RÉPUBLIQUE DE CÔTE D'IVOIRE
Union-Discipline-Travail

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Institut National Polytechnique
Félix HOUPHOUËT-BOIGNY de Yamoussoukro

Ecole Doctorale Polytechnique
UMRI 78

THÈSE

En vue de l'obtention du

DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE FELIX HOUPHOUËT-BOIGNY DE YAMOUSSOUKRO

Mention : Informatique et Télécommunications

Spécialité : Réseaux informatiques

THÈME :

Optimisation de l'allocation des ressources *cloud* et impacts sur la qualité d'expérience des services de *cloud gaming*

Présentée et soutenue publiquement le 16/12/2021 par

M. KONÉ Kigninman Désiré

Devant le Jury Composé de :

M. ZOUEU T. Jérémie	Professeur Titulaire, Institut National Polytechnique Houphouët-Boigny, Yamoussoukro – Côte d'Ivoire	**Président**
M. ASSEU Olivier P.	Professeur Titulaire, Institut National Polytechnique Houphouët-Boigny, Yamoussoukro – Côte d'Ivoire	**Co-directeur de thèse**
M. TABBANE Nabil	Professeur Titulaire, Ecole Supérieure des Communications de Tunis, Tunisie	**Co-directeur de thèse**
M. DIABATE Nabongo	Maître de Conférences, Université Alassane Ouattara, Bouaké – Côte d'Ivoire	**Rapporteur**
M. KERMARREC Yvon	Professeur HDR, Institut Mines Télécom Atlantique Bretagne - Pays de la Loire, France	**Rapporteur**
M. MONSAN Vincent	Maître de Conférences, Université Félix Houphouët - Boigny, Abidjan – Côte d'Ivoire	**Examinateur**

DÉDICACE

À ma petite famille.

REMERCIEMENTS

Une thèse est un cheminement que l'on ne parcourt pas seul. Pour cela, avant tout propos, je tiens à remercier tous ceux qui m'ont soutenu et qui ont aidé à l'aboutissement de ce travail.
Ce travail de recherche est le fruit du partenariat de l'École Supérieure des Technologies de l'Information et de la Communication (ESATIC) avec l'École Doctorale Polytechnique (EDP) de l'Institut National Polytechnique Felix Houphouët-Boigny (INP-HB) et l'École Supérieure des Communications de Tunis (SUP'COM Tunis).

Il a été réalisé au sein de l'unité de recherche en Électronique et d'Électricité Appliquée (LEEA : Laboratoire d'Électronique et d'Électricité Appliquée) de l'INP-HB, du laboratoire de recherche sur les réseaux mobiles et le multimédia (MEDIATRON) de SUP'COM Tunis et du Laboratoire des Sciences et Technologies de l'Information et de la Communication de l'ESATIC (LASTIC).
Cette thèse est née de la codirection de M. ASSEU Olivier Pascal, Professeur Titulaire à l'INP-HB de Yamoussoukro, Directeur de la recherche et de l'innovation à ESATIC et de M. Nabil TABBANE, Professeur titulaire à SUPCOM Tunis (Université de Carthage).
Nous voudrions donc leur exprimer notre profonde reconnaissance d'avoir accepté de diriger cette thèse. Nous les remercions également pour leurs disponibilités, leurs conseils, leurs patiences, leurs encouragements et leurs orientations tout au long de ce projet. Ils ont cru en moi malgré les contraintes et difficultés rencontrées tout au long de mon parcours doctoral.

Nous tenons particulièrement à dire merci au Professeur Sofiane CHERIF, Directeur de SUP'COM Tunis pour avoir facilité notre stage de recherche à SUP'COM au sein du laboratoire MEDIATRON. C'est l'occasion pour nous d'exprimer notre reconnaissance aux membres du laboratoire MEDIATRON pour leur amitié, leur soutien, leur accueil dans le groupe et les collaborations scientifiques.

Nous sommes reconnaissants envers M. Adama KONATE, Directeur Général de l'ESATIC et M. SORO Etienne, Directeur de la pédagogie à l'ESATIC pour leur implication et les encouragements. Nous exprimons notre reconnaissance envers nos collègues à l'ESATIC en particulier à Dr Aliou BAMBA pour sa contribution à l'implémentation de nos algorithmes et à Dr BROU Pacôme pour son assistance.

Nous exprimons notre reconnaissance à M. YAO Benjamin, Professeur Titulaire et Directeur de l'Ecole Doctorale de l'Institut Polytechnique Felix Houphouët-Boigny de Yamoussoukro (EDP). Nous remercions M. ZOUEU Jérémie, Professeur titulaire et M. HABA Cissé, Professeur titulaire, respectivement responsable et responsable adjoint de l'UMRI 78 pour l'organisation des activités de recherches au sein de notre laboratoire. Aussi, j'exprime ma profonde gratitude à tous les différents membres de mon comité de thèse pour leurs commentaires utiles pour faire avancer ce travail.

Nous exprimons aussi notre reconnaissance à Dr ZAMBLE Raoul, enseignant-chercheur à l'ESATIC pour ses observations et pour sa contribution à la publication d'un article scientifique.

A Dr Eya DHIB, Chercheure post-doc au sein du laboratoire MEDIATRON, nous exprimons notre profonde gratitude pour son encadrement, son aide précieuse pour la conception de certains algorithmes que nous avons mis en place et pour toutes les séances de travail pour faire avancer cette thèse.

Nos reconnaissances aussi vont à l'endroit de notre collègue SILUE Mariame épouse COULIBALY et à ma mère Nahoua OUATTARA pour leurs contributions aux corrections des fautes d'orthographe, de grammaire et de syntaxe.

Enfin, nous tenons à remercier tous nos parents, frères, sœurs et amis pour leurs encouragements et leur soutien indéfectible durant toutes ces années de recherches.

A tous les membres du jury, nous adressons notre reconnaissance pour l'honneur qu'ils nous font d'être présents pour l'évaluation de ce travail. Ce sont :

- Professeur ZOUEU Jérémie, Président
- Professeur ASSEU Olivier Pascal, Co-directeur de thèse
- Professeur TABBANE Nabil, Co-directeur de thèse
- Professeur DIABATE Nabongo, Rapporteur
- Professeur KERMARREC Yvon, Rapporteur
- Professeur MONSAN Vincent, Examinateur

RÉSUMÉ

L'industrie des jeux vidéo est devenue très lucrative avec la tendance actuelle à tout transformer en services *cloud*, créant ainsi un nouveau concept appelé *cloud gaming*.

Le *cloud gaming* est un service de jeu à la demande qui libère les appareils des utilisateurs finaux de toute complexité liée à l'informatique, au rendu de graphiques ou au stockage de données ou d'états de jeu. Il combine les avantages du concept de *cloud computing* et du jeu en ligne. Cette nouveauté permet aux utilisateurs de dépasser les limites matérielles des terminaux et leur offre la possibilité de bénéficier de services de jeux gourmands en ressources.

Cependant, bien qu'il soit largement déployé, la qualité de service (QoS) des jeux en ligne actuels n'est pas satisfaisante, en raison des demandes variables des utilisateurs et des exigences strictes en matière de temps de réponse.

Étant donné que ces services sont très sensibles à la latence qui diffère selon la catégorie de jeu, le fournisseur de services de *cloud gaming* doit garantir une qualité d'expérience (QoE) acceptable pour survivre sur le marché. Une des solutions classiques les plus répandues est de surdimensionner l'infrastructure. Pour ce faire, le fournisseur de services de jeux est invité à allouer des ressources en excédent dans le *cloud* pour prendre en charge les périodes de pics de charges. Bien que cette stratégie soit simple, il y a un risque de mauvaise gestion des ressources allouées. En période de baisse d'occupation, les locataires doivent payer pour les ressources inutilisées, ce qui augmente inutilement les coûts d'affectation. En outre, les services de jeux ont parfois des pics de charge inattendus qui nécessitent d'autres allocation de ressources, la migration des ressources existantes vers d'autres *datacenters* ou leurs reconfigurations.

Cette thèse étudie des approches d'allocation de ressources pour améliorer la QoE des services de *cloud gaming* et réduire les coûts d'exploitation des fournisseurs. Trois approches sont présentées pour le processus d'allocation des ressources *cloud* pour les jeux en ligne afin, d'allouer les ressources de manière optimale, d'améliorer la QoE des utilisateurs et de réduire les coûts au niveau du fournisseur.

Dans la première contribution, est proposé une heuristique pour l'allocation de ressources afin d'améliorer la QoE des utilisateurs. Notre algorithme d'optimisation hybride nommé RHSA a permis d'améliorer les métriques de la QOE comme l'équité, le MOS et la perte d'expérience des joueurs.

Dans la deuxième contribution, nous avons pris en compte le critère de la consommation énergétique afin d'allouer les ressources plus efficacement. Cette énième approche d'allocation de ressources nommé FRHSA est une amélioration de notre précédente contribution RHSA avec le concept du calcul fractionnaire. Elle permet de réduire la consommation énergétique des *datacenters* et ainsi les coûts d'exploitation des fournisseurs.

Face à la variation soudaine des charges des jeux en ligne, nous avons dans la troisième contribution, proposé une approche d'allocation proactive. Cette approche d'allocation basée sur un modèle prédictif de la charge future de service de jeux nommé FRDL, nous permet d'éviter la dégradation de la QoE pendant les variations des charges de travail. La QoE des joueurs se trouvent ainsi améliorer avec des meilleurs résultats au niveau des métriques.

Mots clés: *cloud computing, cloud gaming, resource allocation, Massively Multiplayers Online Gaming, Quality of experience (QoE), Optimization, fractional calculus, spark architecture, harmony search algorithm, rider optimization algorithm, fairness, LSTM, Workload Prediction*.

ABSTRACT

The video game industry has become very lucrative with the current trend to turn everything into cloud services, creating a new concept called cloud gaming.

Cloud gaming is an on-demand gaming service that frees end-user devices from the complexities of computing, rendering graphics, or storing game data or states. It combines the benefits of cloud computing and online gaming. This allows users to overcome the hardware limitations of terminals and gives them the opportunity to benefit from resource-intensive gaming services.

However, although widely deployed, the quality of service (QoS) of current cloud gaming services is not satisfactory, due to varying user demands and strict response time requirements.

Since these services are very sensitive to latency, which differs depending on the game category, the cloud gaming service provider must ensure an acceptable QoE to survive in the market. One of the most common solutions is to over-provision the infrastructure. To do this, the gaming service provider is encouraged to allocate excess resources in the cloud to support peak load periods. While this is a simple strategy, there is a risk of mismanaging the allocated resources. During periods of declining occupancy, tenants must pay for unused resources, unnecessarily increasing allocation costs.

In addition, gaming services sometimes have unexpected load peaks that require further resource allocation, migration of existing resources to other datacenters or their reconfiguration.

This thesis investigates resource allocation approaches to improve the QoE of cloud gaming services. Three approaches are presented for the process of allocating cloud resources for online games to optimally allocate resources and thus improve the QoE of users.

In the first contribution, a heuristic for resource allocation to improve user QoE is proposed. Our optimization algorithm named RHSA has improved QoE metrics such as fairness, MOS and loss of player experience.

In the second contribution, we considered the energy consumption parameter to allocate resources more efficiently. This umpteenth resource allocation approach named FRHSA is an improvement of our previous RHSA contribution with the concept of fractional computation. It helps to reduce the energy consumption of datacenters and thus the operating costs of suppliers.

Faced with the sudden variation of online game loads, we proposed in the third contribution a proactive allocation approach. This allocation approach, based on a predictive model of the future game service load called FRDL, allows us to avoid QoE degradation during workload variations. The QoE of the players is thus improved with better values of the metrics.

Keywords: cloud computing, cloud gaming, resource allocation, Massively Multiplayers Online Gaming, Quality of experience (QoE), Optimization, fractional calculus, spark architecture, harmony search algorithm, rider optimization algorithm, fairness, LSTM, Workload Prediction.

PUBLICATIONS SCIENTIFIQUES

Les travaux de recherches que nous avons menés dans cette thèse ont fait l'objet de la rédaction de trois (03) articles scientifiques publiés dans des revues internationales reconnues par le CAMES (voir annexe) :

- **Article 1 (indexé par SCOPUS / EI-COMPENDEX, IF = 0,87)**

Désiré, K. K., Dhib, E., Tabbane, N., & Asseu, O. (2021*). QoS and QoE aware multi objective resource allocation algorithm for cloud gaming*. Journal of High Speed Networks, vol. 27, no.2, pp. 121-138,

DOI: 10.3233/JHS-210655

- **Article 2 (indexé par SCOPUS / EBSCO/ COMPENDEX/ ESCI, IF = 1,14)**

Désiré, K. K., Dhib, E., Tabbane, N., & Asseu, O. (2021). *Energy and Quality Aware Multi-Objective Resource Allocation Algorithm in Cloud*. Journal of Information & Knowledge Management, 20(04), 2150052.
DOI URL: https://doi.org/10.1142/S0219649221500520

- **Article 3 (indexé par CASSI, IF = 0,89)**

Désiré, K. K., Francis, K. A., Kouassi, K. H., Dhib, E., Tabbane, N., & Asseu, O. (2021). *Fractional Rider Deep Long Short-Term Memory Network for Workload Prediction-Based Distributed Resource Allocation Using Spark in Cloud gaming*. Engineering, 13(3), 135-157.

DOI : 10.4236/eng.2021.133011

AUTRE CONTRIBUTION D'ARTICLE :

- Zamble Raoul, Kone Kigninman Desire, Gbegbe Raymond and Bamba S. Seyndou (2020); *Population exposure to radiation emitted by mobile networks in Abidjan, cote divoire* Int. J. of Adv. Res. ***8*** *(Nov). 729-740* (indexé **par CASSI**, IF=7,08)

DOI URL : http://dx.doi.org/10.21474/IJAR01/12063

TABLE DES MATIERES

LISTE DES TABLEAUX

LISTE DES FIGURES

GLOSSAIRE DES ACRONYMES

ARIMA	Auto Regressive and Moving Average
CAMES	Conseil Africain et Malgache pour l'Enseignement Supérieur
CO2	Dioxyde de carbone
DC	DataCenters
EC2	Elastic Compute Cloud.
FC	Fractional calulus calcul fractionnaire
FPS	Frame per Seconde
FRDL	Fractional Rider Deep LSTM
FRHSA	Fractional Rider-HSA
Gaas	Gaming-as-a-service
HSA	Harmony Search Algorithm
IT	Information Technology
LSTM	Long short-term memory
MMOG	Massively Multi-players Online Gaming
NIST	National Institut of Standards and Technologies
PM	Physical Machine
QoE	Quality of Experience
QoS	Quality of service
RHSA	Rider based HSA
RNN	Recurrent Neural Network
ROA	Rider Optimization Algorithm
SDN	Software Defined Network

SLA Service Level Agreement

SLO Service-level objectives

UPL User Preference Level

USA Etats Unis d'Amérique

VM Virtual Machines, machines virtuelles en français

VMs Virtual Machines

WoW World of Warcraft

XaaS X-as-a-Service

LISTE DES ANNEXES

INTRODUCTION GENERALE

Avec l'évolution et la prolifération d'Internet de nos jours, l'accès à l'information, aux services et aux ressources informatiques se fait de manière différente. En effet, avec l'utilisation massive des terminaux mobiles, tout est aujourd'hui, fourni sous forme de service. Ainsi, les utilisateurs dépendent lourdement d'Internet et exigent de plus en plus un accès à leurs données n'importe où, n'importe quand et à partir de n'importe quel équipement. Ils désirent exécuter leurs applications et avoir accès à leurs documents en utilisant des terminaux qui n'ont pas nécessairement de grandes capacités de calcul. Pour cela, ils s'orientent vers un nouveau paradigme connu sous le nom de *cloud computing.*

Le *cloud computing* est un paradigme d'exécution des services informatiques [1]. Il se base sur la déportation des calculs informatiques du prestataire en vue de répondre aux besoins de leurs clients. Ce concept bénéficie de la virtualisation des tâches pour ajouter une notion d'élasticité et un modèle de facturation à la demande. En termes simples, il s'agit d'une nouvelle façon d'utiliser les services *cloud* de manière transparente, dynamique, efficace et rentable.

En raison de la fourniture de ressources élastiques et à la demande, dans le *cloud computing*, de nombreuses entreprises informatiques ont transféré leurs services vers le *cloud*. Aujourd'hui, les services des réseaux sociaux les plus populaires du moment, tel Facebook, les services de courrier électronique, comme Gmail, les services de bureau en ligne, à l'instar de Microsoft Office Online, et le service de jeu à la demande, OnLive, sont tous hébergés sur le *cloud* au sein de vastes *datacenters,* où un ensemble de serveurs partagés est géré pour fournir des ressources de calcul, de communication, et de stockage à la demande et de manière évolutive. Alors pour fournir efficacement des ressources, des techniques de virtualisation ont été appliquées pour regrouper les ressources de calcul en machines virtuelles (VM). A cet effet, les services s'exécutent sur des VMs (machines virtuelles, *virtual machines* en anglais) qui, elles-mêmes, partagent les ressources physiques au sein des *datacenters* géographiquement distribués. En gérant les VMs, le *cloud computing* est capable de fournir ou de libérer les ressources avec une granularité fine pour répondre aux demandes de services [2].

Avec la popularité des appareils mobiles, les services multimédias en ligne comme le *cloud gaming* sont aujourd'hui très populaires et offrent d'immenses opportunités, grâce entre autres aux principes de services à la demande et au modèle économique de location de services (principe du *pay-as-you-go*).

Le *cloud gaming* permet de reposer la puissance de calcul non pas sur la console de jeu mais plutôt sur des serveurs hébergés dans de vastes *datacenters*. Ce qui permet de jouer à des jeux vidéo sophistiqués depuis un simple PC, une tablette ou tout autre écran, à condition d'avoir une connexion internet à très haut débit. Actuellement, les plateformes de *cloud gaming* rivalisent entre elles, et sont désormais les concurrentes principales des plus grands marchés des consoles comme Xbox, PlayStation4, etc. [3].

Par rapport aux services en ligne classiques, les services de jeux en ligne exigent généralement des calculs intensifs et une bande passante élevée. En outre, la charge de travail dans les services des jeux en ligne varie très rapidement.

Afin de répondre à l'explosion de la charge de travail, les fournisseurs de *cloud gaming* ont habituellement tendance à surdimensionner les ressources pour garantir la qualité de service (QoS). Cependant, le surdimensionnement est à la fois coûteux et inefficace, et entraîne une faible utilisation des ressources dans les périodes sans pics et un coût élevé des ressources.

