant les défis auxquels sont confrontées les heuristiques pour trouver des solutions de qualité satisfaisante pour les problèmes d'optimisation complexes, les méta-heuristiques ont été développées pour répondre à ces difficultés.

Une métaheuristique est un processus répétitif qui dirige une heuristique en combinant plusieurs concepts pour explorer et exploiter l'ensemble de l'espace de recherche. Des stratégies

d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales [55].

Les métaheuristiques peuvent être classées en deux catégories [55]:

- Une métaheuristique à base de solution unique : manipule une seule solution durant la procédure de recherche. Elle lance la recherche avec une seule solution et essaye d'améliorer sa qualité au cours des itérations.
- Une métaheuristique à base de population de solutions : manipule un ensemble de solutions parallèlement, elle lance la recherche avec une population de solutions et essaye d'améliorer leurs qualités au cours des itérations dans le but de fournir la ou les meilleures solutions trouvées.

La figure II.2 montre les méthodes de résolution d'un problème d'optimisation

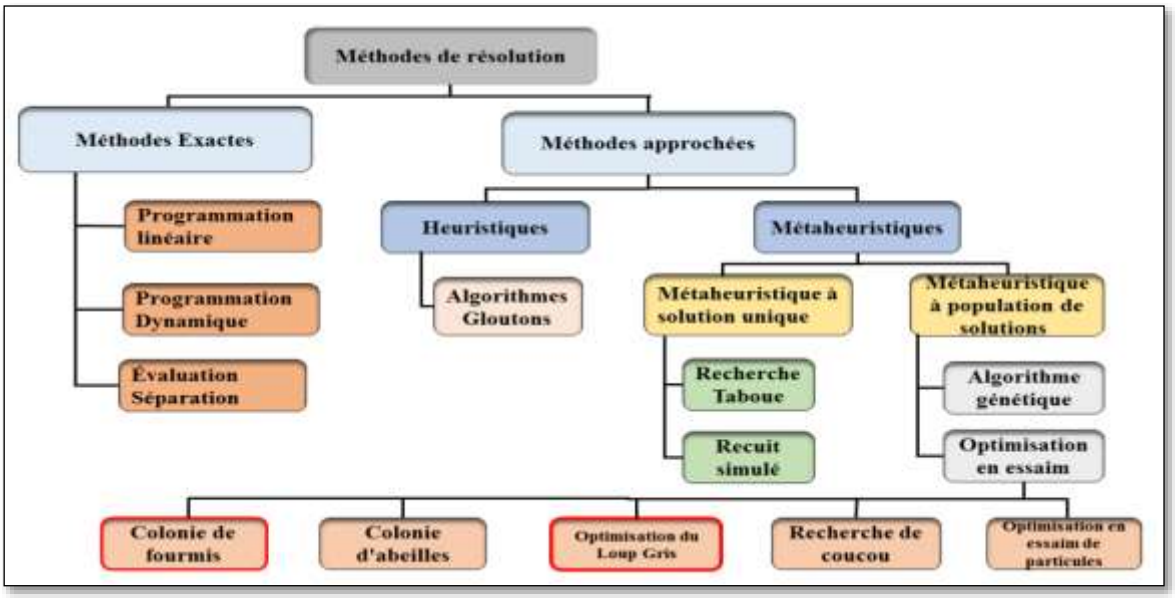

Figure II. 2. Les méthodes de résolution d'un problème d'optimisation

Nous pouvons classer les métaheuristiques en trois catégories : la recherche locale, les métaheuristiques à population et les méthodes hybrides. La figure II.3 illustre ces trois catégories.

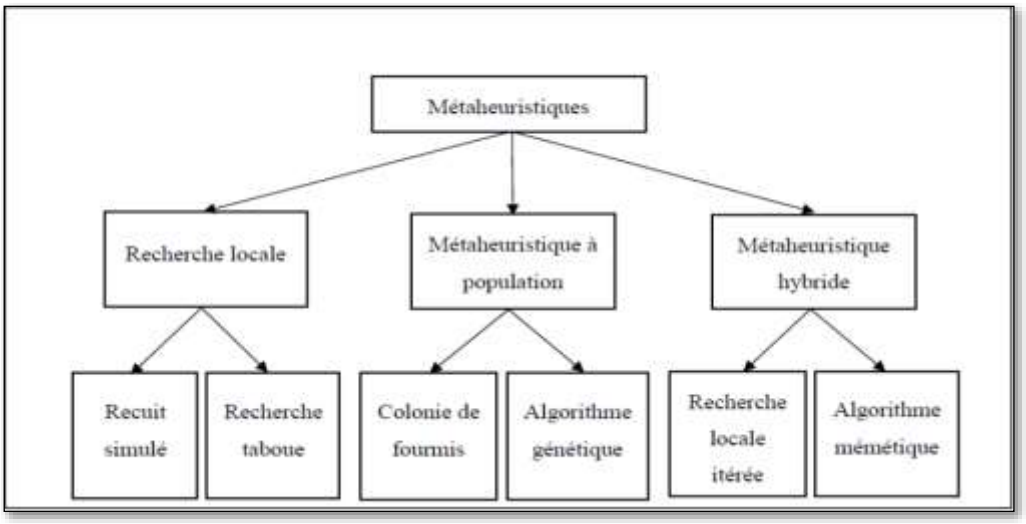

Figure II. 3. Les catégories des métaheuristiques

9. Algorithmes ALO (Ant Lion Optimizer)

9.1. Algorithme d'optimisation d'antlion ALO

Un nouvel algorithme inspiré de la nature, appelé optimiseur du fourmilion, imite le mécanisme de chasse des fourmilions dans la nature. Cet algorithme intègre cinq étapes principales de la chasse aux proies: la marche aléatoire des fourmis, la construction des pièges, le piégeage des fourmis dans ces pièges, la capture des proies et la reconstruction des pièges. Comme les fourmis se déplacent de manière stochastique lorsqu'elles cherchent de la nourriture, une marche aléatoire est utilisée pour modéliser leur mouvement de la manière suivante [56] :

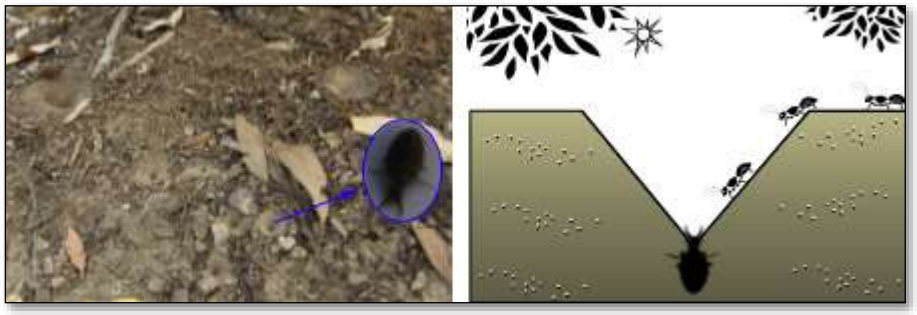

Figure II. 4. Des pièges en forme de cône et le comportement de chasse des fourmilions

9.2. Les opérateurs de l'algorithme ALO

L’algorithme ALO simule cinq étapes principales :

9.2.1. La marche aléatoire de fourmi

Les marches aléatoires des fourmis sont régies par l'équation (II.1). À chaque étape de l'optimisation, les fourmis mettent à jour leurs positions en effectuant une marche aléatoire. Cette marche est ensuite normalisée à l'aide de l'équation (II.3) pour s'assurer que les positions restent à l'intérieur de l'espace de recherche:

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

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

$$r(t) = \begin{cases} 1 \rightarrow rand \ > 0{,}5 \\ 0 \rightarrow rand \ \leq 0{,}5 \end{cases} \qquad \text{(II.2)}$$

La marche aléatoire discutée ci-dessus est utilisée dans l'équation (II.3) pour mettre à jour la position des fourmis:

$$X_i^t = \frac{(X_i^t - a_i) \times (a_i - c_i^t)}{(d_i^t - a_i)} + c_i \qquad \text{(II.3)}$$

Où $\boldsymbol{a_i}$ est le minimum de la marche aléatoire de la variable $\boldsymbol{i}$, $\boldsymbol{d_i}$ est le maximum de la marche aléatoire dans la variable $\boldsymbol{i}$.