L'émergence du *cloud* offre une solution efficace pour les services des jeux en ligne. L'exécution des jeux en ligne sur le *cloud* permet aux utilisateurs de ne plus avoir à mettre à jour leur matériel et à installer des logiciels. Au niveau des jeux en ligne, les tâches de calcul intensif sont exécutées sur les serveurs *cloud*, ce qui réduit considérablement les exigences matérielles du côté de l'utilisateur.

En configurant dynamiquement les ressources *cloud*, les fournisseurs de services de jeux en ligne peuvent améliorer l'utilisation des ressources. Par conséquent, un nombre croissant de fournisseurs de jeux vidéo ont migré vers *cloud*.

Mais, ces services posent de nouveaux problèmes au *cloud*. La quantité de ressources allouées aux applications multimédias hautes performances telles que le *cloud gaming* continue de croître dans les *datacenters* privés et publics. La forte demande et les modèles d'utilisation de ces plates-formes font de l'allocation intelligente de ces ressources une priorité à l'efficacité des *clouds* publics et privés.

Le temps de réponse est l'un des principaux facteurs de qualité d’expérience dans les services de *cloud gaming*. La charge de travail variable de ces services provoque des pics soudains de demande de calcul et allongent par conséquent le délai de réponse [4]. Il est donc nécessaire de développer une méthode d’allocation dynamique des ressources pour répondre aux exigences de temps de réponse en cas de charge de travail variable.

Aussi, existe-t'il différents types de jeux, qui ont des demandes de ressources et des exigences de qualité de service hétérogènes rendant l'allocation des ressources plus complexe. Les VMs sont donc configurées avec des capacités de ressources différentes. C'est un défi de satisfaire toutes les exigences de QoS en allouant des VMs à chaque type de jeux de manière équitable. Par conséquent, les fournisseurs de *cloud gaming* ont besoin d'un modèle d'allocation de ressources efficace pour configurer de manière optimale les ressources.

Outre le temps de réponse, une autre préoccupation importante pour les fournisseurs de service *cloud gaming* est le coût de ces ressources. Les fournisseurs de service des jeux en ligne louent un certain nombre de VMs auprès des fournisseurs de services *cloud*. Si les VMs louées sont plus nombreuses que les demandes réelles, les ressources ne peuvent pas être pleinement utilisées, ce qui entraîne un gaspillage inutile. Inversement, si les ressources allouées sont inférieures aux demandes réelles, le temps de réponse augmente, ce qui entraine la dégradation de la qualité d'expérience des joueurs.

En outre, une bonne allocation de ressources peut potentiellement réduire le coût d'exploitation par le biais d'une réduction de la consommation énergétique au sein des *datacenters*. Et également le coût des utilisateurs, car le modèle de tarification des services est le paiement à l'utilisation [5]. Par conséquent, il est nécessaire de mettre en place un mécanisme d'allocation des ressources économe pour fournir des services à un faible coût.

Compte tenu des problèmes susmentionnés, il est urgent de **mettre en place un système efficace d'allocation des ressources**. Pour cela, nous étudions l'allocation des ressources pour les services de *cloud gaming* dans le cadre de notre projet de recherche.

Dans ce contexte, en plus de fournir une expérience de haute qualité aux joueurs (QoE), un fournisseur de services de *cloud gaming* doit allouer les ressources *cloud* de manière efficace afin de réduire ses coûts d'exploitation et faire face à la concurrence du marché.

Dans cette thèse, nous visons à allouer de manière optimale les ressources *cloud* pour les services de *cloud gaming* afin de maximiser la qualité d'expérience du point de vue des utilisateurs et de minimiser le coût d'exploitation du point de vue des fournisseurs.

Dans le *cloud gaming*, les problèmes d'allocation de ressources représentent donc un défi important [6]. C'est la raison pour laquelle, la question générale qui guide cette thèse est la suivante : **Comment allouer efficacement les ressources *cloud* pour les services de *cloud***

***gaming*, tout en minimisant le coût d'exploitation des fournisseurs et en maintenant une qualité d'expérience (QoE) acceptable pour le joueur ?**

Pour répondre à cette problématique, nous nous proposons de répondre aux préoccupations qui suivent. Dans un premier temps, chercher à savoir lequel des mécanismes d'allocation mettre en place afin d'assurer un partage efficace et équitable des ressources *cloud* en améliorant la QoE des joueurs. En outre, s'intéresser à la stratégie d'allocation de ressources à mettre en œuvre pour minimiser la consommation d'énergie globale sans altérer le niveau de QoE des joueurs. Pour finir, aborder la prédiction de charge qui contribue à allouer les ressources *cloud* plus efficacement en évitant des dégradations de la QoE lors des pics de charge.

A ce propos, le mémoire de thèse se subdivisera principalement en quatre chapitres. Le premier chapitre présente les généralités et l'état de l'art. D'abord, elle donnera un aperçu général du paradigme du *cloud computing*. Ensuite, décrira dans les détails le service de jeu en ligne, nommé *Cloud gaming* ou *Gaming-as-a-service* (Gaas) et sa sensibilité au délai de réponse. Enfin présentera un état de l'art des différents travaux sur le *cloud gaming*.

Après cette étude, le deuxième, troisième et quatrième chapitre seront consacrés à nos différentes contributions pour l'amélioration de l'allocation des ressources *cloud* pour les jeux en ligne. D'abord, nous proposerons dans le deuxième chapitre un algorithme hybride d'optimisation pour l'allocation des ressources avec trois critères. Ensuite, dans le deuxième chapitre en ajoutant la contrainte énergétique, sera proposée une amélioration de cet algorithme. Enfin, dans le quatrième chapitre, sera mis en exergue une méthode d'allocation basée sur la prédiction de la charge de travail. Nos approches sont basées sur une adaptation de la théorie des jeux.

Une conclusion générale des approches proposées et la discussion d'orientations futures termineront ce mémoire de thèse.

1 CHAPITRE 1 : Généralités & État de l'art

Sommaire

1.1 Introduction

Le *cloud computing* est un paradigme d'exécution des services informatiques [1]. Ce concept exploite la virtualisation des tâches pour ajouter une notion d'élasticité et un modèle de facturation à la demande. Ce chapitre présente l'ensemble des concepts clés abordés dans la thèse. Il nous donne un aperçu général du paradigme du *cloud computing* et de l'allocation des ressources. Ensuite, aborde les notions de problèmes d'optimisation et de QoE. Et décrira plus en détails le service *Cloud* de jeux en ligne, nommée *Cloud gaming* ou *Gaming-as-a-service* (Gaas) avec le service très populaire du jeu MMOG (Massively Multi-players Online Gaming) et sa sensibilité au délai de réponse.

1.2 Cloud Computing

1.2.1 Concept

L'idée du *cloud* a émergé dans les années 1960 lorsque le professeur John McCarthy avait imaginé que les ressources informatiques seront fournies comme des services d'utilité publique [7]. Plus tard, à la fin des années 1990, ce concept a pris de l'importance avec l'avènement du *Grid Computing* [8]. Le terme *Cloud* est une métaphore exprimant une similitude avec le réseau électrique, dans lequel l'électricité est générée dans de grandes centrales électriques, puis distribuée aux utilisateurs finaux via un réseau.

Ici, les grandes centrales sont les datacenters, le réseau est principalement Internet et l'électricité correspond aux ressources de calcul. Le *cloud computing* n'est vraiment apparu qu'en 2006 [9] avec l'arrivée d'Amazon EC2. L'explosion du *Cloud* s'est produite avec l'entrée de sociétés comme Google, Microsoft, IBM, Sun et Canonical Ltd en 2009.

D'après une étude menée par le cabinet Forrester, le marché du *cloud computing* dominé par Amazon Web services, Microsoft Azure, google *cloud*, Alibaba *cloud* progresse et devrait atteindre 410,8 milliards de dollars en 2022 comme l'illustre la figure 1.1.

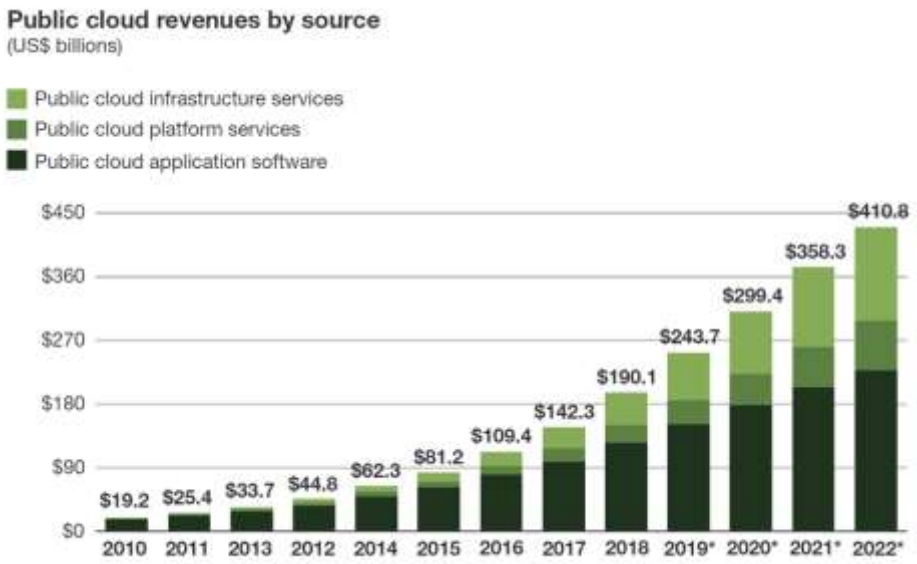

Figure 1. 1 Prévision du marché public du *cloud* [10]

Le *cloud computing*, de plus en plus populaire, permet d'offrir de nouveaux modèles de services Internet. Ainsi, pour comprendre son évolution, il est nécessaire d'avoir une compréhension de ses concepts de base.

1.2.2 Définition

Plusieurs définitions du *cloud computing* ont été données [11] [12] [13]. La raison principale de l'existence de plusieurs définitions est le fait qu'il ne s'agit pas d'une nouvelle technologie, mais plutôt d'un nouveau modèle qui utilise un ensemble de technologies existantes pour répondre à des exigences économiques. La plupart des technologies utilisées par le *cloud*, notamment la virtualisation, ne sont pas nouvelles [14]. Pour éliminer la confusion par rapport à la définition, des organismes de standardisation ont proposé des définitions formelles pour le *Cloud*.

D'après le National Institut of Standards and Technologies (NIST) [15], le *cloud computing* est défini comme *"un modèle qui permet un accès réseau à la demande, pratique et omniprésent, à un pool partagé de ressources informatiques configurables (réseaux, serveurs, stockage, applications, etc.) qui peuvent être rapidement réservées et libérées avec un effort de gestion ou une interaction minimale avec les fournisseurs de services"*.

La définition ci-dessus est souvent expliquée brièvement comme un accès sur demande au réseau, avec une élasticité, rapidité et facilité de gestion des ressources puissantes. En effet, les ressources informatiques sont partagées et leurs puissances de calcul sont configurables en fonctions des besoins. De plus, les utilisateurs *Cloud* peuvent bénéficier d'une flexibilité

importante avec un effort minimal de gestion. NIST ajoute que le *cloud computing* n'est pas une nouvelle technologie, en elle-même, mais une nouvelle façon d'allouer les ressources informatiques. Cette allocation de ressources est typiquement exploitée par un paiement à l'utilisation avec une garantie personnalisée du niveau de continuité de service SLA (*Service Level Agreement*) de la part des fournisseurs.

1.2.3 Principe

Le *cloud computing* consiste en l'utilisation de serveurs délocalisés pour exécuter des services, avec une quantité de travail évaluée et facturée de manière dynamique par le gestionnaire du *Cloud*. Il ne s'agit plus de payer pour la location d'un serveur physique permettant d'héberger des services, mais pour une certaine qualité d'hébergement et une infrastructure adaptable au besoin.

1.2.4 Caractéristiques principales

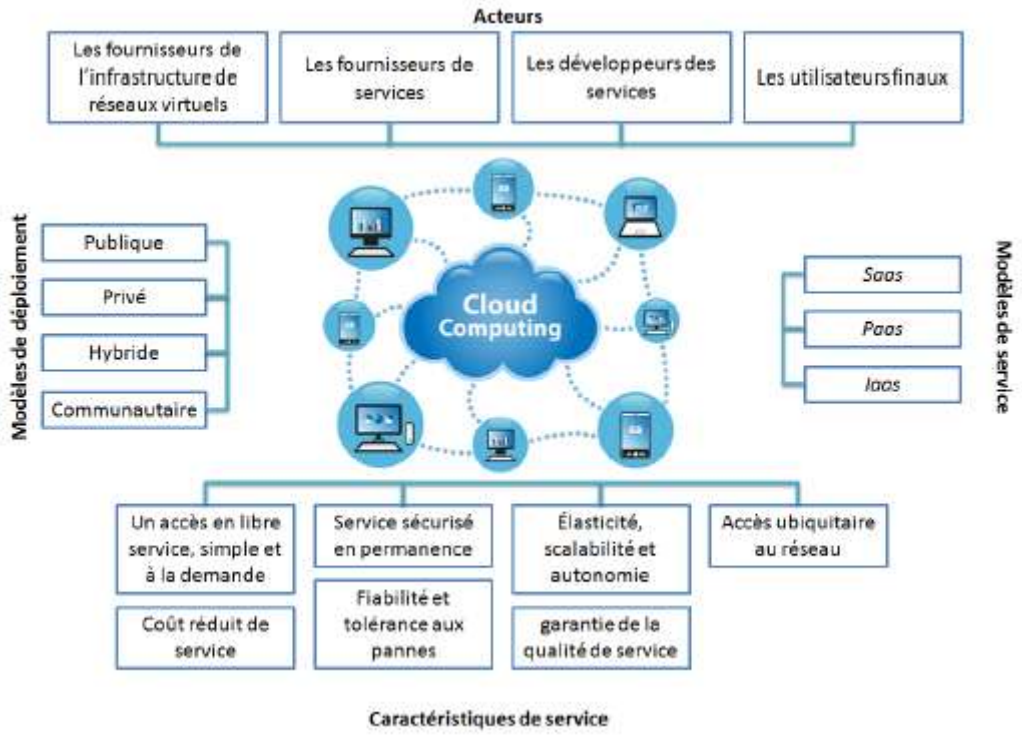

Figure 1. 2 Caractéristiques du *cloud computing* [16]

Comme décrit dans la figure 1.2, le *cloud computing* se caractérise essentiellement par [15]:

Un accès en libre-service, simple et à la demande : les services et les ressources sont offerts aux utilisateurs, sur demande, automatiquement sans intervention humaine. Le *cloud computing* simplifie les usages en permettant de s'affranchir des contraintes de l'outil informatique

traditionnel (installation et mise à jour des logiciels, espace de stockage, portabilité des données, etc.). Le *cloud computing* offre également plus d'élasticité et d'agilité, car il permet un accès plus rapide aux ressources informatiques (serveur, stockage ou bande passante) via un simple portail web et donc sans investir dans du matériel supplémentaire. Par conséquent, il est disponible immédiatement. Libéré de toute tâche de maintenance ou de sécurité, le client *cloud* peut se concentrer prioritairement sur son travail et son activité.

Un accès ubiquitaire au réseau : les capacités sont disponibles sur le réseau et accessibles via des mécanismes standards qui facilitent l'accès au service à partir de différents terminaux (ordinateurs portables, tablettes, smartphones, etc.).

Élasticité, scalabilité et autonomie : les capacités des ressources demandées peuvent être rapidement augmentées ou diminuées, automatiquement reconfigurées, orchestrées et consolidées, selon les besoins et de manière transparente pour l'utilisateur. Les utilisateurs n'ont généralement aucun contrôle ou connaissance de l'emplacement exact des ressources en cours de provisionnement. Cependant, ils peuvent exiger que l'emplacement soit spécifié à un niveau d'abstraction plus élevé (par exemple le pays où le datacenter est implanté).

Fiabilité et tolérance aux pannes : les environnements *Cloud* profitent de la redondance intégrée du grand nombre de serveurs qui les composent et permettent une haute disponibilité et fiabilité des applications qui peuvent en bénéficier[17]. Les contrats SLA qui définissent la continuité de service pour les applications *cloud* sont également respectés.

Garantie de qualité de service (QoS) : les *cloud* sont gérés dynamiquement selon des accords de Service Level Agreement (SLA)[17] entre le fournisseur et l'utilisateur. Le SLA définit des directrices telles que les paramètres de livraison, les niveaux de disponibilité, la maintenabilité, les performances, le fonctionnement ou d'autres attributs du service, comme la facturation, et même des pénalités en cas de rupture de contrat. Le SLA rassure les utilisateurs sur leur idée de déplacer leurs activités vers le cloud en offrant des garanties de QoS.

Une réduction des coûts : le *cloud computing* est économiquement attractif. Il permet de démarrer une activité professionnelle sans avoir à investir en interne dans une infrastructure informatique très coûteuse. Pour les entreprises qui ont déjà leur propre infrastructure et qui ont cependant des besoins informatiques supplémentaires pour faire face aux pics d'activité, le *cloud computing* reste le moyen le moins cher d'y faire face. L'entreprise adapte son infrastructure à ses besoins en augmentant ou en diminuant les ressources disponibles. En souscrivant à des offres dans le *cloud*, l'utilisateur ne paie que ce qu'il consomme. Enfin,

l'utilisateur n'a pas à supporter les coûts liés à l'entretien et à la rénovation de l'équipement. Cela signifie que les entreprises qui bénéficient de services *cloud* pourront réduire considérablement leurs investissements informatiques et optimiser leurs coûts opérationnels.

Un service sécurisé en permanence : faute de temps, de compétences ou de budget, les utilisateurs sont de moins en moins en mesure de garantir pleinement la sécurité de leur propre système d'information. Le *cloud computing* garantit cette sécurité grâce à des dispositifs et services de sécurité bien meilleurs (réplication des données, plan de reprise après sinistre, etc.) avec des mises à jour régulières et des audits de manière transparente.

1.2.5 Classifications du *cloud computing*

Les environnements *cloud* peuvent être classés selon le modèle de service et également selon le modèle de déploiement [15][18].
Nous présentons, dans ce qui suit, les modèles de service, les modèles de déploiement et aussi les acteurs principaux du paradigme *cloud computing*.

1.2.5.1 Modèles de service du cloud computing

La classification selon le modèle de service dépend du type de service fourni. Il existe trois classes de service : Infrastructure en tant que service (IaaS), Plate-forme en tant que service (Paas) et Logiciel en tant que service (SaaS). La figure 1.3 montre les responsabilités du fournisseur et du locataire dans chaque classe.