9.2.2. Le piégeage des fourmis dans les pièges

Les fourmis se déplacent dans une hypersphère définie par les vecteurs c et d, autour d'un fourmilion sélectionné, influencé par les pièges des fourmilions. Pour modéliser mathématiquement cette hypothèse, les équations suivantes sont proposées :

$$c_i^t = Antlion_j^t + c^t, d_i^t = Antlion_j^t + d^t \qquad \text{(II.4)}$$

Où $\boldsymbol{c_i^t}$ et $\boldsymbol{d_i^t}$ sont le minimum et le maximum de toutes les variables à l'itération $\boldsymbol{t}$, et $\boldsymbol{Antlion_j^t}$ est une position de fourmilion **j** sélectionnée à l'itération $\boldsymbol{t}$.

9.2.3. La construction des pièges

Pour représenter la capacité de chasse des fourmilions, on utilise une roue de roulette pour choisir les fourmilions en fonction de leur aptitude lors de l'optimisation. Les fourmis sont envisagées comme étant piégées dans un seul fourmilion sélectionné. Ce processus accorde de meilleures chances aux fourmilions les plus performants pour attraper des fourmis.

9.2.4. Glissement des fourmis vers le fourmilion

Lorsque les fourmilions réalisent qu'une fourmi est piégée, ils projettent du sable vers l'extérieur du centre du piège. Ce mouvement fait glisser la fourmi piégée qui essaie de s'échapper. Pour modéliser mathématiquement ce phénomène, le rayon de l'hypersphère des déplacements aléatoires des fourmis est adaptativement réduit. Les équations suivantes sont avancées à cet effet :

$$c^t = \frac{c^t}{I} \qquad (II.5), d^t = \frac{d^t}{I} \qquad (II.6)$$

Où $\boldsymbol{I}$ est le rapport, $\boldsymbol{c^t}$ est le minimum des variables totales à l'itération **t**, et $\boldsymbol{d^t}$ est le maximum des variables totales à l'itération **t**.

9.2.5. La capture des proies et la reconstruction des pièges

À la dernière étape de la chasse, une fourmi est capturée au fond du piège par le fourmilion. Ensuite, le fourmilion tire la proie dans le sable pour la consommer. Pour reproduire ce processus, on suppose que la capture de la proie se produit lorsque les fourmis deviennent plus performantes que leur fourmilion respectif. En conséquence, le fourmilion ajuste sa position pour se rapprocher de la dernière position connue de sa proie. Une équation est proposée pour formaliser ce processus :

$$Antlion_j^t = Ant_i^t \quad if \ \ f(Ant_i^t) > f\left(Antlion_j^t\right) \qquad (II.7)$$

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

9.2.6. Élitisme

Elitisme est une caractéristique importante des algorithmes évolutifs qui leur permet de maintenir la meilleure ou les meilleures solutions obtenues à n'importe quelle étape du processus d'optimisation. Dans cette étude, le meilleur fourmilion obtenu jusqu'à présent à chaque itération est sauvegardé et considérée comme une élite. Étant donné que l'élite représente le fourmilion la plus adaptée, elle est supposée influencer les déplacements de toutes

les fourmis durant les itérations. Ainsi, il est postulé que chaque fourmi se déplace aléatoirement autour d'un fourmilion sélectionnée par la méthode de la roulette, tout en tenant compte de l'élite, selon la procédure suivante : $\boldsymbol{Ant_i^t} = \frac{R_A^t + R_E^t}{2}$ (II.8)

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

10. Algorithme PSO (Particle Swarm Optimization)

L'optimisation par essaim de particules, introduit en 1995 par Kennedy et Eberhart [57], est l'une des dernières techniques d'optimisation évolutive, inspirées par la nature. C'est une méthode adaptative qui peut être utilisée pour résoudre des problèmes d'optimisation. La recherche est effectuée à l'aide d'une population de particules, chaque particule correspondant à un individu dans l'algorithme évolutif. Un essaim de particules est généré de manière aléatoire initialement, chaque position de particule représentant un point de solution possible dans l'espace du problème. Chaque particule met à jour son vecteur de position X_i et son vecteur de vitesse V_i en se déplaçant à travers l'espace du problème. Kennedy et Eberhart ont proposé la formule de mise à jour du vecteur de position X_i

$$X_i(t+1) = X_i(\text{t}) + V(\text{t+1}) \quad \text{(II.9)}$$

La formule de mise à jour du vecteur de vélocité V_i(t+1)

- $V_i(\text{t+1}) = w * V(\text{t}) + c_1 * rand_1 * (pbest_i - X_i(\text{t})) + c_2 * rand_2 * (gbest - X_i(\text{t+1}))$ (II.10)
- w est le poids d'inertie, contrôlant l'impact de la vélocité précédente.
- c_1 et c_2 ont les coefficients d'accélération pour l'influence personnelle et globale, respectivement.
- $rand_1$ et $rand_2$ ont des valeurs aléatoires comprises entre 0 et 1.
- $pbest_i$ est la meilleure position locale de la particule i.
- $gbest$ est la meilleure position globale de la particule i.
- x_i (t) est la position actuelle de la particule i à l'instant t.

10.1. Les étapes de PSO

10.1.1. Initialisation

- Initialiser les paramètres de l'algorithme, tels que le nombre de particules, les coefficients d'accélération c_1 et c_2, le poids d'inertie w.

- Initialiser aléatoirement la position et la vélocité de chaque particule dans l'espace de recherche.

10.1.2. Évaluation de la fitness

- Calculer la valeur de fitness de chaque particule en utilisant la fonction de fitness définie. Cette valeur représente la qualité de la solution que chaque particule représente.

10.1.3. Mise à jour de la meilleure position locale ($pbest$)

- Comparer la fitness actuelle de chaque particule à sa meilleure fitness locale précédente $pbest_i$. Si la fitness actuelle est meilleure, mettre à jour $pbest_i$ avec la position actuelle de la particule.

$$pbest_i = \begin{cases} x_i(t) & if\ fitness\ (x_i(t) < fitness(pbest_i)) \\ pbest_i & sinon \end{cases} \quad \text{(II.11)}$$

10.1.4. Mise à jour de la meilleure position globale ($gbest$)

- Identifier la particule ayant la meilleure fitness parmi toutes les particules de l'essaim. Cette particule est désignée comme la meilleure globale $gbest$.

$$pbest_i = \begin{cases} x_i(t) & if\ fitness(pbest_i) < fitness(pbest_i)) \\ gbest & sinon \end{cases} \quad \text{(II.12)}$$

10.1.5. Mise à jour de la vélocité et de la position

- Mettre à jour la vélocité de chaque particule en utilisant l'équation de mise à jour de la vélocité PSO.

- Mettre à jour la position de chaque particule en utilisant la nouvelle vélocité calculée.

10.1.6. Critère d'arrêt

- Vérifier si un critère d'arrêt prédéfini est atteint. Cela peut être un nombre maximal d'itérations. Si le critère d'arrêt est atteint, l'algorithme se termine ; sinon, retourner à l'étape 2.

Ces étapes sont répétées jusqu'à ce qu'un critère d'arrêt soit atteint, produisant ainsi une solution approximative au problème d'ordonnancement de tâches.

Algorithme II.1. L'algorithme PSO [58]

Initialisation

Définir la taille de l'essaim S et le nombre de dimensions D

For chaque particule i ∈ [1..S]

Générer aléatoirement X_i et V_i, et évaluer la valeur de fitness de X_i notée $f(X_i)$

Définir $Pbest_i = X_i$ et $f(Pbest_i) = f(X_i)$

End For

Définir $Gbest = Pbest_1$ et $f(Gbest) = f(Pbest_1)$

For chaque particule i ∈ [1..S]

if $f(Pbest_i) < f(Gbest)$ alors

Mettre à jour $f(Gbest) = f(Pbest_i)$

End if

End For

While t < nombre maximum d'itérations

For chaque particule i ∈ [1..S]

Évaluer sa vitesse $V_i(t+1)$ en utilisant l'Équation (II.10)

Mettre à jour la position $X_i(t+1)$ de la particule en utilisant l'Équation (II.9)

if $f(X_i(t+1)) < f(Pbest_i)$ alors

Mettre à jour $Pbest_i = X_i(t+1)$

Mettre à jour $f(Pbest_i) = f(X_i(t+1))$

End if

if $f(Pbest_i) < f(Gbest)$ alors

Mettre à jour $Gbest = Pbest_i$

Mettre à jour $f(Gbest) = f(Pbest_i)$

End if

End for

$t = t + 1$

End while

Return Gbest

11. Conclusion

L'ordonnancement de tâches dans le Fog Computing est un véritable défi en raison de la nature mouvante et complexe de cet environnement. Le Fog se caractérise par la diversité de ses ressources matérielles et logicielles, la variabilité de la charge de travail générée par les appareils connectés, ainsi que la mobilité de ces derniers. Pour optimiser l'utilisation de ces ressources et garantir des performances efficaces, il est essentiel de mettre en place des stratégies de planification des tâches et de gestion des ressources efficaces entre les niveaux Fog. Dans le chapitre suivant, nous aborderons notre proposition pour le problème d'ordonnancement de tâches dans le Fog Computing.

Chapitre III : Proposition et évaluation de performances

1. Introduction

Dans le domaine de l'informatique en brouillard, l'ordonnancement de tâches reste un défi majeur. L'objectif principal est d'assigner efficacement les tâches aux machines virtuelles afin de minimiser le temps d'exécution, assurant ainsi une réponse rapide et satisfaisant les utilisateurs, et réduisant le temps nécessaire pour terminer les tâches.

Dans ce contexte, nous proposons un nouvel algorithme basé sur l'implémentation de la metaheuristique ALO pour résoudre le problème d'ordonnancement dans l'environnement du Fog Computing, puis Afin d'évaluer les performances d'ALO, nous allons réaliser une comparaison avec la métaheuristique PSO.

2. Modélisation du problème de l'ordonnancement de tâches dans le Fog Computing

Dans cette section, nous présentons la modélisation de notre contribution de manière formelle. Notre objectif principal est d'optimiser deux métriques de qualité de service (QoS), le temps de complétion et le makespan, en répartissant efficacement les tâches dans le réseau Fog.

2.1. Les métriques de QoS

- **Le temps de complétion**

Le temps de complétion (CT, Completion Time) est défini comme la somme des temps d'exécution de toutes les tâches dans le système. Il est déterminé par l'équation (III.1).

$$CT = \sum_{i=1}^{n} Texe_i \qquad \text{(III.1)}$$

Où **n** désigne le nombre de tâches, et la valeur de **i** est comprise entre {1, 2, 3, ...,n}, $\boldsymbol{Texe_i}$ désigne le temps d'exécution de la tâche **i**, défini par l'équation (III.2)

$$Texe_{i,j} = \frac{T_{i.length}}{VM_{j.mips}} \qquad \text{(III.2)}$$

$\boldsymbol{T_{i.length}}$ est la taille de la tâche $\boldsymbol{i}$, $\boldsymbol{VM_{j.mips}}$ est la capacité de traitement (en MIPS, Millions d'Instructions Par Seconde) de la machine virtuelle$\boldsymbol{j}$, **j** est l'indice de la machine virtuelle compris entre {1, 2, 3,..., m}, **m** désigne le nombre de machines virtuelles.

- **Makespan**

Le makespan désigne le temps total nécessaire pour achever l'exécution de toutes les tâches. Sa valeur est déterminée par l'équation (III.3)

$$Makespan = max\{Texe\} \qquad \text{(III.3)}$$

Nous précisons que dans ce travail, nous avons traité uniquement le cas de tâches indépendantes.

2.2. La fitness

La fitness, également connue sous le nom de la fonction objectif, évalue la qualité d'un individu en lui attribuant un nombre ou un vecteur. Elle joue le rôle d'une fonction de minimisation qui évalue deux objectifs principaux : le makespan et le temps de complétion. En d'autres termes, la fitness est définie pour mesurer l'optimalité en tenant compte de ces deux critères importants dans le contexte de l'optimisation.

Fitness= a*makespan +b*temps de complétion (III.4)

a et b sont des facteurs de pondération fixés pour souligner l'importance de chacun des deux objectifs évalués (le makespan et le temps de complétion).

D'où : $\begin{cases} a + b = 1 \\ a, b \in [0,1] \end{cases}$

2.3. Problème d'ordonnancement de tâches

Le problème d'ordonnancement de tâches consiste à assigner un ensemble T de N tâches aux différentes machines virtuelles (VM_s) fournis par un service Fog de manière efficace. L'idée est de faire correspondre chaque tâche à une machine virtuelle, et de chercher à réduire le makespan et le temps de complétion.

Le système Fog (Sfog) comprend (N_{pm}) machines physiques (PM), et chaque machine inclut (N_{vm}) machines virtuelles comme indiqué dans le vecteur suivant (III.5).

$$sfog = [MP_1, MP_2\ , \ldots\ldots\ldots MP_{Npm}] \qquad \text{(III.5)}$$

Où MP_i, (i = 1, 2, ...,N_{pm}), désigne la i-ème machine physique utilisée dans le Fog et peut être exprimée comme suit :

$$PM_i = [VM_1, VM_2\ , \ldots\ldots\ldots VM_{Nvm}] \qquad \text{(III.6)}$$

Où VM_k, (k = 1, 2, ..., N_{VM}), désigne la k-ème machine virtuelle, N_{vm} représente le nombre total de machines virtuelles. Les caractéristiques de la VM_k sont déterminées comme suit :

$$VM_k = [IDVM_K, mips_k] \quad \text{(III.7)}$$

Où $IDVM_k$ désigne le numéro d'identification de la k-ème machine virtuelle, et $mips$ représente sa capacité de calcul en millions d'instructions par seconde.

$$T = [T_1, T_2, \dots, T_i, \dots, T_{Ntsk}] \quad \text{(III.8)}$$

Où N_{tsk} représente le nombre de tâches, T_i désigne la i-ème tâche dans la série de tâches qui est caractérisée comme suit :

$$T_i = [IDT_i, Tlength_i, ECT_i] \quad \text{(III.9)}$$

Où IDT_i désigne le numéro d'identification de la i-ème tâche, T_length_i représente la taille de la i-ème tâche et $ECT_{i,j}$ désigne le temps d'exécution prévu pour la i-ème tâche.