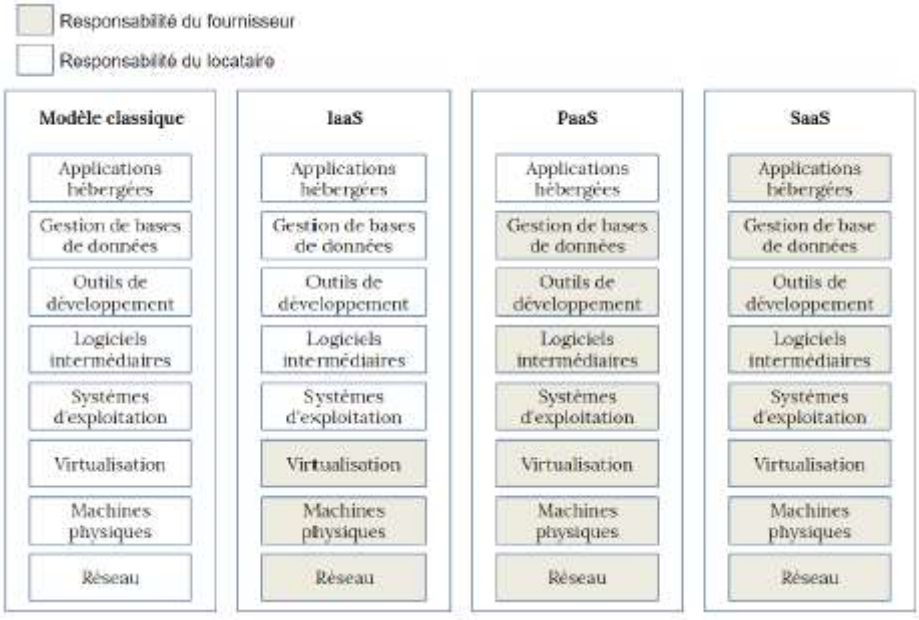

Figure 1. 3 Modèle de service du *cloud* [19]

L'**IaaS** [20] offre le niveau de service le plus bas. C'est à dire l'hébergement de machines virtuelles (VMs) sur les serveurs du datacenter. A ce niveau, le datacenter fournit à ses utilisateurs une infrastructure virtuelle permettant l'exécution des VMs à la demande. Typiquement, le service d'Amazon EC2 permet aux utilisateurs de démarrer de nouvelles VMs, créées entièrement à partir de modèles, et d'y exécuter les programmes de leur souhait. Ce type de service est principalement destiné aux administrateurs système qui préfèrent louer des ressources informatiques plutôt que d'acheter et de gérer des ressources en interne.

Le **PaaS** [21] se positionne au-dessus de IaaS en termes d'abstraction de services. Un tel centre propose à ses utilisateurs un environnement de services travaillant conjointement. Les utilisateurs y hébergent leurs applications en s'appuyant sur cette plate-forme de services. Elle propose en effet des services des services basiques tels que le stockage des fichiers, le support d'une base de données ou l'hébergement de sites web. Elle peut aussi proposer des services spécifiques, comme le développement de logiciel. Par exemple, la plate-forme WINDOWS AZURE 2 héberge des serveurs supplémentaires pour le jeu TITANFALL 3 dans le but de minimiser le traitement au niveau de ses consoles. Outre WINDOWS AZURE, la plate-forme d'applications web GOOGLE APP ENGINE 4 permet aux utilisateurs d'installer leurs applications en tant que scripts. Elle met également à disposition les services de GOOGLE, tels que la gestion des mails ou l'accès à une base de données, réduisant le besoin de configuration des utilisateurs. Au niveau du PaaS, la virtualisation des services n'est pas visible par les utilisateurs qui n'ont accès qu'aux services mis à disposition par la plate-forme. L'élasticité des ressources offertes par le PaaS permet un passage à l'échelle fluide en cas de croissance du nombre de ses utilisateurs. Ainsi, le PaaS est particulièrement adapté pour les start-ups, qui ont besoin de proposer rapidement de nouveaux services à leurs clients.

Le **SaaS** [22] est le dernier niveau du XaaS , au niveau des utilisateurs finaux. A ce niveau, les utilisateurs envoient des requêtes aux applications hébergées chez le fournisseur *Cloud*. Notons que les utilisateurs d'un tel modèle ne sont autres que les utilisateurs des applications déployées par le fournisseur *Cloud*. Le SaaS suppose l'installation d'une plate-forme sous-jacente pour proposer ses services. Ainsi, les éditeurs de solution en ligne, tels que MICROSOFT et sa suite office 365 ; YAHOO et son service de mails ou encore le système de stockage en ligne et partage de Dropbox, peuvent proposer un service client performant, grâce à une plate-forme de services résiliente et à haute disponibilité.

Ces trois niveaux sont complémentaires et dissociables. Dans un centre virtualisé, l'IaaS fournit au PaaS les ressources nécessaires pour supporter la couche logicielle, tandis que le PaaS est configuré par un administrateur pour offrir ses services aux clients sous la forme d'un SaaS.

1.2.5.2 Modèles de déploiement du *cloud computing*

Le modèle de déploiement tient compte de la position du cloud par rapport au locataire. Selon la définition du NIST du cloud computing [15], il existe quatre modèles de déploiement :

Le *Cloud* Public [23] : ce type d'infrastructure est disponible pour un large public et appartient à un fournisseur de *Cloud* services.

Les clients utilisant ce modèle de livraison peuvent accéder aux services cloud via Internet sans savoir exactement où leurs données sont hébergées ou traitées. En pratique, les données des utilisateurs peuvent être déplacées d'un centre de données à un autre pour optimiser les capacités du fournisseur, ce qui peut entraîner de graves problèmes de sécurité. Cependant, l'un des avantages de ce type de mise en œuvre est le coût de mise en œuvre raisonnable. En fait, le coût de mise en œuvre du matériel, des applications et du réseau est supporté par le fournisseur, qui les répartit entre plusieurs clients. Ce type de mise en œuvre offre également l'opportunité de développer les ressources et l'entreprise.

Les principaux prestataires dans le monde du *Cloud* public sont donnés par la Figure 1.4.

Figure 1. 4 Exemples des principaux prestataires de services *cloud* [24]

Le *Cloud* privé [25] : l'infrastructure *Cloud* fonctionne pour une seule organisation. Il peut être géré par l'organisation elle-même ou par un tiers. Dans ce dernier cas, l'infrastructure est entièrement dédiée à l'entreprise et accessible via des réseaux sécurisés de type VPN.

Le *cloud* communautaire : l'infrastructure est partagée par plusieurs organisations qui ont un intérêt commun (par exemple, les exigences de sécurité, conformité, etc.). Comme les clouds privés, les clouds communautaires sont gérés par l'organisation elle-même ou par un tiers.

Le *Cloud* Hybride : l'infrastructure se compose de deux *Cloud* ou plus (privé, communautaire ou public), qui restent des entités certes uniques, mais liées par une technologie normalisée ou propriétaire, qui permette la portabilité des applications ou des données.

1.2.5.3 Acteurs du cloud computing

L'architecture de référence du *cloud computing* décrite par le NIST [26], distingue cinq acteurs principaux :

Cloud consumer : Une personne ou une organisation qui utilise un service fourni par un fournisseur.
Cloud provider : Personne, organisation ou organisme chargé de fournir un service aux parties prenantes.
Cloud broker : Unité qui gère l'utilisation, la performance et la fourniture des services et négocie les relations entre les fournisseurs et les consommateurs.
Cloud auditor : Un organisme qui peut effectuer une évaluation indépendante des services cloud, tels que la sécurité et les performances du cloud.
Cloud carrier : Un intermédiaire qui fournit la connectivité et le transport des services des fournisseurs aux consommateurs.

1.2.6 Notions clés

1.2.6.1 Service Level Agreement (SLA)

Un SLA est un contrat établi entre le fournisseur et le locataire. Il précise d'une façon formelle les différents objectifs et les actions à prendre au cas où ils ne sont pas respectés. Ce concept était introduit pour la première fois dans les années 80 pour gérer la qualité de service dans la télécommunication. Puis il s'est développé avec l'apparition d'Internet, les architectures orientées services et le *cloud*.
Les principaux composants d'un SLA sont les suivants [27] :

- **L'objet** : objectifs à atteindre en utilisant le SLA.
- **Les restrictions** : les mesures ou actions nécessaires qui doivent être prises pour s'assurer que le niveau demandé services est fourni.
- **Période de validité** : période d'application du SLA.

- **Portée** : les services qui seront fournis au locataire et les services qui ne seront pas couverts dans le cadre du SLA.
- **Parties** : les organisations ou les personnes concernées et leurs rôles (principalement le fournisseur et le locataire).
- **Service-level objectives (SLOs)** : niveaux de services sur lesquels les parties s'entendent. Parmi les indicateurs, on trouve : la disponibilité et la performance.
- **Pénalités** : si le service fourni n'atteint pas les SLO des pénalités seront imposées.
- **Services facultatifs** : services qui ne sont pas obligatoires, mais qui pourraient être requis.
- **Administration** : processus utilisés pour garantir la réalisation des SLOs et les responsabilités organisationnelles connexes en matière de contrôle de ces processus.

Le cycle de vie des SLA peut être représenté par six étapes [28] :

- **Étape 1** : découvrir les fournisseurs de services : les fournisseurs de services sont situés en fonction des exigences du locataire.
- **Étape 2** : définir du SLA : cette étape inclut la définition des services, des parties, les pénalités et les SLOs.
- **Étape 3** : établissement d'un accord : le SLA est établi et les parties s'engagent à le respecter.
- **Étape 4** : surveillance de la violation du SLA : la performance du service est mesurée par rapport au contrat.
- **Étape 5** - Terminaison du SLA : le SLA se termine à cause d'un timeout ou d'une violation de la part d'une partie.
- **Étape 6** - l'application des pénalités en cas de violation du SLA : si l'une des parties ne respecte pas les termes du contrat, les clauses correspondantes sont invoquées et exécutées

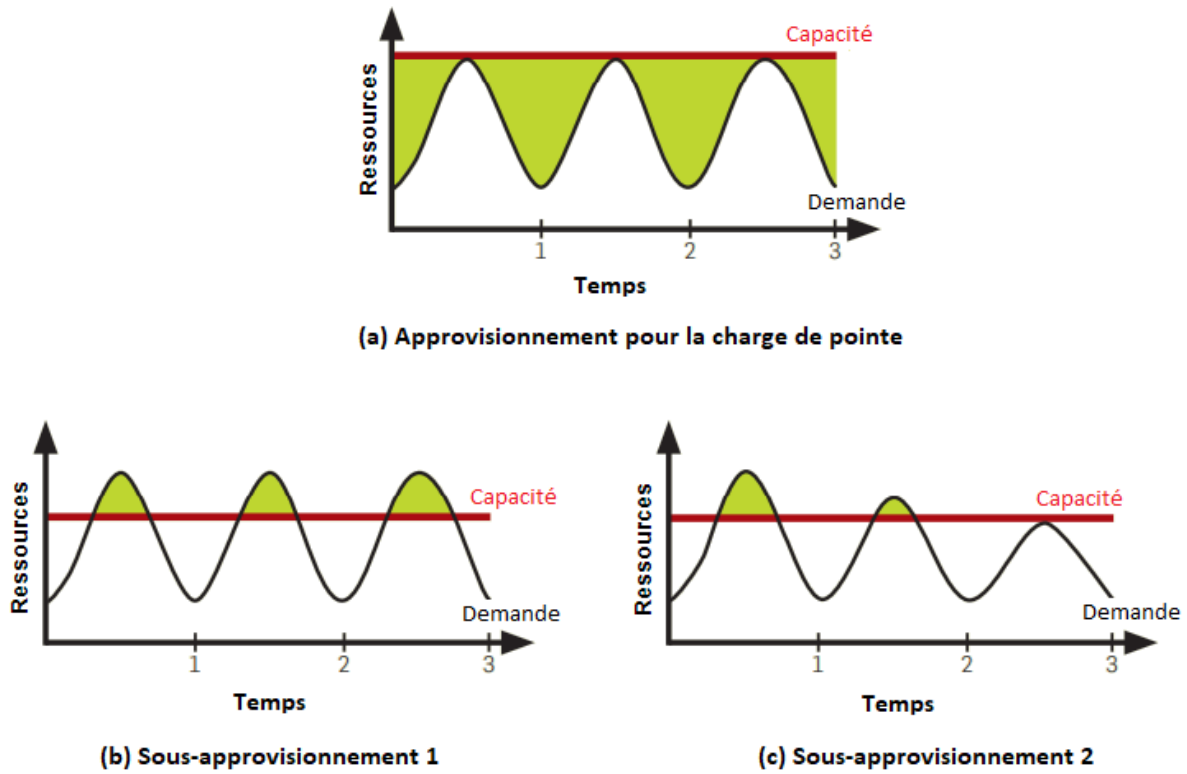

Figure 1. 5 Exemple de sur-provisionnement et sous-provisionnement [29]

1.2.6.2 Élasticité

L'une des principales caractéristiques du *cloud* est son élasticité rapide. L'élasticité est définie comme la capacité d'un système à ajouter et à supprimer des ressources. Ou encore modifier leur configuration, pour s'adapter à la variation de charge en temps réel [30].

Le terme élasticité est associé à l'évolutivité. Cette dernière est définie comme étant la capacité d'un système à supporter des charges de travail croissantes en utilisant des ressources supplémentaires. L'évolutivité est indépendante du temps, il s'agit donc d'une propriété statique. L'élasticité peut être considérée comme une automatisation du concept d'évolutivité, mais elle vise aussi à optimiser au mieux et le plus rapidement possible les ressources à un moment donné [30].

Trois cas peuvent être cités qui montrent l'intérêt d'un système élastique par rapport à un système non élastique. Le premier cas est celui où la demande d'un service varie dans le temps. Dans ce cas, la capacité réservée dans un système non élastique doit satisfaite la charge de pointe ce qui entraîne une sous-utilisation à d'autres moments. Au lieu de cela, un système élastique permet de rajouter ou supprimer des ressources en fonction de la charge ce qui limite le gaspillage. Un deuxième cas est celui où la demande est inconnue à l'avance. Par exemple, une organisation qui démarre devra prendre en charge une hausse de la demande lorsqu'elle

deviendra populaire, suivie éventuellement d'une réduction lorsque certains visiteurs se détourneront. Enfin, les organisations qui réalisent occasionnellement des traitements parallèles massifs peuvent profiter de l'élasticité pour réserver un nombre important de ressources uniquement quand il y a le besoin [29].

Le sur-provisionnement et le sous-provisionnement sont deux facteurs qui caractérisent un système élastique. Le système est dans un état de sur-approvisionnement quand les ressources attribuées sont supérieures aux ressources nécessaires. Comme c'est le cas du provisionnement qui correspond à la charge de pointe de la Figure 1.7(a). La qualité de service est satisfaite en revanche cet état entraîne des coûts supplémentaires et inutiles.

Le système est dans un état de sous-provisionnement quand les ressources attribuées sont inférieures aux ressources requises. C'est le cas de la Figure 1.7(b). Ceci entraîne une dégradation des performances et éventuellement le départ des locataires non satisfaits [30]. La Figure 1.7(c) montre une baisse des demandes à la suite du départ des locataires.

Le terme élasticité est aussi associé à l'efficacité. Cette dernière dépend de la quantité de ressources consommées pour traiter une charge de travail donné. Plus cette charge est faible, plus l'efficacité d'un système est élevée. L'allocation et le dimensionnement automatique visent à répondre aux compromis de la maximisation du bénéfice du fournisseur et la satisfaction des locataires.

1.2.6.3 Modèle économique

L'émergence du *Cloud* est certes une évolution majeure sur le plan technique. Mais son succès est dû au fait que cette évolution est accompagnée de mouvements significatifs sur le plan économique. En effet, ce qui change avec l'introduction de ce nouveau paradigme est la façon dont les entreprises utilisent (consomment) les ressources informatiques en souscrivant à des abonnements qui leur permettent d'utiliser des ressources, de stockages, de calcul et/ou des applications et des plateformes de développement, à la demande. Ainsi, les utilisateurs ne seront facturés que pour ce qu'ils consomment réellement.

Le modèle économique du *Cloud* est basé sur le pay-per-use (paiement à l'utilisation) qui se décompose en deux principales modalités :

- **Pay-as-you-go** : c'est un modèle d'abonnement basé sur l'utilisation réelle de ressources approvisionnées dynamiquement.
- **Pay-as-you-grow** : c'est un modèle d'abonnement similaire au précédent mais légèrement moins flexible dans la mesure où la demande ne peut que s'accroître.

Néanmoins, il applique des tarifs à la baisse en fonction du franchissement des paliers d'utilisation des ressources.

L'objectif du fournisseur *cloud* est de garantir le plus grand bénéfice, tandis que l'objectif de chaque locataire est d'obtenir un service maximal à faible coût. Le bénéfice du fournisseur est la différence entre le revenu et les dépenses. Le revenu quant à lui est la somme d'argent facturée aux locataires qui utilisent les services. Les dépenses sont la somme du coût monétaire lié à l'exécution des services et les éventuelles pénalités dans le cas de violation des SLAs.

Le prix des services est l'une des mesures les plus importantes que le fournisseur doit contrôler pour encourager l'utilisation de ses services et la maximisation de son bénéfice.

Il existe différents schémas de tarification, principalement : la tarification fixe et la tarification dynamique [31]. La tarification fixe comprend le paiement à l'utilisation (les locataires paient en fonction du temps et de la quantité qu'il consomme sur un service spécifique) et l'abonnement (les locataires s'abonnent à une combinaison pré sélectionnée d'unités de service et s'engagent sur une période longue, généralement mensuelle ou annuelle [32]. Dans la tarification dynamique, le prix est déterminé en fonction de l'offre et la demande. Ce modèle permet au fournisseur de mieux exploiter le potentiel de paiement des locataires et donc de réaliser plus de bénéfice. Le prix peut aussi être choisi suite à une négociation entre le fournisseur et le locataire [33].

Dans ce cas, un protocole de négociation est utilisé afin de satisfaire les deux parties.

1.2.7 Virtualisation vs *cloud computing*

La virtualisation[34] une technologie qui fait abstraction des détails du matériel physique et fournit des ressources virtualisées pour les applications de haut niveau. En effet, la virtualisation rassemble toute technologie matérielle ou logicielle qui permet d'exécuter plusieurs configurations informatiques (systèmes d'exploitation, etc.) sur une seule machine physique pour créer plusieurs machines virtuelles qui reproduisent le comportement. La virtualisation est le fondement du cloud computing car elle offre la possibilité de mettre en commun les ressources informatiques des clusters de serveurs et d'affecter ou de réaffecter dynamiquement des machines virtuelles à des applications à la demande.