La mesure $ECT_{i,j}$ (Expected Complete Time) représente le temps d'exécution nécessaire pour effectuer la tâche T_i dans machine virtuelle VM_j, qui peut être déterminé par la matrice suivante :

$$ECT_{i,j} = \begin{bmatrix} ECT_{1,1} & ECT_{1,2} \dots \dots \dots \dots & ECT_{1,Nvm} \\ ECT_{2,1} & ECT_{2,2} \dots \dots \dots & ECT_{2,Nvm} \\ \vdots & \vdots & \vdots \\ ECT_{Ntsk,1} & ECT_{Ntsk,2} \dots & ECT_{Ntsk,Nvm} \end{bmatrix} \quad \text{(III.10)}$$

3. Motivation

Au fil du temps, les chercheurs ont constamment cherché des moyens efficaces pour résoudre des problèmes extrêmement difficiles. Au départ, ils se sont concentrés sur le développement de méthodes exactes adaptées à des cas particuliers, mais ces approches avaient leurs limitations. L'avènement des heuristiques a ouvert la voie à la découverte de solutions de qualité générale. Plus tard, l'émergence des métaheuristiques a suscité un fort intérêt, offrant des outils puissants pour résoudre une large gamme de problèmes complexes de manière pratique et efficace en proposant des techniques algorithmiques puissantes [59].

Les métaheuristiques jouent un rôle crucial dans l'ordonnancement de tâches car elles permettent de trouver rapidement de bonnes solutions. Leur flexibilité les rend adaptables à

divers types de problèmes et contraintes, et elles se distinguent par leur capacité à explorer efficacement l'espace de recherche tout en améliorant progressivement les solutions existantes. Cette combinaison d'exploration et d'exploitation permet de trouver des solutions optimales de manière pratique et efficace. De plus, elles peuvent être combinées avec d'autres techniques pour encore mieux optimiser les résultats.

En résumé, les métaheuristiques représentent une avancée majeure dans la résolution de problèmes complexes, offrant une solution efficace là où les approches traditionnelles atteignent leurs limites.

4. Application de la métaheuristique ALO pour le problème d'ordonnancement de tâches (Task Scheduling-ALO : TS-ALO)

L'algorithme d'optimisation ALO, inspiré du comportement de chasse des fourmis-lions, est implémenté pour optimiser l'ordonnancement de tâches dans le Fog Computing (TS-ALO), en minimisant le makespan et le temps de complétion. Il utilise une approche de marche aléatoire influencée par les meilleures solutions (fourmilions), favorisant ainsi une exploration efficace de l'espace de recherche. Les fourmilions, représentant les solutions les plus performantes, guident les déplacements des fourmis vers des configurations optimales. Cette méthode permet d'éviter les optimaux locaux et accélère la convergence vers des solutions optimales [60].

Algorithme III.1. L'algorithme TS-ALO pour l'ordonnancement de tâches

nop: la taille de la population ;

Max_iter: nombre d'itérations ;

ub: nombre maximale de machines virtuelles

lb: nombre minimale de machines virtuelles

dim: nombre de tâches ;

Tlength : taille des tâches ;

Vmips : mips des machines virtuelles,

Initialiser la première population de fourmis et de fourmilion aléatoirement ;

Calculer le fitness des fourmis et des fourmi-lions ;

Trouver les meilleurs de fourmi-lions et les considérer comme l'élite (optimum déterminé)

While (i≤ Max_iter)

For chaque fourmi

Sélectionner un fourmilion en utilisant la roulette ;

Créer une marche aléatoire et la normaliser en utilisant les équations (II.1) et (II.3)

Mettre à jour la position de la fourmi en utilisant (II.8)

End For

Calculer la qualité de toutes les fourmis

Remplacer un fourmilion par sa fourmi correspondante si elle devient plus performante (Eq. (II.7))

Mettre à jour l'élite si un fourmilion devient plus performant que l'élite ;

$i = i + 1;$

End while

Return l'élite

5. Réalisation et évaluation de performances

Pour implémenter l'algorithme proposé, nous utilisons le logiciel MATLAB (MathWorks, 2021). Afin d'évaluer les performances de l'algorithme TS-ALO, nous menons une expérience et comparons ses résultats à ceux d'un autre algorithme d'ordonnancement de tâches, l'algorithme TS-PSO (task Scheduling-PSO), pour déterminer celui qui optimise le mieux le makespan et le temps de complétion. Les paramètres expérimentaux sont clairement définis dans le tableau III.1.

Nous proposons deux expériences. Dans la première, nous créons cinq scénarios dans lesquels le nombre de tâches est incrémenté de 100 à chaque scénario. Le nombre de machines virtuelles reste constant à 50 tout au long de l'expérience.

Dans la deuxième expérience, nous créons également cinq scénarios où le nombre de tâches reste constant à 200 tout au long de l'expérience. Cependant, dans cette expérience, nous faisons varier le nombre de machines virtuelles par incréments de 10.

Les résultats obtenus sont présentés dans les deux tableaux III.2, III.3 et sont discutés en profondeur dans la section résultats et discussion.

Expérience / Paramètre	Expérience 1	Expérience 2
CPU vitesse du Processeur	[500,2000] mips	
Taille des tâches	[500 ,20000] mi	
Taille de la population	50	
Nombre d'itération	100	
Nombre de tâches	[100,200,300,400,500]	200
Nombre de machines virtuelles	50	[10,20,30,40,50]

Tableau III. 1.Ensemble de paramètres d'expérimentation

6. Résultats et discussions

Dans les deux Tableaux III.2 et III.3, nous avons présenté les résultats de l'algorithme TS-ALO comparé avec l'algorithme TS-PSO dans les deux expériences, en termes de temps de complétion, de makespan et de fitness pour différentes entrées de données. Les résultats indiquant que l'algorithme TS-ALO a amélioré le temps de complétion et le makespan par rapport à l'algorithme TS-PSO ainsi que la fitness.

6.1 Résultats de la première expérience

Algorithmes / Nombre de tâches	TS-PSO			TS-ALO		
	Temps de complétion	Makespan	Fitness	Temps de complétion	Makespan	Fitness
100	779.96	11.48	376.76	640.2152	10.42	256.1
200	1569.81	13.91	770.91	1313.40	11.65	703.98
300	2390.81	15.99	1234.91	1793.14	12.93	1104.65
400	3152.35	17.84	1509.5	2676.24	13.87	1380.5
500	4015.98	18.86	1813.6	2903.18	15.17	1783.8

Tableau III. 2.Ensemble des résultats obtenus dans la première expérience

Les graphiques suivants illustrent les résultats de tableau III.2

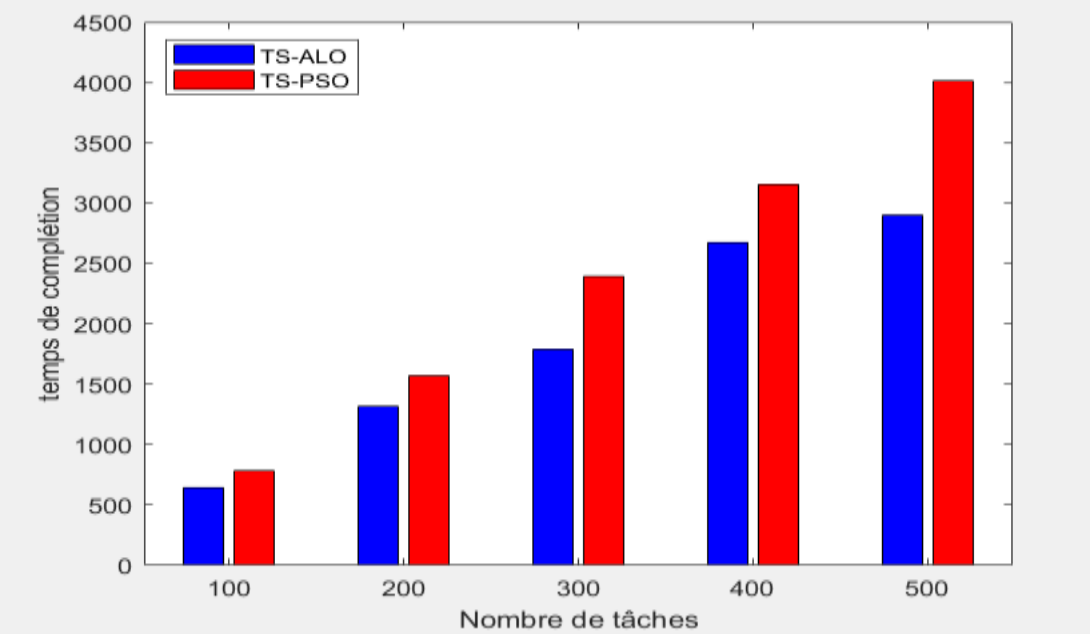

Figure III. 1. Comparaison de temps de complétion de l'algorithme TS-ALO proposé avec l'algorithme TS-PSO

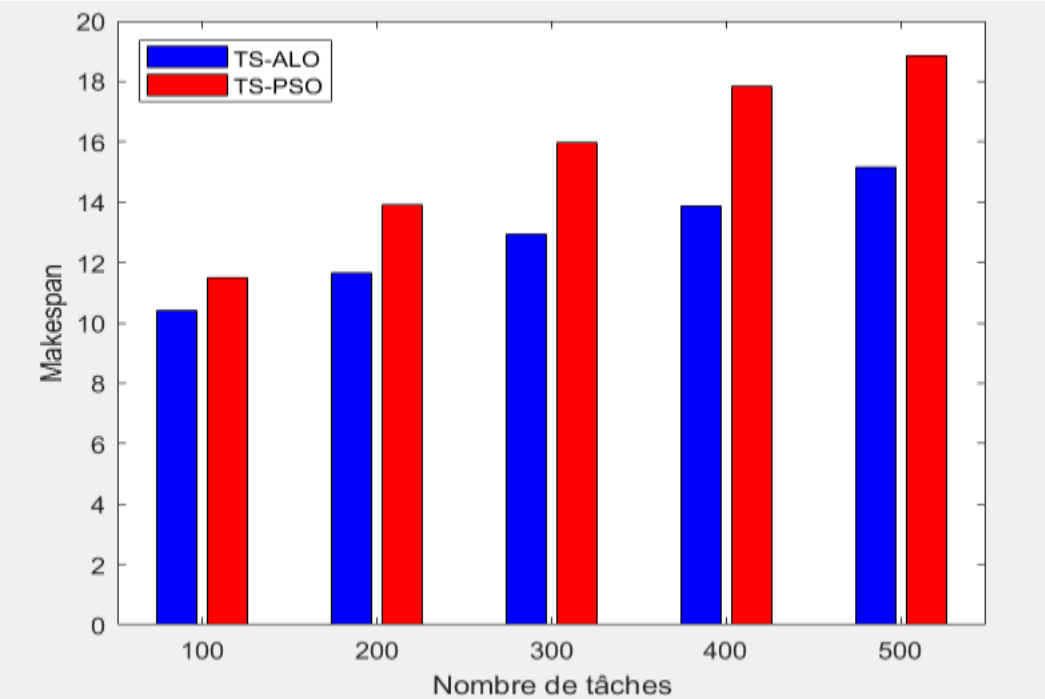

Figure III. 2. Comparaison de makespan de l'algorithme TS-ALO proposé avec l'algorithme TS-PSO

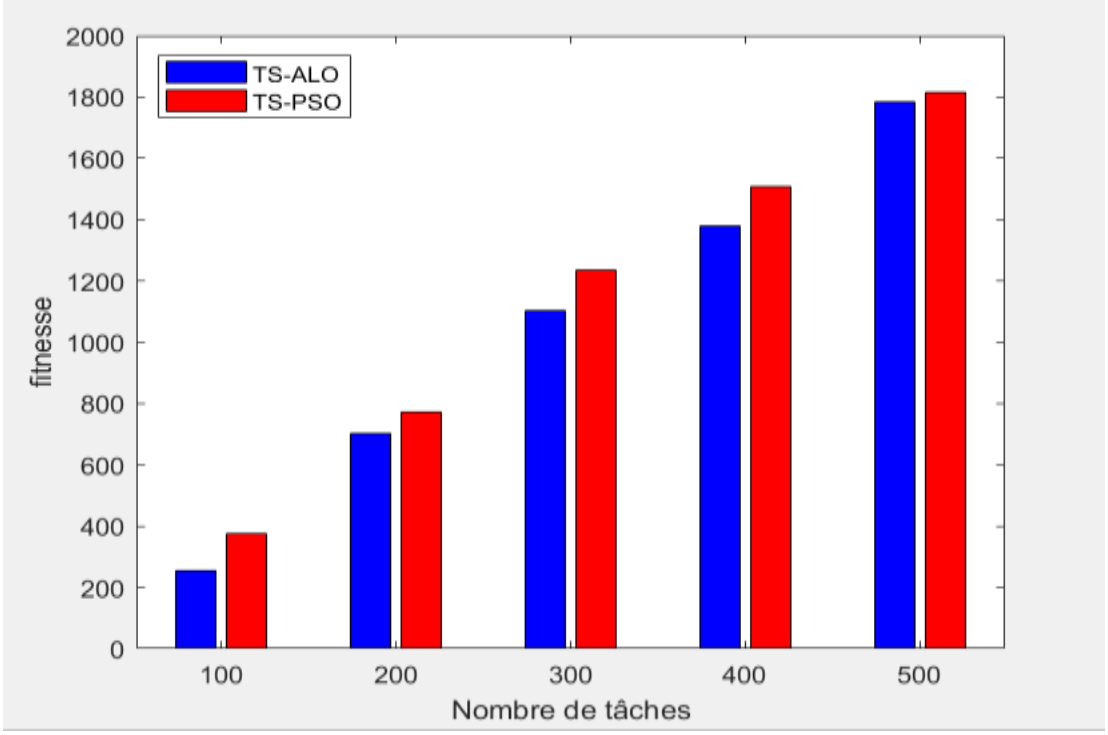

Figure III. 3. Comparaison de fitness de l'algorithme TS-ALO proposé avec l'algorithme TS-PSO

Les résultats de la première expérience montrent que l'exécution de TS-ALO génère systématiquement des valeurs de makespan, de temps de complétion et de fitness inférieures par rapport à celles obtenues de TS-PSO. À mesure que le nombre de tâches augmente, les valeurs de makespan et de temps de complétion augmentent pour les deux algorithmes. Ainsi, dans nos exemples, les performances de TS-ALO sont meilleures que celles de TS-PSO de 17.28% en terme de makespan et de 20.39% en terme de temps de complétion. Cela indique que TS-ALO se distingue comme l'algorithme le plus efficace en termes de temps de complétion et de makespan, tandis que TS-PSO montre des performances moins efficaces.