La Figure 1.6 montre l'architecture multicouche de la technologie de virtualisation. Virtual Machine Monitor (VMM), également appelé hyperviseur, divise la ressource physique du serveur physique sous-jacent en plusieurs machines virtuelles différentes ; chacune s'exécutant sur un système d'exploitation distinct et une pile de logiciels.

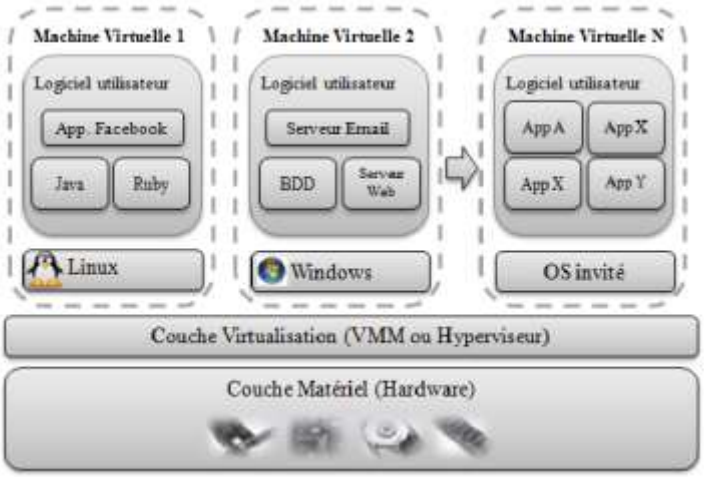

Figure 1. 6 Architecture en couche de la virtualisation [16]

Le cloud computing utilise des technologies telles que la virtualisation, l'architecture orientée services et les services Web. La convergence de plusieurs paradigmes informatiques, tels que Grid *Computing*, Utility *Computing*, Autonomic *Computing* et la virtualisation, a mené à l'avènement du *cloud computing* [35]. La figure 1.7 montre les aspects communs que partage le *cloud computing* avec les technologies connexes.

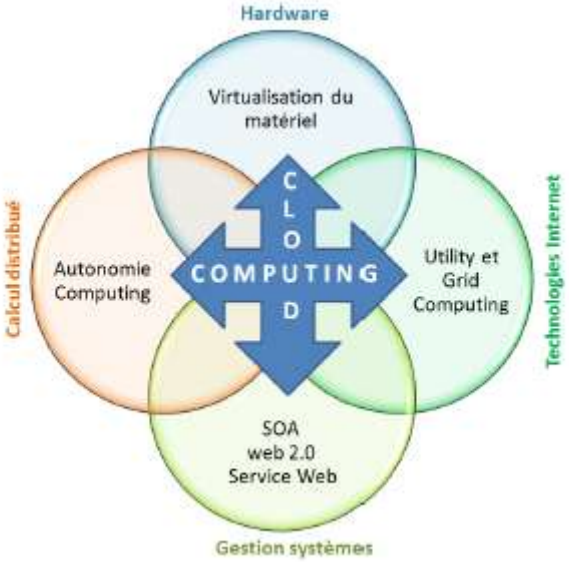

Figure 1. 7 Technologies connexes du*cloud computing* [16]

1.2.8 Enjeux du cloud computing

Le plus grand défi pour l'adoption de services *Cloud* est celui de la sécurité des données. Confier ses propres données à un prestataire externe n'est pas aussi simple, encore moins pour des entreprises. Pour limiter cette perte de contrôle, il est important que les fournisseurs de services *Cloud* localisent les données du client ainsi que les mutualisations correspondantes. En effet, le

cloud computing entraine souvent une dissémination géographique des données, que le client peut ne pas maîtriser. Dans la pratique, les principaux fournisseurs utilisent plusieurs datacenters repartis dans le monde et les données peuvent transiter d'un datacenter à un autre, en fonction des charges respectives et de la fréquence d'utilisation de ces données.
Ainsi, pour un utilisateur européen d'un service de courriel, il est tout à fait possible que sa boîte de réception soit stockée en Europe pendant la journée, puis en Asie pendant la nuit, et que ses mails archivés soient stockés aux Etats-Unis. Cette organisation permet de plus de garantir une meilleure disponibilité et un meilleur temps de latence pour les clients.
En outre, la maîtrise des consommations énergétiques présente aussi un enjeu majeur pour l'offre de services *Cloud*, car les datacenters consomment d'importants volumes d'électricité. En plus de l'énergie électrique nécessaire au traitement des données s'ajoute celle dédiée à la climatisation. Une étude effectuée par Gary Cook [36] a mis en exergue l'évolution de la consommation d'énergie due à l'utilisation des réseaux *Cloud*. Une prise de conscience du problème énergétique dans le domaine de l'IT a incité les fournisseurs de service *Cloud* de rendre leurs équipements de moins en moins gourmands en énergie. Cela les amène également à intégrer des techniques de gestion à leurs systèmes afin d'utiliser le moins de ressources possibles et diminuer aussi la consommation énergétique de l'ensemble de leurs dispositifs.

L'approvisionnement en ressources est un enjeu critique lors de la planification des réseaux. Les fournisseurs peuvent varier la quantité et le type de ressources suivant la baisse ou la montée en charge du service. Ceci permet de véritablement satisfaire les clients en assurant un niveau de qualité de service et d'expérience que les fournisseurs s'engagent à respecter. La gestion des ressources *Cloud* se manifeste dans la planification, la consolidation ou la migration des machines virtuelles.

1.3 Cloud gaming

1.3.1 Définition

Le *Cloud gaming* est un service de jeu en ligne et à la demande. Il présente une solution logicielle émergente qui transforme les jeux vidéo traditionnels en un service *Gaming-as-a-service* (Gaas) [37]. Le jeu n'est plus installé localement sur le terminal de l'utilisateur mais plutôt hébergé dans le *Cloud*. Migrer les jeux vidéo sur le *Cloud* permet la dématérialisation des services de jeux. De ce fait, les puissances hardware des terminaux finaux ne sont plus exigées.

Désormais, les tablettes ou les smartphones sont considérés comme des clients légers. Il n'est plus question d'installer des moteurs de jeu complets sur ces périphériques. Grâce au *Cloud*, la gestion des mises à jour de jeu se fait d'une façon transparente vis-à-vis des utilisateurs. Ainsi, les services de jeux sont des services "Click-and-Play". Le *Cloud gaming* intervient aussi dans la réduction de la consommation de l'énergie des terminaux finaux puisque le calcul de jeu se fait majoritairement aux niveaux des serveurs *Cloud*. Un autre avantage du *Cloud gaming* est la portabilité de service de jeux. Le service de jeu peut tourner sur n'importe quel terminal sans exiger une plateforme spécifique (Linux, Windows, etc).

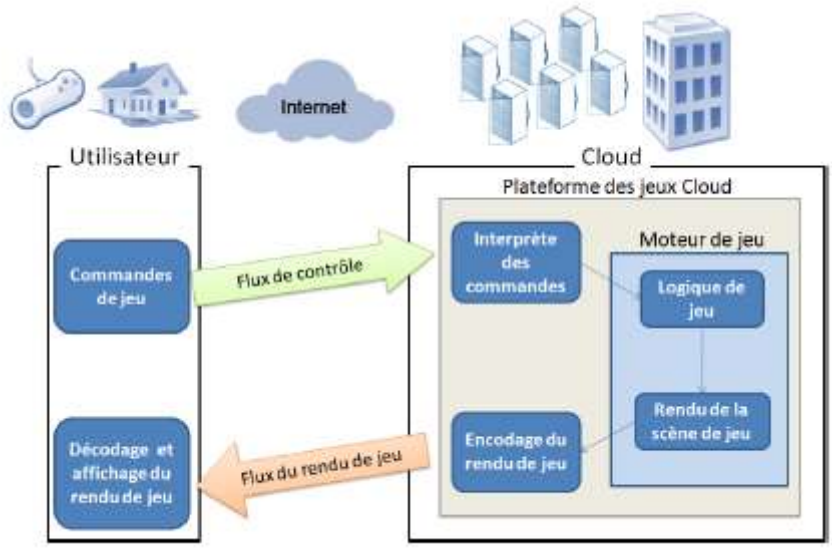

Figure 1. 8 Plateforme standard de *cloud gaming* [37]

La figure 1.8 décrit brièvement la plateforme typique de service *Cloud gaming* [37]. Le flux des informations ascendant est appelé flux de contrôle. Le terminal final est responsable de capturer les interactions de l'utilisateur, les empaqueter en format standard et les envoyer aux serveurs *Cloud*. A leurs tours, ces derniers traitent les paquets de contrôle reçus comme s'ils étaient générés localement, puis produisent le rendu de service de jeu. Une phase d'encodage précède la transmission de flux de rendu de jeu vers l'utilisateur. Une phase de décodage est exécutée au niveau du terminal final pour finir le traitement de flux descendant, voire son affichage.

1.3.2 Plateformes du *Cloud gaming*

G-Cluster, Onlive et Gaikai sont les fournisseurs commerciaux les plus répandus.

G-cluster a commencé le développement des services de jeux *Cloud* depuis le début des années 2000. Au cours de cette période, les datacenters ainsi que la qualité de connexion Internet n'étaient pas aussi performants qu'aujourd'hui. Ceci a poussé G-Cluster à s'appuyer sur des services supplémentaires de qualité de service (QoS) fournis par des opérateurs réseaux. A la fin des années 2000, les entreprises émergentes du *Cloud* ont commencé à proposer leurs services de jeux comme Onlive et Gaikai. Alors pour contrôler la latence de ses services, Onlive a installé en 2009 plusieurs datacenters dans les différents états de l'USA. Quant à Gaikai, pour des raisons de marketing, il a permis aux utilisateurs d'essayer de nouveaux jeux gratuitement ; Puis, à la fin de chaque jeu, il suggère son achat. Ces derniers présentent des systèmes commerciaux de jeux vidéo en streaming. Cependant, la plateforme Gaming Anywhere (GA) [38] est une plateforme du *Cloud gaming* open-source, développée par National Taiwan Ocean University.

1.3.3 Catégories de service du *Cloud gaming*

La catégorisation classique des types des services *Cloud gaming* a été largement adoptée par une étude menée dans [39]. Selon la nature des tâches confiées aux ressources *Cloud*, on distingue différentes catégories de service de jeu. La plus courante exploite le *cloud computing* pour des fins de stockage, de distribution de contenus et/ou des besoins de synchronisation. Une deuxième stratégie consiste à confier le calcul entier de service de jeu aux serveurs *Cloud*. Le flux descendant est alors diffusé en streaming vers les clients légers. Un client léger est tout périphérique, connecté au réseau, capable de traiter les commandes des utilisateurs ou le flux vidéo descendant. Dans ce cas, on parle bien des jeux vidéo en streaming. Une troisième stratégie consiste à allouer des serveurs *Cloud* multi-joueurs et gérer uniquement les changements d'états des joueurs connectés, tandis que d'autres tâches seront exécutées par le terminal final de l'utilisateur. Il s'agit bien des jeux *Cloud* en ligne ou Online *Cloud gaming*.

Tableau 1. 1 Type de jeux *cloud* [40]

Sous-catégories	MMOFPS	MMORPG	MMORTS
Acronyme pour	Massively Multi-player Online First Person Shooter	Massively Multi-player Online Role Playing Game	Massively Multi-player Online Real Time Strategy
Principe de jeu	Jeu de tir	Jeu de rôle	Jeu de stratégie en temps réel
Exemple	World War II Online, Planet Slide	World of Warcraft, Star-Wars Galaxies	Mankind, Shattered Galaxy
Sensibilité au délai	Haute	Moyenne	Faible
Seuil (ms)	100	500	1000

Les MMOGs adoptent la stratégie des jeux *Cloud* en ligne et sont regroupés en plusieurs sous catégories à savoir :

- **Jeux de tir** : ayant l'abréviation de Massively Multi-players Online First Person Shooter (MMOFPS). Ce sont des jeux vidéo massivement multi-joueur reprenant le concept du jeu de tir. Il s'agit de jeux qui présentent généralement un combat en équipe sur un vaste terrain. La figure 1.9 montre un jeu de tir MISSION AGAINST TERROR.

Figure 1. 9 Jeux de Tir

- **Jeux de rôle** : son acronyme anglais est MMORPG pour Massively Multiplayers Online Role Playing Game. Les joueurs physiques, représentés par des avatars, interagissent avec

le monde virtuel ou entre eux pour un objectif bien déterminé. L'évolution temporelle de ce jeu suit celle de la réalité. La figure 1.10 ci-dessus montre une capture d'image d'un jeu de rôle World of Warcraft. Développée par *Blizzard Entertainment Incorporation*, *World of Warcraft (WoW)* est un jeu très populaire des jeux MMOPG. Déjà en 2007, ce jeu tenait le *Guinness World Record* pour la plus grande popularité pour un MMORPG [41].

Figure 1. 10 Jeux de rôle World of Warcraft

- **Jeux de stratégie temps réel** : ils sont connus sous l'acronyme anglais MMORTS pour Massively Multi-players Online Real Time Strategy. Ces jeux sont les fruits de l'union des jeux de stratégie en temps réel avec les jeux massivement multi-joueurs. Pour jouer, les utilisateurs n'ont besoin que d'un simple navigateur Web.

La différence entre eux réside surtout en fonction du principe de jeu et leurs sensibilités au délai de réponse, comme indiqué dans le tableau 1.1.

Figure 1. 11 Jeux MMORTs

1.3.4 Enjeux du Cloud gaming

Parmi les enjeux critiques des services *Cloud* de jeux en ligne, la qualité graphique de jeu vidéo ainsi que sa réactivité en termes de délai de réponse bénéficient d'une attention particulière des chercheurs et fournisseurs de service [39]. Ces derniers sont la clé primaire pour la survie du marché *Cloud gaming* et l'attraction de ses fans.

1.3.4.1 Qualité de l'image

La qualité de l'image est mesurée en fonction de la dégradation vidéo perçue par l'utilisateur par rapport à la vidéo originale générée par le serveur. En fait, le flux vidéo descendant subit des opérations de codage et décodage pour s'afficher à la fin sur l'écran du terminal final. Cela peut provoquer des pertes de données soit au niveau de l'encodeur et/ou le décodeur. Il s'avère alors essentiel de sélectionner un excellent encodeur / décodeur vidéo pour les jeux *Cloud*. Actuellement, l'encodeur H.264 / MPEG-4 AVC [42] est adopté par deux principaux fournisseurs de *Cloud gaming* Onlive et Gaikai. Cet encodeur se caractérise par un taux de compression élevé en temps réel. D'autres effets mènent aussi à une mauvaise qualité d'image comme le délai de réponse, la gigue, la perte de paquets et leur re-ordonnancement.

1.3.4.2 Sensibilité au délai

Les services *Cloud gaming* imposent des exigences strictes vis-à-vis de l'infrastructure déployée. Les ressources *Cloud* doivent être en mesure de calculer, coder et renvoyer le rendu

graphique de jeu avec haute performance. Cela doit être exécuté à l'échelle des millisecondes de sorte que le délai de réponse global soit maintenu au-dessous d'un seuil prédéfini [37] [43]. Nous schématisons l'importante de la contrainte de sensibilité au délai de réponse par un exemple de jeu en ligne de football FIFA. Le jeu de football FIFA est un jeu très populaire en Côte d'Ivoire. Il existe même des compétitions de ce jeu permettant aux joueurs de gagner des gains considérables. Ces jeux en ligne en côte d'Ivoire sont sponsorisés par le groupe de téléphonie ORANGE CI.

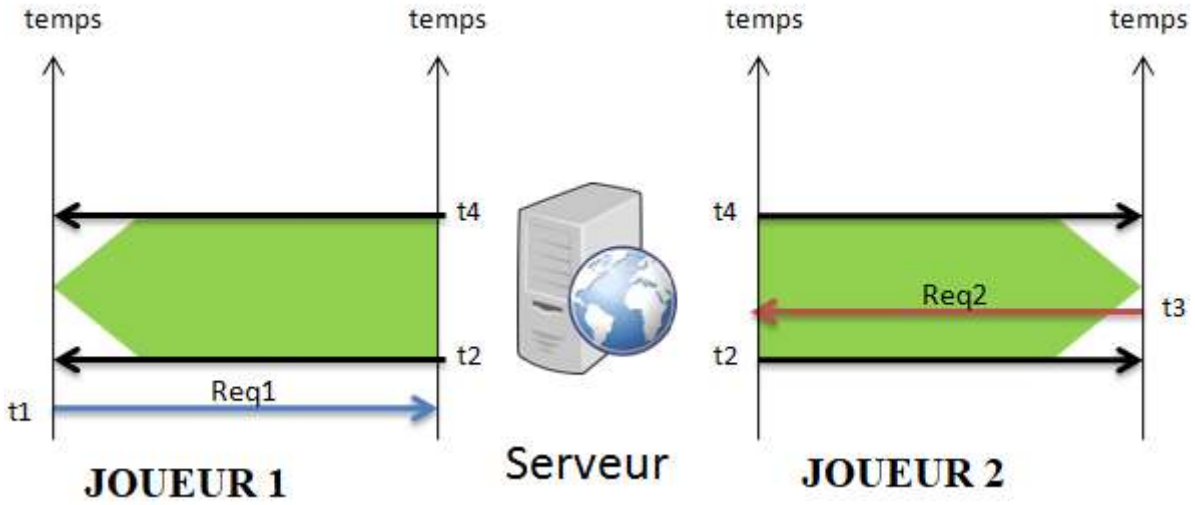

Figure 1. 12 Scénario Jeu de foot FIFA en mode normal

La figure 1.12 décrit le scénario d'un jeu de foot. A l'instant t1, le joueur 1 émet sa requête Req1 pour marquer un but. Reçu à l'instant t2, le serveur de jeu exécute la commande émise et diffuse le rendu de jeu vers les joueurs jusqu'à t4. Le joueur 2 réagit à l'instant t3 en envoyant sa requête

Req2 pour déplacer le gardien vers la balle afin d'empêcher un but. A son tour, le serveur traite la deuxième commande et renvoie le rendu graphique vers les deux joueurs à l'instant t4.

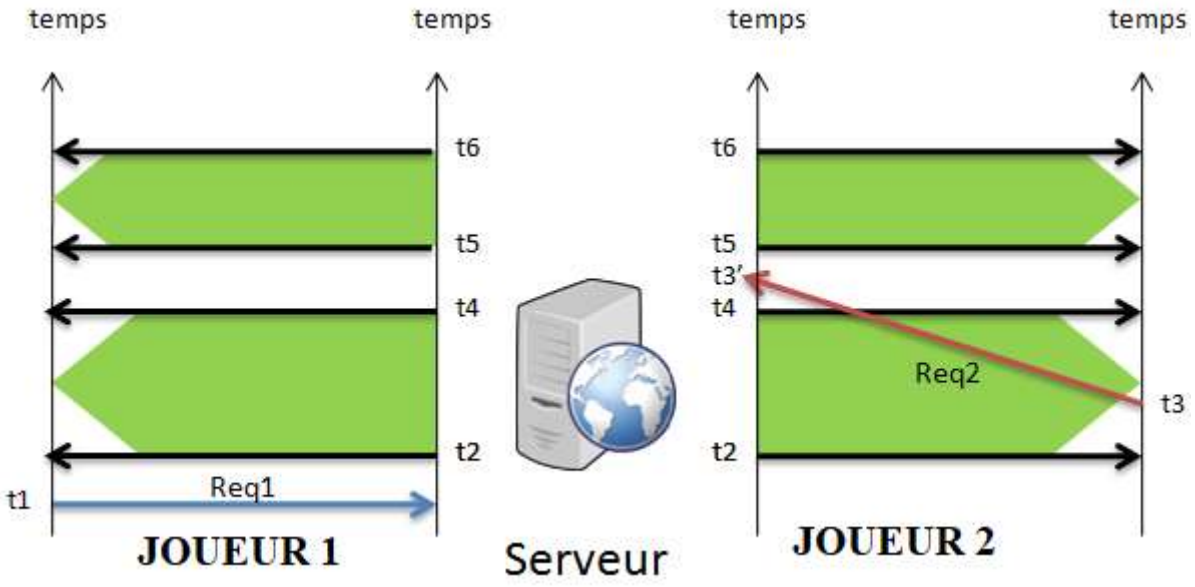

Figure 1. 13 Scénario Jeu de foot FIFA avec délai de réponse

Le deuxième scénario, présenté par la figure 1.13, montre l'effet de délai au niveau des jeux en ligne. Les deux scénarios sont similaires jusqu'à l'instant t3, où la requête Req2 arrive retarder au serveur à l'instant t3'. Alors que la fin de traitement de la requête Req1 est déjà terminé à l'instant t4, avec t4 < t3'. Par conséquence, le gardien ne réagira pas à temps pour empêcher le but. Par la faute de ce retard, le joueur 2 encaisse un but.

Dans la suite, nous introduisons la métrique de délai de réponse. Cette dernière est définie comme le temps écoulé entre l'instant de l'émission d'une commande utilisateur et l'instant d'affichage de son action sur l'écran [39]. Trois éléments essentiels la constituent à savoir : le délai de traitement, le délai réseau et le délai *playout*.

- **Le délai de traitement** résume tous les délais serveurs depuis la réception de la commande de l'utilisateur jusqu'au calcul du rendu de jeu.
- **Le délai réseau** est le temps d'aller-retour (Round Trip Time) entre l'utilisateur et le serveur *Cloud*. C'est le temps que nécessite une commande de jeu pour traverser le réseau vers le serveur approprié et le rendu de jeu pour revenir à l'utilisateur
- **Le délai *playout*** est la différence de temps entre la réception du rendu graphique de jeu et son affichage sur l'équipement de l'utilisateur. La figure 1.14 présente les trois composantes de délai de réponse global.

Figure 1. 14 Délai de réponse global

Les délais de traitement et de *playout* varient en fonction des ressources déployées de calcul, de mémoire. Plus ces ressources sont puissantes, plus ces délais sont minimisés. Toutefois, le délai réseau dépend étroitement de la distance physique entre l'utilisateur et le datacenter. Le choix des emplacements géographiques des datacenters est généralement basé sur les opportunités de refroidissement, le coût d'électricité ainsi que l'emplacement géographique des utilisateurs. De même, l'état du réseau (congestion, perte de paquets, etc.) agit bien évidement sur le délai réseau.

La figure 1.15 illustre plus en détail le calcul de délai de réponse pendant une session de jeu. La réactivité de jeu se base essentiellement sur le délai de réponse global. Le délai de réponse est défini comme le temps d'aller- retour de flux de données ou bien le Round Trip Time (RTT) entre le joueur physique et le serveur *Cloud* y attaché.

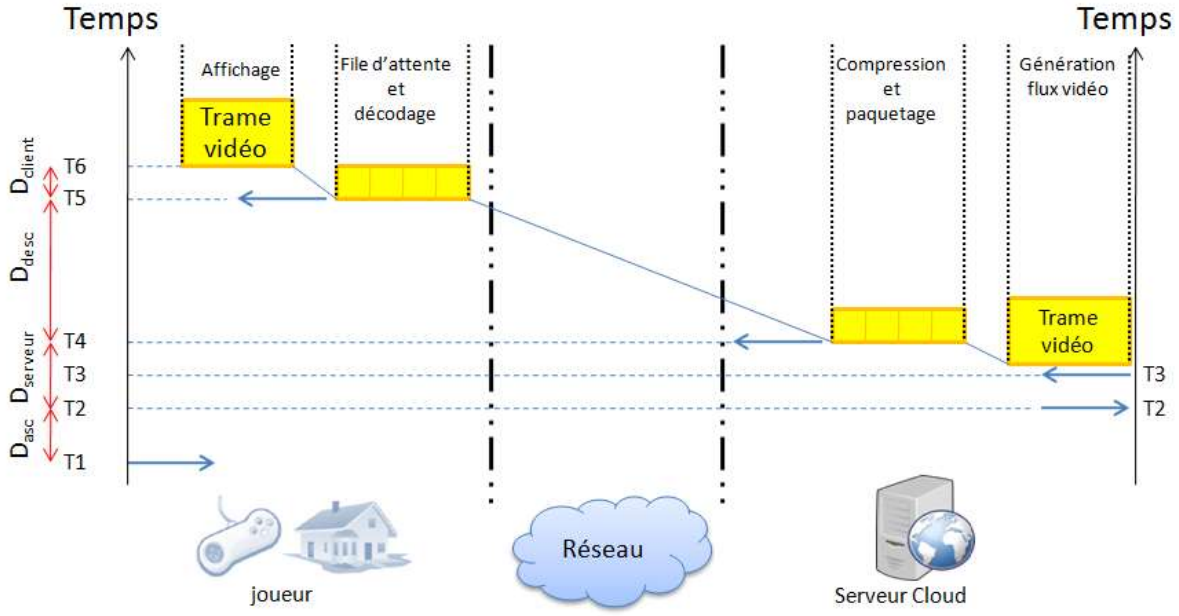

Figure 1. 15 Cycle RTT d'une session de jeu *cloud* [16]

La figure 1.15 décrit le cycle RTT d'une session de jeu *Cloud gaming*. Le joueur émet une requête de jeu à l'instant T1. Reçu à l'instant T2, le serveur de jeu l'encaisse dans sa file d'attente si d'autres requêtes sont en cours de traitement. Ce délai d'attente est noté Ds. Le moteur de jeu exécute la requête et génère son rendu graphique pendant un délai de traitement, noté Dp. Le codage et le paquetage de rendu descendant se fait pendant un délai de codage, noté Dc, égale à (T4 -T3). Le joueur reçoit les paquets vidéo au bout d'un délai réseau Dtr égal à (T5 -T4). L'affichage de la vidéo sur le terminal nécessite un délai de file d'attente et de décodage, noté Dd, entre les instants (T6 - T5).
Ainsi le délai RTT est exprimé comme suit :

$$RTT = D_s + D_p + D_c D_d + 2 * D_{tr} \qquad (1.1)$$

1.3.5 Avantages du *cloud gaming*

Les avantages des services du *cloud gaming* comprennent l'évolutivité, la rentabilité, une solution anti-piratage efficace, une prise en charge omniprésente et multiplateforme.

- **Évolutivité :** L'évolutivité permet de surmonter les contraintes de ressources des terminaux de jeu, notamment l'énergie de la batterie, la capacité de traitement et le stockage des données des appareils mobiles.
- **Rentabilité** : Le rapport coût-efficacité minimise le coût de production avec une technique d'expansion utilisée.
- **Support omniprésent et multiplateforme** : Le support omniprésent et multiplateforme offre un jeu sans faille et une expérience multiplateforme.
- **Solution anti-piratage efficace** : Les jeux en ligne offre une meilleure solution aux questions complexes de confidentialité. Le code binaire est situé dans le serveur *cloud* sécurisé, il est impossible de cracker les jeux et les consoles.
- **Cliquer et jouer :** La fonction "Click and play" permet de faire fonctionner le mode "play-as-you-go" où le joueur peut initialiser la session de jeu sans avoir à installer ou télécharger tout le logiciel de jeu.
- **Efficacité énergétique** : L'efficacité énergétique a des fonctionnalités bien construites qui permettent d'économiser la batterie des terminaux mobiles. De prolonger les temps de jeu des joueurs en déchargeant les programmes liés aux jeux qui produisent d'importantes complexités opérationnelles vers le *cloud*. Par conséquent, le développement de solutions de jeu pour le nuage est très important pour les industries de réseau et les jeux en nuage[44].

1.4 . Qualité d'expérience (QoE)

1.4.1 Définition

Dans diverses définitions formelles et informelles, l'expérience de jeu a été définie comme une " mesure subjective du degré de satisfaction de l'utilisateur cible par rapport à son statut et à ses attentes explicites en matière d'expérience "[45].

Une définition complète et générale n'existe pas pour la "qualité d'expérience" ou QoE. C'est une notion complexe qui prend en compte de nombreux paramètres, dont certains sont liés aux sentiments et à des notions très subjectives de l'être humain.

Le terme de "qualité de l'expérience" est une extension d'un terme beaucoup plus ancien nommé "qualité de service" ou QoS. La QoE a émergé, principalement, pour donner suite à l'insuffisance de la QoS pour évaluer subjectivement qu'objectivement un service ou une application. Tandis que la QoS évalue les paramètres techniques des réseaux, la QoE donne une idée sur le niveau de perception de l'utilisateur final envers une application ou un service. Pour comprendre la notion de la QoE, l'Union Internationale des Télécommunications (UIT) l'a défini comme l'acceptabilité globale d'une application ou d'un service tel que perçu subjectivement par l'utilisateur final.

Dans les jeux vidéo, l'expérience du joueur est un concept large qui détermine la perception globale d'un jeu par le joueur. Elle comprend de nombreux facteurs subjectifs qui influencent le degré de satisfaction du joueur à l'égard d'un jeu. Certains de ces facteurs sont ses sentiments, sa compréhension des règles du jeu, la difficulté du jeu, les interactions, l'esthétique, la narration, etc.

1.4.2 Facteurs de la Qualité d'Expérience

L'application demandée, son contexte d'utilisation, les conditions de réseaux, agissent sur le niveau de la QoE. La figure 1.16 résume les relations entre les facteurs objectifs et la qualité subjective perçue à la fin. Ces facteurs agissent de façon complexe et significative sur la qualité visuelle et la réactivité de jeu.

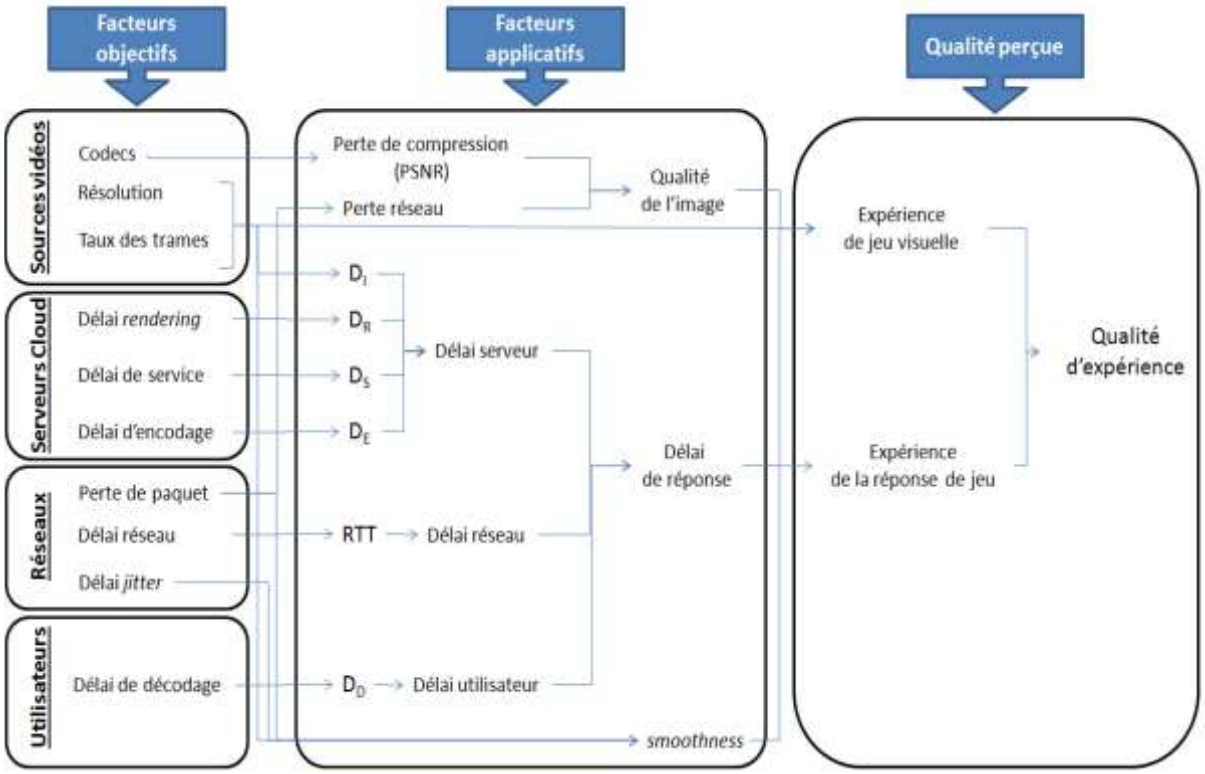

Figure 1. 16 Facteurs influant la QoE [16]

Nous distinguons quatre sous-groupes de facteurs objectifs à savoir :

- La Source vidéo : la qualité de l'image vidéo est déterminée par le taux des trames vidéo, la perte des paquets et la gigue. Par exemple, la compression de flux vidéo produit certaine dégradation de la qualité vidéo. Le niveau de dégradation est lié au codec utilisé et le paramètre de quantification qui représente le degré de compression de l'image.
- Les Conditions de réseaux : la congestion des réseaux génère des délais serveur et réseau importants ainsi qu'un taux élevé de perte des paquets, ce qui dégrade énormément la QoE.
- Le Traitement au niveau des serveurs,
- Et le traitement du côté utilisateur.

Il existe une relation directe entre la traditionnelle qualité de service (QoS) et le QoE [46]. En effet, dans les jeux vidéo, il est prouvé que les mesures de la QoS telles que le taux de trames vidéo et le temps de réponse ont une influence directe sur la QoE perçue par le joueur.

Aussi, il existe une forte corrélation entre les conditions de réseau, la réactivité du jeu et la qualité vidéo.

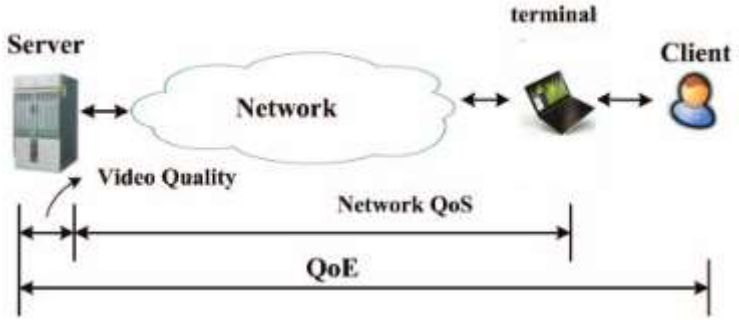

Figure 1. 17 Relation en QoS et QoE

Dans la section ci-dessous, nous présentons les travaux existants dans la littérature qui traite de notre sujet.

1.5 Principaux challenges

Ces dernières années, le *cloud gaming* s'est considérablement développé en raison de l'essor du *cloud* et des réseaux haut débits. Le *cloud gaming* est une nouvelle façon de proposer des jeux vidéo aux utilisateurs. Ils sont exécutés sur de puissants serveurs *cloud*, et le rendu de jeu est diffusé en continu sur Internet aux joueurs équipés de clients légers.

Depuis la fin des années 2000, des services de jeux en ligne proposés par des start-ups, comme OnLive (www.online.com) et Ubitus (www.ubitus.net) font leurs parutions. Pour ne pas être en marge d'une activité à très fort potentiel et se préparer à la disparition prochaine des consoles, le groupe japonais SONY(www.sony.fr), l'un des principaux développeurs de consoles de jeu, a racheté en 2012 la plateforme de *cloud gaming* Gaikai pour 380 millions de dollars [47]. L'immense popularité du *cloud gaming* est due à plusieurs avantages pour les joueurs, les développeurs de jeux, et les fournisseurs de services.

Le *cloud gaming* permet aux joueurs d'accéder à leurs jeux partout et à tout moment. D'acheter ou de louer des jeux à la demande. D'éviter de mettre régulièrement à jour leur matériel et de bénéficier d'expérience unique comme la possibilité de migrer d'un ordinateur client à un autre pendant les sessions de jeu, et le partage des jeux avec leurs amis géographiquement éloignés.

Pour les développeurs de jeux, le *cloud gaming* leur permet de se concentrer sur une seule plate-forme, ce qui réduit les coûts de portage et de test ; de contourner les détaillants pour obtenir des marges bénéficiaires plus élevées ; de toucher un plus grand nombre de joueurs ; d'éviter le piratage puisque le logiciel de jeu n'est jamais téléchargé sur les ordinateurs clients.

Pour les fournisseurs de services, le *cloud gaming* conduit à de nouveaux modèles de gestion, et crée davantage de demandes sur les ressources *cloud* déjà déployées ; contribue à l'amélioration de la qualité de service des applications d'exécution à distance, étant donné que le jeu en ligne impose des contraintes strictes sur les diverses ressources informatiques et le réseau.

Mais malgré les grandes possibilités qu'offre ce nouveau service, plusieurs défis cruciaux doivent être relevés par la communauté scientifique avant de pouvoir exploiter pleinement son potentiel et attirer davantage de joueurs, de développeurs de jeux et de fournisseurs de services. Pour cela, de nombreux travaux sur le sujet ont été menés, et d'autres le sont encore pour faire face aux nombreuses difficultés liées à ce service. Cette section donne un aperçu de différents travaux de recherche pour l'amélioration des services de *cloud gaming*.