6.2 Résultats de la deuxième expérience

Algorithmes / Nombre de machines	TS-PSO		TS-ALO	
	Temps de complétion	Makespan	Temps de complétion	Makespan
10	1771	19.27	1616.19	17.27
20	1735.7	18.56	1562.76	16.14
30	1649.6	17.30	1447.5	15.91
40	1476.3	15.86	1311.16	14.95
50	1300.6	14.55	1274.68	13.11

Tableau III. 3. Ensemble des résultats obtenus dans la deuxième expérience

Les graphiques suivants illustrent les résultats de tableau III. 3

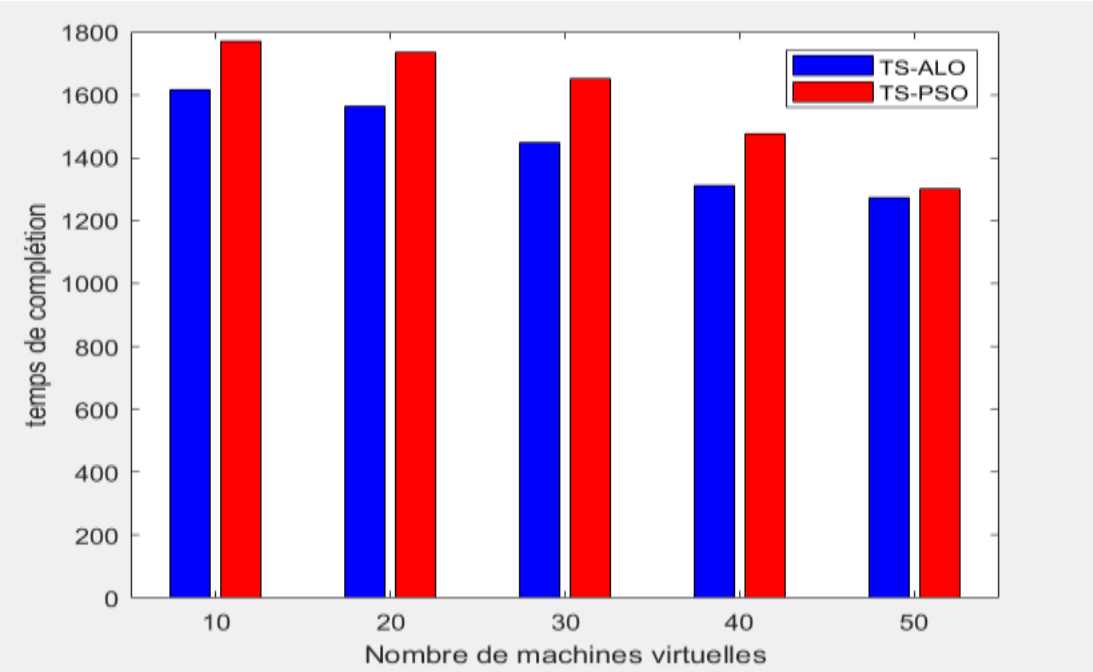

Figure III. 4. Comparaison de temps complétion de l'algorithme TS-ALO proposé avec l'algorithme TS-PSO

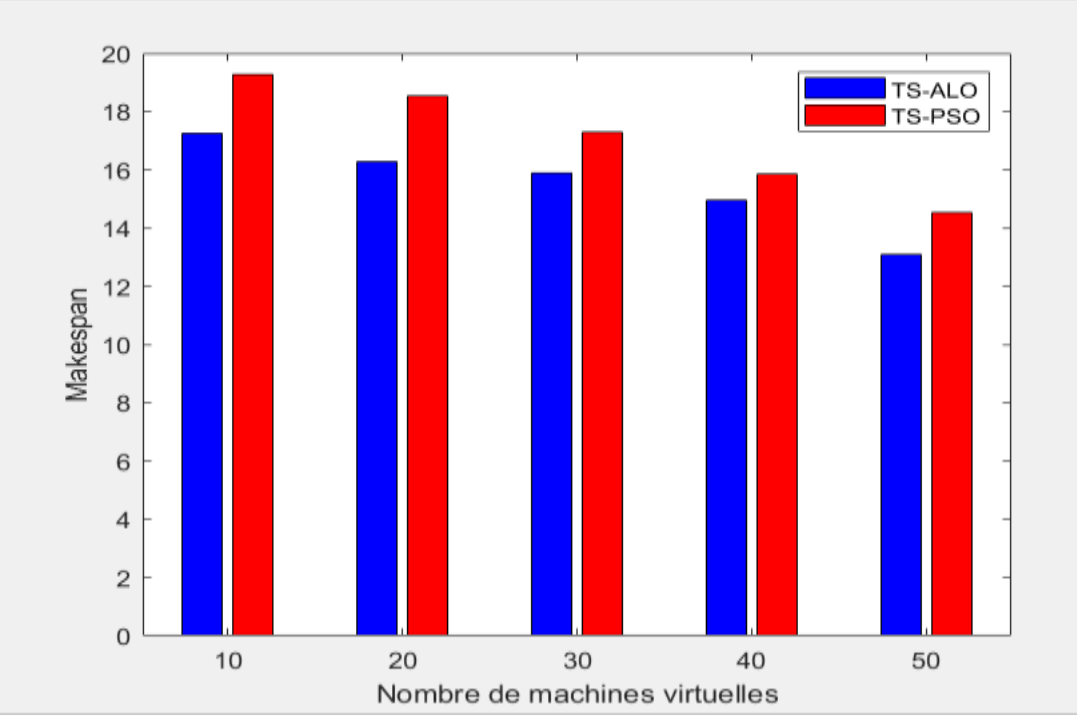

Figure III. 5. Comparaison de makespan de l'algorithme TS-ALO proposé avec l'algorithme TS-PSO

Les résultats de la deuxième expérience, montrent que TS-ALO maintient des performances supérieures à celles de TS-PSO de 9.01% en termes de makespan et de 8.82% en termes de temps de complétion. À mesure que le nombre de machines virtuelles augmente, les valeurs de makespan et de temps de complétion diminuent pour les deux algorithmes. Ainsi, TS-ALO reste plus efficace en termes de temps de complétion et de makespan par rapport à TS-PSO.

7. Synthèse

Après avoir analysé et comparé les différents résultats, nous constatons que les performances de l'algorithme TS-ALO sont toujours meilleures que celles de l'algorithme TS-PSO pour les deux objectifs étudiés (Makespan et temps de complétion). Ainsi, nous pouvons conclure que l'algorithme TS-ALO se révèle être meilleur et plus performant dans ces domaines spécifiques.

8. Conclusion

Au cours de cette dernière étape de notre travail, nous avons présenté notre proposition qui consiste en l'implémentation de l'algorithme ALO pour résoudre le problème d'ordonnancement de tâches dans le Fog Computing (TS-ALO), en tenant compte des deux objectifs (makespan et temps de complétion). Puis nous avons réalisé une comparaison avec l'algorithme TS- PSO.

Conclusion générale

Dans ce projet de fin d'études, nous avons exploré l'une des technologies les plus cruciales émergées ces dernières années dans le domaine informatique : le Fog Computing.

Notre travail s'est concentré sur le problème d'ordonnancement des tâches dans le Fog Computing, visant à minimiser à la fois le makespan et le temps de complétion. Nous avons introduit l'algorithme ALO, une métaheuristique inspirée par les comportements des fourmis, comme un point central de notre étude. Nous avons développé et implémenté cet algorithme spécifiquement pour résoudre le problème d'ordonnancement des tâches dans le Fog Computing (TS-ALO).

Pour évaluer l'efficacité de notre approche, nous avons réalisé une série de simulations et une analyse comparative avec l'algorithme TS-PSO. Ces tests nous ont permis de quantifier et d'analyser les performances respectives de chaque méthode, démontrant ainsi que l'algorithme TS-ALO est plus performant que l'algorithme TS-PSO.

En conclusion, ce projet nous a permis de nous familiariser avec une technologie émergente et cruciale dans le domaine informatique moderne, tout en proposant une approche novatrice pour résoudre un problème complexe d'ordonnancement. Pour les perspectives futures, nous envisageons d'améliorer et de diversifier notre approche en explorant, par exemple, d'autres techniques pour optimiser l'ordonnancement des tâches ou en étudiant l'impact de différents paramètres de performance dans des contextes variés.

Résumé

Ce mémoire se concentre sur l'ordonnancement de tâches dans le contexte du Fog Computing, une technologie intermédiaire entre l'Internet des Objets (IoT) et le Cloud Computing. L'objectif principal est de minimiser à la fois le makespan et le temps de complétion de tâches, essentiel pour optimiser les performances des applications déployées sur des réseaux distribués et hétérogènes.
L'étude introduit l'algorithme ALO, inspiré du comportement des fourmis, comme une métaheuristique efficace pour résoudre ce problème complexe. L'approche proposée, appelée TS-ALO, est développée spécifiquement pour répondre aux défis spécifiques d'ordonnancement dans le Fog Computing.
Pour évaluer l'efficacité de TS-ALO, des simulations détaillées ont été menées, incluant une comparaison approfondie avec l'algorithme TS-PSO. Cette analyse comparative met en évidence les avantages et les domaines d'amélioration potentiels de l'algorithme ALO dans ce contexte particulier.
Mots clés : Fog Computing, Ordonnancement, Optimisation, Métaheuristique, ALO, PSO

Abstract

This paper focuses on task scheduling in the context of Fog Computing, an intermediary technology between the Internet of Things (IoT) and Cloud Computing. The primary objective is to minimize both makespan and task completion time, crucial for optimizing the performance of applications deployed on distributed and heterogeneous networks.
The study introduces the Ant Lion Optimizer (ALO) algorithm, inspired by ant behavior, as an effective metaheuristic for tackling this complex problem. The proposed approach, called task Scheduling-ALO (TS-ALO), is specifically developed to address the specific scheduling challenges in Fog Computing.
To evaluate the effectiveness of TS-ALO, detailed simulations were conducted, including a comprehensive comparison with the task Scheduling Particle Swarm Optimization algorithm (TS-PSO). This comparative analysis highlights the advantages and potential areas for improvement of the ALO algorithm in this particular context.

Keywords: Fog Computing, Scheduling, Optimization, Metaheuristic, ALO, PSO

Références bibliographiques

[1] Sadeeq, M. M., Abdulkareem, N. M., Zeebaree, S. R., Ahmed, D. M., Sami, A. S., & Zebari, R. R. (2021). IoT and Cloud computing issues, challenges and opportunities: A review. Qubahan Academic Journal, 1(2), 1-7.

[2] P.Guillemin , P.Friess, « Internet of things strategic research roadmap », The Cluster of European Research Projects, 2009

[3] T.Lu, W. Neng, « Future Internet: The Internet of Things », Université normale de chine orientale, 2010

[4] P-J.Benghozi, S.Bureau et F.Massit-Folléa, « L'Internet des objets », Orange Ecole polytechnique Telecom Paris Tech, 2008

[5] NASIRI, Mina, et al. « Internet of Things as an enabler in disruptive innovation for sustainability ». 2016

[6] K.Chandrasekaran, « Livre Essentials Of Cloud Computing », décembre 2014

[7] P. Mell, T.Grance, « The NIST Definition of Cloud Computing », National Institute of Standards and Technology Gaithersburg, septembre 2011 https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf

[8] R.Buyya, J.Broberg et A. M. Goscinski, « Cloud Computing: Principles and Paradigms », Publié par : John Wiley et Sons, INC, Publication, Hoboken, New Jersey, 2011

[9] R.P.Padhy, M.R.Patra, « Evolution of Cloud Computing and Enabling Technologies », Institute of Advanced Engineering and Science, 2012

[10]Yoo, S. Christopher, « Cloud Computing: Architectural and Policy Implications », Université de Pennsylvanie Carey, 2011

[11] B.Bhagat, S.Khobragade, A.Arudka, « The Evolution of Cloud Computing », journal of ,networksecurity,2023

[12] Khadidjatou Iman BAMBA, « LES FONDAMENTAUX DU CLOUD COMPUTING», 2014

[13] L.N.SABOUK, M.L.SIDMOU « L'émergence du Cloud Computing au service de la transparence des collaborations inter-organisationnelle et de la confiance numérique: revue de littérature », Revue Internationale des Sciences de Gestion, Université Cadi Ayyad Marrakech ,2022

[14]V.Kherbache, M.Moussalih, Y.Kuhn, A.Lefort,« Cloud Computing »,IUT Nancy Charle magne, 2010

[15] E.H.Bourhim, « La Communication et Le Placement De Conteneurs Docker Dans Le Fog Computing », Université Du Québec À Montréal, 2020

[16] H. REGUIEG. « Cloud Computing: services informatiques dynamiques basés sur le Web-Concepts et notions de base », 2020

[17] H. Atlam, R. Walters, and G. Wills, « Fog computing and the internet of things : A review », Big Data and Cognitive Computing, 2(2), ISSN 2504- 2289. doi : 10.3390/bdcc2020010, 2018

[18] LIU, Qianyu, WEI, Yunkai, LENG, Supeng, et al. « Task scheduling in fog enabled Internet of Things for smart cities ». IEEE 17th International conference on communication technology (ICCT). IEEE. p. 975-980, 2017

[19] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, « Fog computing and its role in the internet of things », in Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, pp. 13–16, 2012

[20] Ai, Y. Peng, M.Zhang, K. « Edge cloud computing technologies for internet of things: A primer ». Digit. Commun. Netw. in press. [CrossRef], 2017

[21] Yi, S. Hao, Z. Qin, Z. Li, Q. « Fog computing: Platform and applications ». In Proceedings of the 3rd Workshop on Hot Topics in Web Systems and Technologies, HotWeb Washington, DC, USA, 24–25 October 2016; pp. 73–78, 2015