Les principaux défis du *cloud gaming* concernent la qualité d'expérience et la réduction des coûts d'exploitation. Une étude exhaustive sur la littérature dans le *cloud gaming* permet d'identifier les principaux challenges dans le *cloud gaming* afin de permettre à la communauté scientifique de contribuer aisément au développement et à la construction des futures plateformes de jeux en ligne [6]. La figure 1.18 montre une classification de la littérature sur le *cloud gaming*.

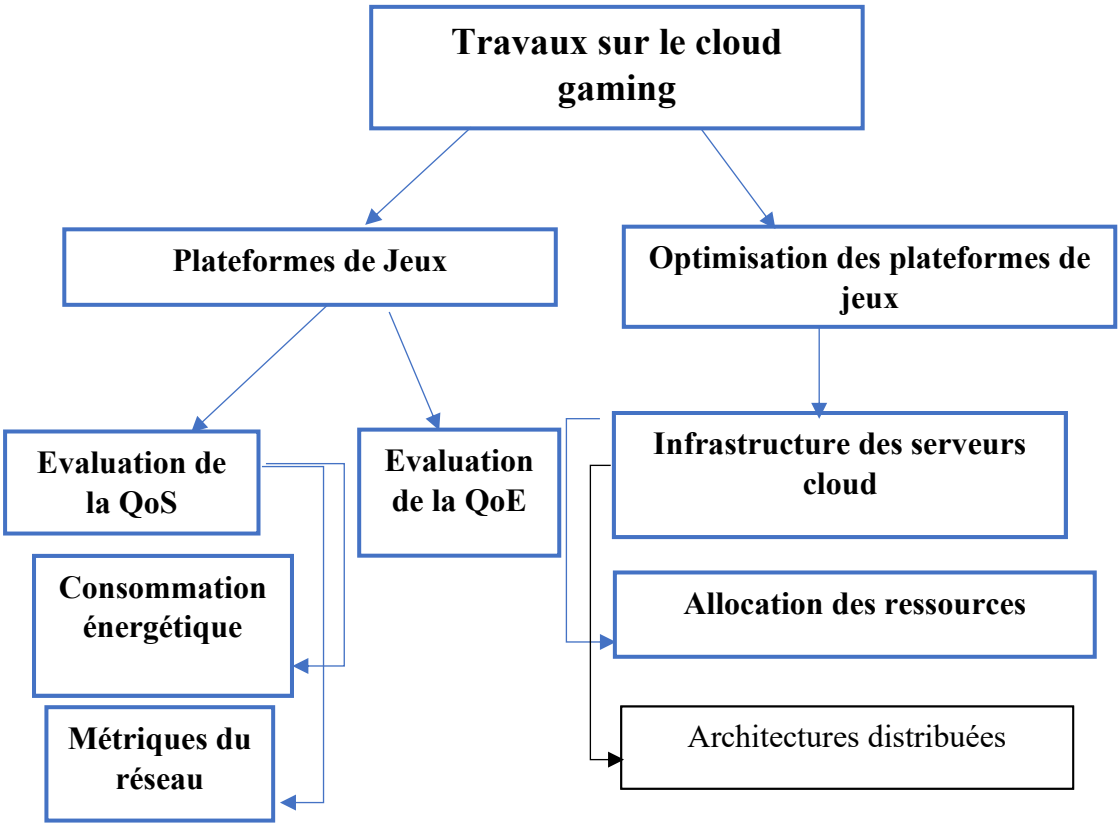

Figure 1. 18. Classification des travaux sur le *cloud gaming*

Nous explorons les axes qui traitent des plateformes et de l'optimisation des plateformes de *cloud gaming* à savoir les défis sur la QoE, la QoS, la consommation énergétique et l'allocation des ressources.

1.5.1 Evaluation de la qualité de service (QoS)

1.5.1.1 Problématique

La mesure de la QoS est essentielle pour quantifier les performances des plateformes de jeux cloud. Cela nous permet de résoudre efficacement les problèmes et même d'optimiser dynamiquement les plateformes de cloud gaming. Pour beaucoup d'autres applications multimédia distribuées, la QoS dépend fortement du réseau. Par conséquent, l'évaluation de différentes métriques de réseau dans les jeux en ligne est cruciale, et nous présentons ci-dessous des travaux existants.

1.5.1.2 Travaux et contributions existants

Plusieurs méthodes ont été proposées pour résoudre le problème d'évaluationde la QoS des jeux en ligne multi-joueurs. Principalement, ces méthodes visent à améliorer les métriques du réseau. Par exemple certaines applications nécessitent un taux de perte optimisé voire nul ou un délai de traitement réduit vue la sensibilité d'un service pareil au temps de réponse. Dans ce sens, les auteurs [43] ont analysé les délais de réponse de système de jeu en ligne et les ont segmenté en trois parties, à savoir le délai du réseau, le délai du traitement et le délai de diffusion. Grâce à cette décomposition, les auteurs ont proposé une méthodologie pour mesurer les composants de latence et ont appliqué la méthodologie sur OnLive et StreamMyGame, deux plates-formes de jeux cloud très populaires. Les auteurs ont constaté que le système OnLive est plus performant que StreamMyGame en termes de latence, en raison de la stratégie d'approvisionnement en ressources différente, basée sur les genres de jeux. Un travail ultérieur [48] réalisé par le même groupe étend le modèle en ajoutant le délai du jeu, qui représente la latence introduite par le programme de jeu pour traiter les commandes et restituer l'image vidéo de la scène du jeu. Aussi, Ils ont étudié également comment la conception du système et les paramètres sélectifs affectent la réactivité avec la complexité de la scène, la taille des régions mises à jour, les résolutions d'écran, et la puissance de calcul. Leurs observations montrent qu'une qualité de réseau médiocre avec un taux de perte de paquets élevé et une bande passante insuffisante, a un impact négatif sur ces plateformes de jeux, entraînant des taux de fréquence d'images plus faible et une mauvaise qualité vidéo.

En outre, Suznjevic et al. (Suznjevic, et al., 2014) ont évalué 18 jeux sur la plateforme de jeu GamingAnywhere [50] pour analyser la corrélation entre les caractéristiques des jeux et leur trafic réseau. Les auteurs ont observé des valeurs les plus élevées pour les jeux de mouvement, d'action et les jeux de tir, tandis que la majorité des jeux de stratégie sont relativement faibles. Ils ont conclu également que l'utilisation de la bande passante pour la plupart des jeux se situent dans une fourchette de 3 à 4 Mbit/s, sauf les jeux de stratégie qui consomment moins de ressources réseau. Une autre conclusion notable est que le taux d'action des joueurs introduit une légère augmentation du débit de paquets, mais n'affecte pas le volume de trafic réseau généré.

Quant aux auteurs Kyungmin Lee et al. [51], ils ont utilisé le modèle de spéculation Outatime pour permettre l'interaction séquentielle à faible latence dans le *cloud gaming*. Le modèle Outatime prédit les trames spéculatives et les diffuse au client, puis les récupère rapidement en cas d'erreur de spéculation. La latence a été réduite en combinant certaines propriétés d'Outatime. Bien que le temps de traitement de cette méthode soit faible, elle ne permet pas toujours d'obtenir de meilleurs résultats en raison des erreurs.

Dans (Z. Xue, et al.,2015), les auteurs ont mené une étude de mesure passive et active pour CloudUnion, un système de jeu en ligne chinois. En comparant CloudUnion et GamingAnywhere [50] , les auteurs ont observé quatre problèmes communs. Premièrement, les débits de données ascendants et descendants sont asymétriques. Deuxièmement, les jeux à faible mouvement perçoivent une gigue périodique à l'intervalle de 10 secondes. Troisièmement, les flux audio et vidéo souffrent d'un problème de synchronisation. Quatrièmement, la perte de paquets dans la transmission du réseau dégrade considérablement l'expérience de jeu de manière significative.

Dans l'optique d'améliorer les paramètres QoS, Yuhua Lin et Haiying Shen [53] ont développé un modèle *Cloud*fog. Le modèle *Cloud*fog est composé d'un modèle de 'fog' et d'un modèle de réputation. Le 'fog' est composé de super-nœuds, capables de restituer des vidéos de jeux et un modèle de streaming. Une stratégie de super-nœuds basée sur la réputation a été utilisée pour allouer chaque joueur aux nœuds appropriés. Ces deux modèles ont été utilisés pour réduire la latence et améliorer la qualité de service. Cette méthode a permis d'augmenter la couverture de l'utilisateur et la consommation de bande passante, bien qu'elle ne dispose pas de modèles de sécurité pour préserver le réseau.

Maryam Amiri et al. [54] quant à eux ont utilisé un modèle d'optimisation bi-objectif pour estimer le chemin optimal de transmission des données entre l'utilisateur et le centre de données. Ce modèle a été développé pour diminuer le délai et augmenter l'utilisation de la bande passante. Le modèle hiérarchique AGAR permet de sélectionner le meilleur chemin de routage pour une session de jeu en fonction du type de jeu ainsi que de ses exigences, telles que le délai et la bande passante. Cette méthode a été mise en œuvre dans un contrôleur SDN, qui fournit une vue globale des datacenters. Elle a permis d'obtenir de meilleurs résultats en termes de délai de communication et de bande passante.

1.5.2 Qualité d'expérience (QoE)

1.5.2.1 Problématique

L'expérience utilisateur, connue en anglais sous le nom *User Experience*, est devenue l'une des métriques les plus pertinentes pour évaluer les fonctionnalités de bout en bout des systèmes cloud. Pour conserver leur part de marché et rester compétitifs, les développeurs de jeux cloud et les fournisseurs de services doivent maintenir et augmenter les abonnements de leurs clients. Pour répondre à leurs besoins et assurer leur satisfaction, ils doivent disposer de solutions efficaces de suivi et d'estimation de la qualité d'expérience (QoE). Cependant, la QoE est toujours une mesure subjective. De plus, son évaluation est à la fois coûteuse et fastidieuse, car elle nécessite une forte participation humaine, que l'on appelle « panel d'évaluation ». Par conséquent, développer un outil en temps réel pour mesurer la QoE avec une précision raisonnable est devenu un besoin primordial et est défi intéressant. Nous présentons ci-dessous les travaux qui traitent de cette problématique.

1.5.2.2 Travaux existants

La garantie d'une QoE acceptable est le défi majeur du marché *Cloud gaming* envers sa clientèle. Cependant, sa mesure, sa modélisation et même sa prédiction ne sont pas évidents à cause de la subjectivité de ses métriques.

Dans cet axe, les auteurs [55] ont proposé une méthodologie de mesure et modélisation de la QoE pour les services *Cloud gaming* reposant sur une architecture de test à trois équipements distants. Leurs expérimentations ont montré que la QoE perçue par le joueur est une fonction de la qualité de l'image et son débit de trames (frame rate FPS). Ils ont montré également que ces équipements distants disposent de divers niveaux de QoE sous les mêmes conditions réseaux.

Dans [56] [57], les auteurs ont proposé une plateforme de simulation pour les services *Cloud gaming*. Cette plateforme mesure la QoE en fonction d'une métrique "agrégée". Cette mesure est également relative au ressenti de l'utilisateur appelée Mean Opinion Score ou MOS. Parmi les facteurs influants cette métrique, on trouve notamment les paramètres QoS comme le délai de retard, la perte de paquets, la catégorie de service de jeu, etc. Leur enquête a montré que très peu de joueurs ont l'intention de payer des frais mensuels pour les services *Cloud gaming*. Il s'est avéré nécessaire de proposer des modèles de paiement ou d'abonnement innovants. Une autre plateforme de simulation présentée dans [58], a été implémentée pour des tests subjectifs. Les MOS considérés sont la sensibilité des inputs, la qualité vidéo, la qualité audio, la qualité globale, la complexité, le niveau de plaisir et les valeurs perçues. Leur étude a synthétisé une forte interaction entre les métriques de la QoE, de la QoS, de la configuration des plateformes et du software implémenté.

L'article de [59] a montré que les délais de réponse variaient suivant le type de service de jeu. Les auteurs ont aussi développé un algorithme de mesure de la latence en fonction de la dynamique des scènes de jeu et la fréquence d'émission des commandes joueurs. Les auteurs de [60] ont tiré les mêmes conclusions après avoir réalisé des expérimentations exhaustives. Par exemple, ils ont déduit que les jeux d'actions sont plus sensibles aux délais que les catégories de jeu.

Le travail [61] présente une étude des effets objectifs et subjectifs des délais réseaux sur les services *Cloud gaming*. Ils déduisent que les valeurs MOS se dégradent progressivement avec la croissance du temps de latence. Plus encore, les services *Cloud gaming* sont généralement très sensibles aux délais réseaux.

Les chercheurs dans (Huang et al., 2014) ont conduit des expérimentations exhaustives pour les services *Cloud gaming* avec des populations variées, mobiles et statiques. Leur travail a abouti à plusieurs idées intéressantes. A titre d'exemple, les joueurs mobiles étaient plus satisfaits avec la qualité de l'image, tandis que les joueurs non mobiles étaient plus à l'aise avec la qualité de contrôle via leurs postes. En outre, le débit, le taux des trames et la latence réseau influent d'une manière significative la qualité graphique ainsi que sa fluidité. Néanmoins, la qualité de contrôle ne dépend que du type de l'utilisateur (mobile ou statique).

Les auteurs des travaux [63] et [64] ont réalisé une plateforme de simulation des services *Cloud gaming* mobiles pour des tests subjectifs de la QoE. Ils proposent alors le modèle GMOS pour "Game Mean Opinion Score". Il s'agit d'une modélisation de la QoE en fonction de la catégorie

de jeu, de la configuration de son flux en streaming, du signal PSNR (Peak Signal to Noise Ratio), du délai de latence et de la perte des paquets. Ce modèle peut être déployé pour des fins d'optimisation de l'expérience de jeu.

Plus récemment, les auteurs [65] ont proposé une méthode heuristique pour l'optimisation des ressources en fonction de la QoE. Ce modèle basé sur la théorie des jeux a permis de répondre aux exigences globales des joueurs, de réduire le coût de calcul et d'offrir une meilleure performance par rapport à d'autres méthodes.

Quant à Fatima Haouari et al. (Haouari et al., 2019), ils ont développé un modèle d'allocation des ressources basé sur la prédiction pour augmenter la qualité d'expérience des joueurs et diminuer le coût de l'allocation des ressources. L'emplacement des téléspectateurs et le nombre de téléspectateurs disponibles ont été prédits par une approche d'apprentissage automatique. Leur algorithme a fait correspondre le résultat prédit avec les valeurs réelles, puis a alloué les ressources à proximité des spectateurs. Mais, cette approche pourrait être évalué avec d'autres modèles de prédiction et différents scénarios.

Ivan Slivar et al. [67] ont eux aussi à proposer une approche pour l'optimisation de l'allocation des ressources *cloud* en fonction de la QoE pour plusieurs utilisateurs. Le problème d'optimisation est défini à l'aide d'algorithmes qui utilisent leurs modèles d'estimation de la qualité d'expérience dérivés d'études subjectives pour différents types de jeux [68]. Plus spécifiquement, l'étude s'est penchée sur le processus d'allocation des ressources lorsque l'on considère conjointement deux objectifs d'optimisation à savoir la qualité et l'équité. Les résultats ont montré de bonnes performances et confirment que les fournisseurs doivent tenir compte du type de jeu lors de l'adaptation des paramètres de codage vidéo et de l'allocation des ressources mais la complexité de calcul de cette méthode était élevée.Mais, ces résultats peuvent être étendues en allégeant le processus automatique de décision des stratégies d'adaptation de codage optimale pour un jeu spécifique.

Plus récemment, les auteurs Yiwen Han et al. [69] ont conçu un modèle de compétition de ressources basé sur la QoE pour les jeux *cloud* sur mobile en utilisant l'optimisation du placement des machines virtuelles. La méthode repose sur l'application de la théorie des jeux. La méthode a obtenu de meilleures performances que les autres, mais une fonction fitness plus adaptée n'a pas été développée pour l'évaluation. Cette performance peut être encore améliorée en introduisant une fonction d'ajustement appropriée pour une meilleure prise en compte des

contraintes du *cloud gaming*. En outre, les contraintes des variables multidimensionnelles du *cloud* peuvent être prises en compte simultanément pour une optimisation plus poussée.

1.5.3 Consommation énergétique

1.5.3.1 Problématique

Avec la popularité du *cloud computing*, la consommation énergétique des *datacenters cloud* augmente de plus en plus. Selon l'estimation d'Amazon, les coûts liés à l'énergie des centres de données cloud représentent 42% des coûts d'exploitation totaux [70]. En outre, l'augmentation de la consommation d'énergie a entraîné une augmentation spectaculaire des émissions de CO2 et a affecté directement notre environnement. Par conséquent, la réduction de la consommation d'énergie est un problème urgent qui doit être résolu. Elle réduira non seulement les coûts de l'énergie mais aussi la durabilité environnementale.

1.5.3.2 Travaux existants

Certains auteurs cités ci-dessous ont proposé des approches pour faire face aux défis susmentionnés.

Macro Guazzone et al. [71] ont proposé un modèle de fédération de fournisseurs *cloud* pour réduire la consommation d'énergie. Cet algorithme basé sur le concept de la théorie de jeux coopératif permet une allocation de la charge en tenant compte de l'énergie. Le modèle de leur système peut être étendu en améliorant la fonction de valeur de la coalition pour tenir compte des pertes probables de demandes dues à la perte de ressources physiques. En outre, le modèle de théorie des jeux peut être étendu pour améliorer les approches de théorie des jeux et d'optimisation afin d'inclure les coûts sur la base de la perte de revenus ainsi que d'autres aspects, comme celui qui dépend de la confiance entre cloud providers.

Dans [72], les auteurs ont conçu un modèle d'allocation de ressources en fonction de l'énergie par le biais du jeu de Stackelberg. Cette méthode vise à minimiser la consommation d'énergie dans le *cloud*.

Yang Ge et al. [5] ont développé une stratégie basée sur théorie des jeux pour optimiser la consommation d'énergie dans un système mobile-*cloud*. Cette méthode a été modélisée pour résoudre le problème de minimisation de l'énergie. Dans ce modèle, chaque mobile est considéré comme un joueur, dont la motivation est de choisir un des serveurs pour décharger le calcul, tout en réduisant la consommation d'énergie. L'équilibre de Nash est toujours présent

dans ce jeu, et un algorithme efficace a été développé pour atteindre l'équilibre de Nash en temps polynomial.