[22] Peralta, G.; Iglesias-Urkia, M.; Barcelo, M.; Gomez, R.; Moran, A.; Bilbao, J. Fog « computing based efficient IoT scheme for the Industry » 4.0. In Proceedings of the 2017 . IEEE International Workshop of Electronics, Control, Measurement, Signals and their application to Mechatronics, San Sebastian, Spain, 24–26 May 2017; pp. 1–6

[23] KIM, Hyung-Sin. « Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are White Paper, 2016

[24]F.Bonomi,R. Milito, P. Natarajan, J. Zhu « Fog Computing: A Platform for Internet of Things and Analytics. In Big Data and Internet of Things: A Roadmap for Smart Environments »; Studies in Computational Intelligence; Springer: Cham, Switzerland, Volume 546, pp. 169–186,2014

[25] Aviral Kumar Singhal, Niraj Singhal, « Cloud Computing vs Fog Computing: A Comparative, Study », 2021

[26] HADRI Abdelkader, « Contribution à la résolution des problèmes d'ordonnancement en temps réel dans un système de production de type Job Shop », Université Batna 2 – Mostefa Ben Boulaïd, 2022

[27] J. Carlier and P. Chrétienne, « Problèmes d'ordonnancement : Modélisation, Complexité, Algorithmes », Paris, Masson, 1988

[28] P. Esquirol and P. Lopez, « L'ordonnancement », Paris, Economica, 1999

[29] Pinedo, M.L, « Scheduling: Theory, Algorithms, and Systems ». Springer. https://doi.org/10.1007/978-1-4614-2361-4, 2012

[30] Z. Bouafia, B. Benmammar, M. Hakem, « Algorithmes d'ordonnancement de taches dans un environnement Cloud », Université Abou Bekr Belkaid Tlemcen, Algérie, Institut FEMTO-ST - UMR CNRS Université de Franche-Comté France, 2015

[31] M.Hosseinzadeh, E.Azhir, J.Lansky , S.Mildeova,O. Hassan Ahmed, M. Hussain Malik Et F. Khan « Task Scheduling Mechanisms for Fog Computing:A Systematic Survey », IEEE Access, 2023

[32] M. Hemamalini, « Review on grid task scheduling in distributed heterogeneousenvironment ». International Journal of Computer Applications, 40(2) :24–30,2012.

[33] Pardeep Kumar et Amandeep Verma , « Scheduling using improved geneticalgorithm in cloud computing for independent tasks », In Proceedings of the In-ternational Conference on Advances in Computing, Communications and In- formatics , ICACCI '12, 137–142, New York, NY, USA, ACM , 2012

[34] Liu,chung laung L.J.W, « Scheduling algorithms for multiprogramming in a hardreal-time environment journal of the ACM » (JACM), 1973

[35] M.Bouslama, « Implémentation d'un ordonnanceur embraqué temp réel », Université D'ORAN, 2015

[36] D. E. Insaf, « optimisation d'ordonnancement et d'allocation de ressources dans le cloud computing », thèse de doctorat, 2016

[37] Y. Kyung, « Prioritized task distribution considering opportunistic fog computing nodes »,Sensors, vol. 21, no. 8, p. 2635, 2021

[38] J.-F. Tsai, C.-H. Huang, et M.-H. Lin, « An optimal task assignment strategy in cloud-fog computing environment », Appl. Sci., vol. 11, no. 4,p. 1909, 2021

[39] F. Xu, Z. Yin, A. Gu, Y. Li, H. Yu, and F. Zhang, « Adaptive scheduling strategy of fog computing tasks with different priority for intelligent production lines », Proc. Comput. Sci., vol. 183, pp. 311–317, 2021

[40] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, « Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications », IEEE Trans. Ind. Informat., vol. 17, no. 7, pp. 5068–5076, Jul. 2021

[41] M.Saad, R.N.Enam et R.Qureshi « Optimizing multi-objective task scheduling in fog computing with GA-PSO algorithm for big data application », Frontières dans les Données Massives, 2024

[42] H. T. T. Binh, T. T. Anh, D. B. Son, P. A. Duc et B. M. Nguyen, « An evolutionary algorithm for solving task scheduling problem in cloudfog computing environment », dans les actes du 9ème Symposium sur les Technologies de l'Information et de la Communication (SOICT), Da Nang City, Vietnam, pp. 397–404, 2018

[43] Q.Liu, Y.Wei, S.Leng et Y.Chen, « Task Scheduling in Fog Enabled Internet of Things for Smart Cities », Université des Sciences et Technologies Électriques de Chine, 2017

[44] X. Liu et al, « FogWorkflowSim: an automated simulation toolkit for workflow performance evaluation in fog computing », 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1114–1117, 2019

[45] S.Bansal, H.Aggarwal. « A hybrid particle whale optimization algorithm with application to workflow scheduling in cloud–fog environment ». Decis. Analyt. J.9:100361. doi: 10.1016/j.dajour.100361, 2023

[46] P.Pirozmand, A.A.R.Hosseinabadi, M.Farrokhzad,M. Sadeghilalimi,S.Mirkamali, ,et A. Slowik. « Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing ». Neural computing and applications, 33, 13075 13088.2021

[47] « Metaheuristics: from Design to Implementation », Wiley, 2009

[48] « Handbook of Applied Optimization », Oxford University Press, 2002

[49]Y. C. -. P. Siarry, Optimisation multiobjectif, BLD Saint Germain, 75240 Paris Cedex 05, 2002

[50] Z.AZIZOU,"Découverte et localisation de services dans l'Internet des Objets", Université Akli Mohand Oulhadj de Bouira, 2022

[51] M. Douiri, S. Elbernoussi, and H. Lakhbab. « Cours des méthodes de résolution exactes heuristiques et métaheurstiques ». Université Mohamed V, Faculté des sciences de Rabat, pages 5–87, 2009

[52] M.I.NAAS, « placement des données de l'internet des objets dans une infrastructure de fog », Université de Bretagne Occidentale, 2019

[53] L.Kherbouche,Z.Oubahri, « Quelques méthodes de résolutions en optimisation combinatoire »,Université Mouloud Mammeri, Tizi Ouzou,2017 https://dspace.ummto.dz/items/aaf75b7b-0376-4468-ac4e-b0b5104d916f.

[54] A. Kout, « Contributions à la résolution du problème de routage dans les réseaux mobiles Ad-hoc par les méthodes bio-inspirées ». PhD thesis, Abdelhamid Mehri - Constantine 2, 2017

[55] A.Gherboudj, « Méthodes de résolution de problèmes difficiles académiques », PhD thesis, Université de Constantine, 2013

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

[57] J. Kennedy et R. Eberhart, « Particle swarm optimization », Proc. of IEEE Int'l Conf. on Neural Networks, pp.1942-1948, Piscataway, NJ, USA, 1995

[58] M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh and S. Mirjalili, « Particle Swarm Optimization: A Comprehensive Survey », in IEEE Access, vol. 10, pp. 10031-10061, 2022

[59] M.SEVAUX, « Métaheuristiques: Stratégies pour l'optimisation de la production de biens et de services », Thèse de doctorat. University of Valenciennes, 2004

[60] L.ABUALIGAH et A. DIABAT, « A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments ». Cluster Computing, vol. 24, no 1, p. 205-223, 2021