Damian Fernandez et al [73] ont conçu un outil de simulation, appelé GAME-SCORE pour les jeux *cloud*. L'outil de simulation a mis en œuvre le modèle d'ordonnancement du jeu de stackelberg. Ce jeu introduit deux joueurs à savoir le planificateur et l'agent d'efficacité énergétique. L'efficacité du modèle d'ordonnancement a été analysée à l'aide du modèle de simulation GAME-SCORE et les résultats obtenus montrent que l'ordonnanceur de Stackelberg est plus performant que les stratégies statiques d'optimisation de l'énergie.

Ali Vafamehr & Khodayar [74] ont développé des approches d'estimation du *cloud* sensibles à l'énergie et un modèle de marché du *cloud* pour minimiser la consommation d'énergie. Cette méthode a été utilisée pour améliorer la qualité, ce qui a réduit le coût de l'approvisionnement en énergie.

Haibing Guan et al. [75] ont développé une conception structurelle de contrôle à deux couches pour un protocole de jeu en nuage économe en énergie, basée sur les approches de contrôle de rétroaction bien organisées. La première boucle de contrôle employait un contrôleur proportionnel intégral pour garantir les assurances SLA, qui sont mesurées à chaque image par seconde pour chaque jeu en ligne. La deuxième boucle a été modélisée pour réduire la consommation d'énergie, en se basant sur la trame actuelle obtenue par la boucle primaire. Bien que cette méthode ait permis de réduire la consommation d'énergie, elle n'a pas été adaptée aux applications du monde réel.

Hua-Jun Hong et al. [76] ont proposé une heuristique sensible à l'énergie pour le *cloud gaming*. Leur étude vise à optimiser le profit des fournisseurs tout en obtenant une bonne qualité d'expérience.

Mohammad Mehedi Hassan et al. [77] ont développé une approche basée sur la théorie des jeux de coalition pour un partage efficace des ressources dans un environnement *cloud* fédéré. Cette méthode concerne la stratégie de partage de la capacité qui peut conduire à une économie d'énergie. Le concept de jeu de coalition a été utilisé pour modéliser les interconnexions distinctes entre les fournisseurs. Cette méthode a permis d'améliorer les revenus des fournisseurs avec un faible coût de calcul. Ici, le SDN et la virtualisation des ressources du réseau pourrait être explorés pour utiliser efficacement la conception structurelle du réseau.

1.5.4 Allocation des ressources

1.5.4.1 Problématique

La quantité de ressources allouées à des applications multimédias hautes performances telles que le les jeux en ligne, ne cesse de croître dans les centres de données. La forte demande et les modèles d'utilisation de ces plateformes rendent l'allocation intelligente de ces ressources primordiale pour l'efficacité du cloud. Du placement des machines virtuelles (VM) aux GPU partagés, la communauté scientifique explore comment utiliser efficacement le cloud pour héberger des plateformes de cloud gaming. Nous étudions ci-dessous les travaux importants réalisés dans ce domaine afin de faciliter le déploiement efficace de plateformes de cloud gaming.

1.5.4.2 Travaux existants

Des travaux importants ont été réalisés à la fois sur le placement des machines virtuelles et sur l'ordonnancement du cloud afin de faciliter une meilleure qualité de service des jeux en ligne. Par exemple, Faheem Zafari et al. [78] ont développé une stratégie de partage des ressources pour atténuer les problèmes de pénurie de ressources au niveau du *cloud* edge. Cette méthode partage les ressources entre divers fournisseurs de services de *cloud* edge, où chaque fournisseur de services a une fonction spécifique à optimiser. La méthode d'optimisation multi objectif traite du problème d'allocation et de partage des ressources, en se basant sur la théorie des jeux coopérative, où chaque fournisseur de services en périphérie satisfait d'abord ses applications natives, puis partage ses ressources restantes. Bien que cette méthode ait amélioré l'utilisation de tous les fournisseurs de services, elle est couteuse en temps de traitement.

Yao Liu et al.[79] ont conçu une approche de codage vidéo améliorée pour le *cloud gaming*. Ce modèle a introduit deux concepts pour améliorer la qualité de la vidéo et diminuer le coût de calcul, respectivement. La qualité de la vidéo a été améliorée en introduisant un codage prioritaire basé sur le rendu pour améliorer la qualité vidéo perçue du jeu en fonction de la bande passante du réseau. La complexité de calcul de ce modèle a été réduite par un mécanisme d'encodage dépendant du rendu. Cette méthode a permis de réduire le temps de codage, mais elle dégrade légèrement la qualité de la vidéo.

Mohammad Sadegh-Aslanpour et al. [80] ont utilisé une approche de fourniture de ressources en fonction de l'apprentissage pour les jeux *cloud* massivement multi-joueurs, qui est l'intégration du modèle d'automates d'apprentissage et du paradigme de l'informatique

autonome. Le temps de réponse de cette méthode est faible malgré la faible performance de l'auto-dimensionnement des ressources. On pourrait intégrer le modèle MMOG aux approches d'ordonnancement des tâches, et évaluer l'approche proposée dans une infrastructure *cloud*, telle qu'OpenStack.

Mostafa Ghobaei-Araniet al. [81] ont proposé un cadre de provisionnement autonome des ressources pour les fournisseurs de MMOG. Cette méthode a été développée pour réduire le coût provisionnement des ressources inutilisées. Un système d'interférence neuro-floue adaptatif (ANFIS) est modélisé pour prédire la distribution de l'entité de jeu à partir des données de traces historiques dans le service d'estimation de charge. Ensuite, un algorithme d'arbre de décision flou est utilisé pour estimer le nombre approprié de ressources à allouer à chaque niveau de l'application MMOG en utilisant la charge de travail prédite et le SLA de l'utilisateur. Cette approche prédictive montre de bonnes performances en termes de précision et d'efficacité avec des données réelles ou synthétiques.

Anand Bhojan et al.[82] proposent une nouvelle architecture logicielle pour le développement des jeux vidéo *Cloud*yGame pour réduire le coût de l'initialisation et des ressources. Cette approche de *cloud*yGame implique deux fonctions. La première est que les ressources sont communiquées et gérées plus efficacement entre les différentes occurrences de jeu. La seconde est que les ressources du jeu sont disponibles à la demande afin de réduire les coûts. Cependant, cette méthode ne prend pas en compte un nombre élevé d'utilisateurs.

Wei Wei et al. [83] propose une approche d'allocation des ressources basée sur un jeu de Stackelberg à information imparfaite (CSAM-IISG) utilisant un modèle de Markov caché (HMM) dans un environnement *cloud*. Le modèle CSAM-IISG augmente le profit du fournisseur de ressources. Le modèle HMM a été utilisé pour prédire l'offre actuelle du fournisseur de services à travers les demandes antérieures des ressources.

Yang Liu et al. [84] ont développé une stratégie d'incitation pour l'estimation de la surcharge par le biais du *cloud* edge basée sur le jeu de stackelberg visant à maximiser les bénéfices de l'opérateur de services. Cette méthode a été développée pour fournir des calculs à la périphérie des réseaux à proximité des consommateurs mobiles. Bien que cette méthode ait réduit le délai et la complexité, elle n'offre pas de sécurité de réseau.

Xiangming Zhu et al. [85] ont conçu une approche d'allocation des ressources dans le *cloud*. Ce modèle comprend un jeu de stackelberg entre l'utilisateur et l'opérateur et minimise les problèmes de consommation d'énergie. La minimisation de l'énergie a été réalisée pour

l'allocation de la puissance et le calcul des ressources des utilisateurs. Cette méthode a permis d'obtenir un meilleur processus de tarification et d'allocation des ressources.

Zhengfa Zhu et al [86] ont proposé une approche pour réduire le coût des ressources dans le *cloud*. La question de la maximisation des revenus dans le jeu de stackelberg a été résolue par la combinaison des fournisseurs IaaS et SaaS. Les fournisseurs IaaS avaient la capacité de réduire le coût de la machine virtuelle pour générer plus d'opportunités. Les fournisseurs SaaS ont été employés pour réduire le nombre de ressources utilisées, ce qui a réduit le coût du modèle de jeu. Cette approche de jeu dynamique a diminué le coût du modèle à la fois par IaaS et SaaS. Cela pourrait être amélioré en choisissant la concurrence ainsi que la coopération entre les différents fournisseurs IaaS, ce qui donnera aux clients la possibilité de choisir des fournisseurs plus appropriés.

Seyed Javad Seyed Aboutorabi et Mohammad Hossein Rezvani [87] ont développé une approche pour les problèmes d'allocation des joueurs en utilisant l'algorithme d'optimisation Bees. Cette méthode a permis d'améliorer la rentabilité et de diminuer le gaspillage du côté serveur. Bien que cette méthode ait résolu les problèmes de rentabilité et d'allocation des ressources, elle n'est pas adaptée aux jeux en temps réel.

Qiang He et al. [88] ont proposé une méthode d'allocation EUAGame basée sur théorie des jeux pour les utilisateurs en utilisant une approche décentralisée. Cette méthode admet l'équilibre de Nash dans le jeu.

Yunhua et al.[89] ont proposé une méthode d'allocation de serveurs pour les jeux en ligne multi-joueurs. La complexité des approches de jeu en ligne basées sur plusieurs joueurs a été résolue grâce aux approches heuristiques. Cette méthode a permis de réduire le coût optimal ainsi que les paramètres de latence.

Yusen Li et al. [90] ont proposé un algorithme de répartition pour les requêtes de jeu des utilisateurs du *cloud gaming*. Le problème d'utilisation des ressources dû à la charge de travail maximale a été réduit en estimant l'heure de fin de toutes les sessions de jeu.

Mohaddeseh-Basiri et Abbas Rasoolzadegan [91] ont développé une approche d'allocation de ressources réduisant les coûts sans délai dans les jeux *cloud*.

Fangyuan Chi et al. [92] ont développé une approche du *cloud gaming* pour le problème d'attribution des tâches. Cette méthode fait appel à deux modules de traitement de la conception structurelle, tels que le téléchargement progressif des ressources de jeu et l'attribution de tâches

coopératives ad hoc dépendantes du mobile. La session de téléchargement progressif a été utilisée pour télécharger les ressources de jeu à partir des serveurs *cloud* ou des joueurs mobiles les plus proches. L'allocation de tâches coopératives ad-hoc dépendantes du mobile a été utilisée pour exécuter dynamiquement les dispositifs locaux, les plus proches et les serveurs *cloud*. La conception structurelle des deux modules a été conçue pour résoudre les problèmes d'optimisation.

Yuan Zhang et al. [93] ont proposé une méthode qui a permis de résoudre le problème de placement en ligne en influençant le contrôle prédictif de modèle (MPC) et en relevant les défis dans chaque fenêtre d'évaluation.

Abbas Soltanian et al. [94] ont développé une solution d'optimisation pour l'allocation des ressources et des échelles sur la base du nombre de joueurs tout en évaluant la qualité de service. Cette méthode est composée de diverses approches de traitement des médias pour répondre à la demande des participants. Cette méthode a formulé les problèmes d'allocation des ressources de manière analytique sous la forme d'une programmation linéaire en nombres entiers. Cette méthode a été utilisée pour différents nombres de joueurs et pour la distribution géographique des différents joueurs.

Mohaddeseh-Basiri et Abbas Rasoolzadegan [91] ont proposé un cadre d'allocation de ressources (RA) pour les centres de données, qui bénéficie de ressources CPU/GPU hautement virtualisées pour une meilleure rentabilité. À cette fin, un modèle précis de délai a été présenté, en tant que paramètre de contrôle principal de la qualité d'expérience, en tenant compte de toutes les sources de délai. Ensuite, un schéma RA minimisant les coûts en fonction du délai et une approche traçable pour dériver les limites de performance sur la probabilité de blocage du système. Les résultats de la simulation montrent que les performances des schémas proposés sont supérieures à celles des autres schémas en termes de rentabilité, tout en satisfaisant aux exigences de délai des utilisateurs connectés.

Yusen Li et al [95] ont développé un modèle d'apprentissage automatique efficace dans le domaine des jeux en ligne pour l'allocation ressources avec des contraintes de QoS. Deux approches d'allocation des ressources sont introduites, l'une est l'algorithme de force brute et l'autre les collocations de jeu. L'algorithme de force brute est utilisé pour estimer l'ensemble complet de toutes les colocations de jeu efficaces. Bien que cette méthode soit réalisable, le délai d'exécution de cette méthode est élevé.

Dans l'optique d'optimiser les plateformes de jeux en ligne, des auteurs ont utilisées des algorithmes d'optimisation. L'utilisation des algorithmes d'optimisation permet d'améliorer l'efficacité des systèmes *cloud* et ainsi la qualité d'expérience des utilisateurs. Récemment, Hossein Ebrahimi Dinaki et al. [96] ont développé deux approches efficaces pour la sélection des serveurs basés sur les GPU pour le *cloud gaming*. Leurs méthodes sont les versions améliorées de deux métaheuristiques bien connus à savoir l'optimisation par essaims de particules (PSO) et l'algorithme génétique (GA), appelée Bossted-PSO et Bossted-GA. Ces méthodes visent à améliorer les profits des fournisseurs de services et la QoE en maximisant l'utilisaton du GPU. Cette méthode a permis d'obtenir des résultats efficaces en termes d'utilisation de GPU (énergie) et de QoE bien qu'elle n'ait pas pris en compte des paramètres et métriques supplémentaires pour obtenir de meilleures solutions.

Dans [96], la méthode Boosted-PSO a obtenu l'efficacité maximale parmi les méthodes de résolution. Aussi, elle présente une bonne efficacité face à un grand nombre de joueurs. Cette méthode peut être étendue en incluant d'autres paramètres de qualité, ainsi que des contraintes de réseau, afin d'évaluer le problème sous des aspects supplémentaires et d'obtenir une solution exhaustive. L'un des moyens les plus efficaces pour les jeux en ligne multi-joueurs est l'utilisation d'approches d'apprentissage automatique, en particulier l'apprentissage par renforcement pour maximiser la rentabilité du côté serveur. Pour l'amélioration de ce modèle, on pourrait développer une approche basée sur le fog pour améliorer encore la qualité d'expérience dans les jeux.

Gongzhuang Peng et al. [97] ont proposé un algorithme génétique polyvalent pour obtenir une meilleure allocation des ressources. Cet algorithme génétique multi objectif basé sur l'optimisation s'est appuyé sur l'archive élitiste ainsi que sur les méthodologies K-means. Le modèle analytique du problème d'optimisation a été utilisé pour représenter l'approche difficile de l'allocation des ressources dans un environnement de prédiction multi-locataires en considérant la satisfaction des locataires basée sur les priorités, le coût total de calcul et l'équilibre de la charge à plusieurs niveaux. Cette méthode a permis de réduire le coût de calcul.

1.6 Conclusion

Ce chapitre présente un aperçu des différents concepts clés et de l’état de l’art des méthodes pour l’amélioration des services de *cloud gaming*. Il s'agit d'approches basées sur la QoE, la QoS, la consommation énergétique, l’allocation des ressources et d'approches utilisant des algorithmes et des métaheuristiques d'optimisation. Sur la base de l'étude que nous avons menée, nous allons présenter nos différentes contributions en vue d’améliorer l’allocation des ressources pour les services de *cloud gaming* dans les chapitres ci-dessous.

2 CHAPITRE 2 : Proposition d'une approche d'allocation de ressources multi-objectifs basée sur la QoE

Sommaire

2.1 Introduction

Dans le *cloud gaming*, l'allocation des ressources se base sur une négociation des contrats de service appelés SLA (Service Level Agreement). En effet, les fournisseurs de services doivent s'assurer de la disponibilité des ressources nécessaires pour exécuter les requêtes des utilisateurs et ainsi améliorer leurs expériences. Par conséquent, le fournisseur de services de jeux en ligne doit déterminer l'allocation optimale des ressources dans le but d'éviter le surdimensionnement ou le sous-dimensionnement et, optimiser les ressources pour les services différenciés. Ce genre de problème est généralement exprimé sous forme de "problème d'optimisation". Il s'agit en effet de maximiser ou minimiser un ou plusieurs objectifs représentés sous forme d'une fonction objectif en respectant certaines contraintes. Ici, le scénario d'allocation de ressources pourrait être modélisé comme étant un jeu dans lequel les joueurs sont les utilisateurs et les fournisseurs qui, selon le modèle du jeu choisi, peuvent adopter plusieurs stratégies. La théorie des jeux est un cadre d'étude permettant d'analyser des situations de concurrence entre des acteurs et de prendre des décisions optimales.

Dans ce chapitre, nous formulons le problème en un problème d'optimisation multi-objectif et proposons un algorithme d'optimisation hybride nommé RHSA (*Rider based HSA*) pour une bonne allocation des ressources et une amélioration de l'efficacité du système de *cloud gaming*. Notre approche se base sur la théorie des jeux pour répondre aux besoins de ressources des utilisateurs dans le *cloud gaming* et aider les fournisseurs de services à minimiser les coûts d'exploitation.

2.2 Problème d'optimisations combinatoires

2.2.1 Définition

« Combinatoire » est le terme général utilisé pour décrire des problèmes dont la solution est trouvée avec une explosion du nombre de combinaisons à étudier. D'un point de vue mathématique, l'optimisation combinatoire regroupe toutes les méthodes permettant de déterminer l'optimum d'une fonction objectif avec ou sans contraintes[98].
En théorie, un problème d'optimisation combinatoire est défini par l'ensemble de ses solutions, souvent infinies. En pratique, le problème se résume à résoudre numériquement l'une de ces solutions par un processus algorithmique. La modélisation d'un tel problème et sa solution à l'aide d'un programme informatique conduit à une automatisation du processus de prise de décision et, par conséquent, à une simplification de la tâche du décideur.

Le but est donc de trouver une configuration meilleure que toutes les autres (appelée configuration optimale), mais cela peut vite devenir très difficile car l'ensemble des options possibles a tendance à être très large dans la réalité. Pour un décideur, la forme de choix la plus simple est entre deux alternatives. Un tel choix, dit binaire, est modélisé avec une variable de décision qui ne peut prendre que les valeurs zéro ou un. Ensuite, la solution d'un problème de décision consiste à choisir l'une des deux alternatives pour l'ensemble des variables du problème. L'attribution d'une valeur à une variable peut avoir un impact direct sur d'autres variables. La solution obtenue est représentée par un ensemble de variables de décision. La fonction objectif du problème évalue la solution par rapport à une combinaison de valeurs associées à chacune des variables. Une fonction objectif peut être modélisée de manière linéaire ou non linéaire (par exemple, quadratique), elle suit généralement des contraintes sur les variables, ce qui réduit l'espace des solutions réalisables au problème. Ces contraintes définissent un ensemble de règles qui doivent être respectées avec ce qu'on appelle une solution réalisable. Le but ultime du problème est de trouver la meilleure solution possible en termes de fonction objectif, qui prend en compte toutes les conditions aux limites définies dans le problème. Par exemple, un voyageur doit choisir entre une autoroute et une route nationale. On peut représenter ce choix au moyen d'une variable de décision qui est un si le mode de transport choisi est l'autoroute et zéro si l'autoroute choisie est la route nationale. Les valeurs attribuées aux variables dans la fonction objectif peuvent correspondre, par exemple, à des préférences. On peut donc imaginer que la préférence pour l'autoroute en termes de confort, de sécurité et de vitesse est plus grande que celle de la route nationale. On peut aussi supposer que la distance parcourue est inférieure à celle d'une autoroute.

1. Optimisation linéaire : L'optimisation linéaire examine le cas où la fonction objectif et les conditions aux limites qui caractérisent l'ensemble sont linéaires. Il s'agit d'une méthode largement utilisée pour créer des programmes de raffinage du pétrole, mais aussi pour déterminer la composition la plus rentable d'un mélange de sels sous certaines conditions à partir des prix du marché.

2. Optimisation d'entiers linéaires : Dans le cas de l'optimisation d'entiers linéaires, il est examiné si certaines ou toutes les variables sont restreintes à accepter des valeurs d'entiers.

3. Optimisation d'entiers non linéaires : L'optimisation non linéaire examine le cas général où la cible ou les contraintes (ou les deux) contiennent des parties non linéaires, éventuellement non convexes**.**

4. Optimisation stochastique : l'optimisation stochastique examine le cas où certaines des conditions aux limites dépendent de variables aléatoires.

5. Optimisation quadratique : L'optimisation quadratique examine le cas où la fonction objectif a une forme quadratique (avec contraintes linéaires)

6. Optimisation multi-objectifs : C'est une optimisation multi-critères dans laquelle un compromis est recherché entre plusieurs objectifs contradictoires. La fonction objectif ne lie pas une valeur numérique, mais plutôt un point dans un ensemble qui est le plus souvent affecté à un vecteur. Le but est alors d'optimiser toutes les composantes de ce vecteur en même temps. On peut aussi voir l'optimisation multi-objectifs comme un ensemble de problèmes d'optimisation qui impliquent les mêmes paramètres, qui ont des objectifs potentiellement conflictuels, et que l'on veut résoudre le mieux possible.

2.2.2 Complexité des problèmes combinatoires

Le concept de problème combinatoire est formellement caractérisé par la théorie de la complexité. Elle propose une classification des problèmes selon la complexité de leur solution. Selon l'étude [99], La « complexité d'un problème » est une estimation du nombre d'instructions qui doivent être exécutées pour résoudre des instances de ce problème. Il s'agit d'une estimation du pire des cas dans laquelle la complexité d'un problème est définie en considérant son instance la plus difficile. En fait, il existe un grand nombre de classes différentes (problèmes à complexité constante, problèmes linéaires, L, NL, P, etc.[100]. La classe NP résume la plupart des problèmes d'optimisation combinatoire intéressants. Elle contient de nombreuses propriétés associées à des problèmes qui surviennent dans divers domaines, tels que problèmes de tournées de véhicules et ses variantes, placement de tâches, affectation de fréquences, et notamment le problème de sac à dos et ses variantes. Les problèmes les plus difficiles de NP définissent la classe des problèmes NP-complets.

2.2.3 Résolution des problèmes combinatoires

Avec un petit nombre de VM et de PM, il est possible pour un opérateur de pouvoir placer lui-même les VM sur l'infrastructure physique. Dans les cas où le nombre de VM et de PM augmente considérablement, la combinaison de la recherche des meilleurs emplacements physiques pour les VM devient trop importante. Un espace de solutions viables de cette dimension ne peut être épuisé en un temps raisonnable. Par conséquent, nous utiliserons des heuristiques intelligentes, en particulier pour les grandes instances, dont les solutions sont très proches de la solution optimale.

2.2.3.1 Approches complètes

Il existe plusieurs techniques exactes pour obtenir des solutions optimales pour des problèmes NP difficiles, mais elles sont plus efficaces pour des ensembles de problèmes avec des sous-ensembles. Cependant, ces techniques exactes ne sont pas très efficaces pour les grands ensembles de problèmes car leur temps de calcul varie de façon exponentielle avec leur taille. Parmi ces méthodes exactes, se trouvent des méthodes pour lister toutes les solutions possibles. Nous identifions également des techniques pour diviser intelligemment l'espace des solutions possibles afin de trouver la solution optimale. Ces solutions précises répondent aux objectifs à atteindre et aux contraintes à respecter. L'ensemble des restrictions à respecter permet de limiter l'ensemble des solutions possibles pour trouver celle qui maximise ou minimise la fonction objectif, en tenant compte des restrictions.

2.2.3.2 Approches incomplètes

Les métaheuristiques sont des techniques permettant d'optimiser la recherche de l'optimum sans avoir à effectuer une recherche exhaustive de toutes les solutions possibles. Une métaheuristique est une stratégie générale de résolution de problèmes qui utilise, coordonne et guide d'autres heuristiques pour parvenir à une solution à un problème difficile. Alors que les heuristiques sont généralement spécifiques à un problème, les métaheuristiques sont conçues pour s'adapter à n'importe quel problème. Une caractéristique commune à de nombreuses métaheuristiques est qu'elles s'inspirent de processus naturels, par exemple en physique ou en biologie. Les plus connus sont les algorithmes évolutionnaires et les algorithmes de colonies de fourmis. La littérature scientifique est riche en métaheuristiques et leurs applications. Par exemple, on peut citer les heuristiques de la recherche locale et de la recherche tabou (TS). Il existe également des heuristiques de colonies de fourmis, un recuit simulé, des recherches de voisinage variable (VNS), des essaims de particules et des algorithmes génétiques (GA). A partir de ces métaheuristiques, il est possible d'implémenter des heuristiques pour un problème NP difficile. Ces heuristiques permettent de trouver des solutions simples qui consomment peu de ressources et permettent de trouver des solutions très proches des optimales avec des délais raisonnables [101]. Notre méthode de résolution se base sur une heuristique dérivée de deux méta-heuristiques à savoir le ROA [102] et le HSA [103].

2.3 Proposition de l'algorithme RHSA

Le *cloud* offre la possibilité de fournir des services complexes sur les appareils des utilisateurs. Pour la diffusion de la vidéo, le *cloud* doit fournir davantage de ressources à mesure que le nombre de joueurs simultanés augmente. Cependant répondre aux exigences de qualité d'expérience avec une disponibilité limitée des ressources est un défi important pour les applications cloud. De nombreuses méthodes d'optimisation, telles que la consolidation de serveurs et la migration des machines virtuelles à des niveaux de calcul distincts, sont utilisées dans le cloud. Il est donc important d'évaluer l'impact de l'optimisation et de la mise à l'échelle automatique des ressources. Par conséquent, les limites de l'allocation des ressources conventionnelle dans les services basés sur le *cloud* sont considérées comme motivation pour développer un nouveau modèle de migration pour les applications hébergées dans le *cloud* telles que les jeux en ligne.

2.3.1 Modèle du système

Le cloud gaming donne aux utilisateurs la possibilité de jouer à des jeux complexes via une connexion haut débit. Ces jeux sont joués sans qu'il soit nécessaire d'installer ou de télécharger un logiciel de jeu. De plus, l'utilisateur n'a plus à mettre à jour constamment le matériel. De nombreux jeux peuvent être joués avec de faibles coûts de matériel et de logiciel. Les fournisseurs de services de jeux d'argent utilisent des centres de données géographiquement dispersés pour offrir leurs services à leurs utilisateurs. Lorsque les demandes sont reçues par l'infrastructure de jeu, elles sont exécutées par une machine virtuelle (VM) au sein d'un centre de données spécifique en utilisant une méthode spécifique. Le modèle cloud alloue des ressources aux tâches utilisateur pour une période spécifique. L'allocateur ou le courtier de ressources assure la synchronisation entre les utilisateurs et les fournisseurs de services cloud [104]. Les ressources de la machine virtuelle utilisent différentes configurations avec différents types de stockage, d'alimentation et de mémoire. L'allocation des ressources étant un critère de qualité pour les fonctions cloud, la moindre perte de performance rend la plateforme cloud inefficace. Par conséquent, il est important de concevoir une méthode d'allocation des ressources qui permette une répartition optimale des tâches entre les machines virtuelles.

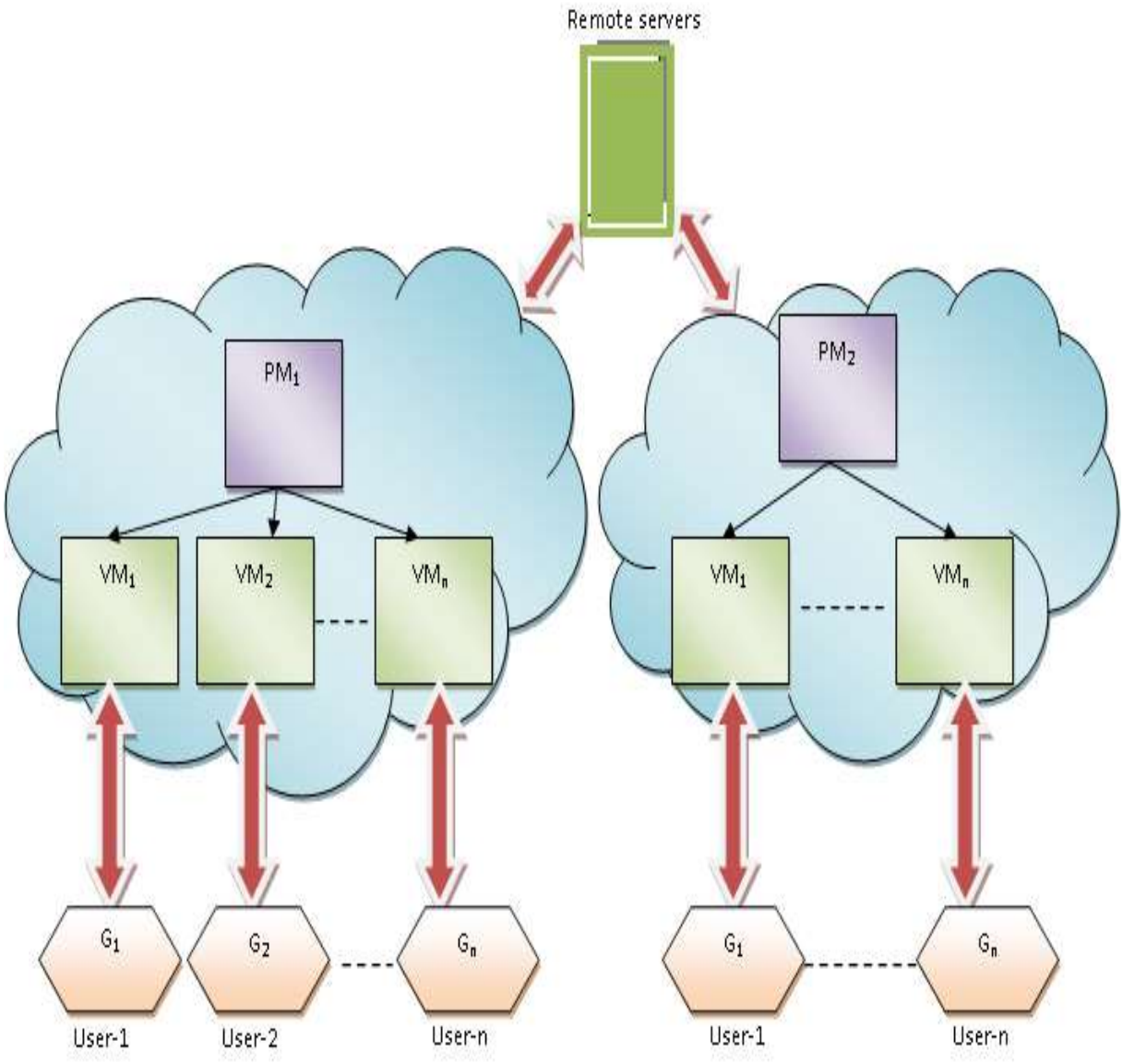

Figure 2. 1 Modèle de système du *Cloud*

Le modèle développé pour l'allocation des ressources dans l'environnement cloud est illustré à la figure 3.1 ci-dessus. Le but de ce modèle est de déterminer les ressources optimales qui peuvent être affectées à chaque jeu demandé par l'utilisateur.

Considérons une plateforme *cloud* composée de h machines physiques PMs (Physical Machines). Ces PMs sont représentés par $M = \{M_1, M_2, \cdots, M_g, \cdots M_h\}; 1 \leq g \leq h$, et chaque PM contient plusieurs machines virtuelles VMs (Virtual Machines). Les VMs dans le g^{th} PM sont modélisées par $N = \{N_1, N_2, \cdots, N_k, \cdots, N_l\}; 1 \leq k \leq l$ où l est le nombre total de VMs à un instant donné.

Les requêtes de jeux de chaque utilisateur sont allouées aux VMs de manière *round-robin* et sont représentés par $G = \{G_1, G_2, \dots, G_p, \dots, G_q\}$ où q est le nombre total de jeux alloués aux VMs. Les joueurs sont représentés par $U = \{U_1, U_2, \dots, U_p, \dots, U_q\}$.
Dans le cloud gaming, les jeux vidéo 3D sont traités à distance depuis des datacenters distants en temps réel via une connexion Internet. La résolution vidéo est une mesure essentielle pour évaluer la qualité des services de cloud gaming. Les débits binaires de chaque image sont exprimés comme suit $B = \{B_1, B_2, \dots, B_p, \dots, B_q\}$. Pour chaque jeu, la fréquence FPS de l'image vidéo est $V = \{V_1, V_2, \dots, V_p, \dots, V_q\}$. La QoE des utilisateurs précédents est définie par $Q = \{Q_1, Q_2, \dots, Q_s, \dots, Q_t\}$.

Chaque VM sélectionnée est configurée en fonction de certains paramètres, tels que la bande passante, les processeurs, la mémoire, les millions d'instructions par seconde (MIPS), et est modélisée comme suit :

$$N_{g,k} = \{R_{g,k}, Y_{g,k}, C_{g,k}, P_{g,k}\} \quad (2.1)$$

Où $R_{g,k}$ est le nombre de processeurs, $Y_{g,k}$ la mémoire, $C_{g,k}$la bande passante, $P_{g,k}$ le nombre MIPS, de la k^{th} VM au sein de la g^{th} PM. Les paramètres $R_{g,k}, Y_{g,k}, C_{g,k}, P_{g,k}$ ont une valeur comprise entre 1 et une constante f.

Les ressources (*resources units*) de chaque jeu sont modélisés par :

$$K = \{K_1, K_2, \dots, K_p, \dots, K_q\} \quad (2.2)$$

Le niveau de préférence de l'utilisateur UPL est compris entre [0, 1], où 1 représente une forte préférence. On considère T comme le temps d'initialisation de la VM.
Nos valeurs appartiennent à l'ensemble des nombres entiers naturels.

2.3.2 Optimisation de l'allocation des ressources avec l'algorithme RHSA

Les utilisateurs de jeux en ligne finissent par entrer en conflit et tout le monde essaie d'obtenir toutes les ressources nécessaires pour une très bonne expérience lors de l'exécution des sessions de jeu. La théorie des jeux peut être utilisée pour analyser la concurrence pour les ressources entre différents joueurs jouant à différents jeux. Par conséquent, nous proposons un modèle de jeu en ligne avec optimisation des ressources afin de déterminer les solutions optimales pour résoudre le problème d'allocation des ressources. L'objectif de l'allocation des ressources avec

une technique d'optimisation est de sélectionner les tâches appropriées pour une ressource particulière. Notre algorithme RHSA a été créé en combinant le *Rider Optimization Algorithm* ROA [102] et le *Harmony Search Algorithm* HSA [103].

Le ROA est un algorithme d'optimisation récemment proposé basé sur l'inspiration d'un groupe de coureurs se dirigeant vers un objectif commun. Chacun des coureurs veut être le leader de la course. Ils sont répartis en quatre (04) groupes, à savoir *attaker* l'attaquant, *overtaker* le dépasseur, *follower* le suiveur, *bypass rider* le coureur de détour, et chaque groupe a une stratégie pour atteindre l'objectif. Le ROA est très efficace et suit les étapes du calcul fictif pour résoudre les problèmes d'optimisation [105][106][107], mais il présente une convergence moins bonne et est sensible aux hyperparamètres. Il offre de hautes performances dans des espaces de recherche inconnus et peut résoudre des problèmes avec ou sans contraintes

HSA, une métaheuristique utilisée pour résoudre des problèmes dans divers domaines, s'inspire du processus d'improvisation des musiciens à la recherche d'un parfait état d'harmonie. Cet algorithme s'appuie sur la musique pour trouver la meilleure harmonie, c'est-à-dire la mélodie la plus joyeuse lorsqu'un musicien joue une partition. Si l'on associe la résolution de cet algorithme avec un problème d'optimisation, on peut supposer que l'harmonie correspond au vecteur solution et que l'improvisation du musicien correspond à la recherche locale et globale de l'algorithme. Le HSA prend en compte tous les vecteurs existants et génère de nouveaux vecteurs. Cette caractéristique de l'algorithme HSA augmente la flexibilité et conduit à de meilleurs résultats. Il offre de meilleures performances pour trouver l'espace de solution à un coût raisonnable. L'algorithme HSA offre une vitesse de convergence améliorée et une grande précision. En outre, l'algorithme HSA offre de meilleurs compromis entre les tendances d'exploitation et d'exploration. Il est appliqué pour résoudre les problèmes d'optimisation de l'ingénierie.

La modification du ROA avec les propriétés HSA permet une meilleure efficacité avec une convergence plus rapide et la possibilité d'éviter les minima locaux. Par conséquent, le délai est réduit et l'efficacité du système est améliorée. Ainsi, la combinaison de HSA et de ROA est réalisée pour obtenir une solution globale optimale.

Notre fonction objectif (fonction *fitness*) est basée sur certains paramètres de qualité d'expérience, à savoir l'indice d'équité, l'expérience du joueur (QE) et le score d'opinion moyen (MOS). La figure 3.2 montre un schéma fonctionnel d'allocation de ressources utilisant l'algorithme RHSA.

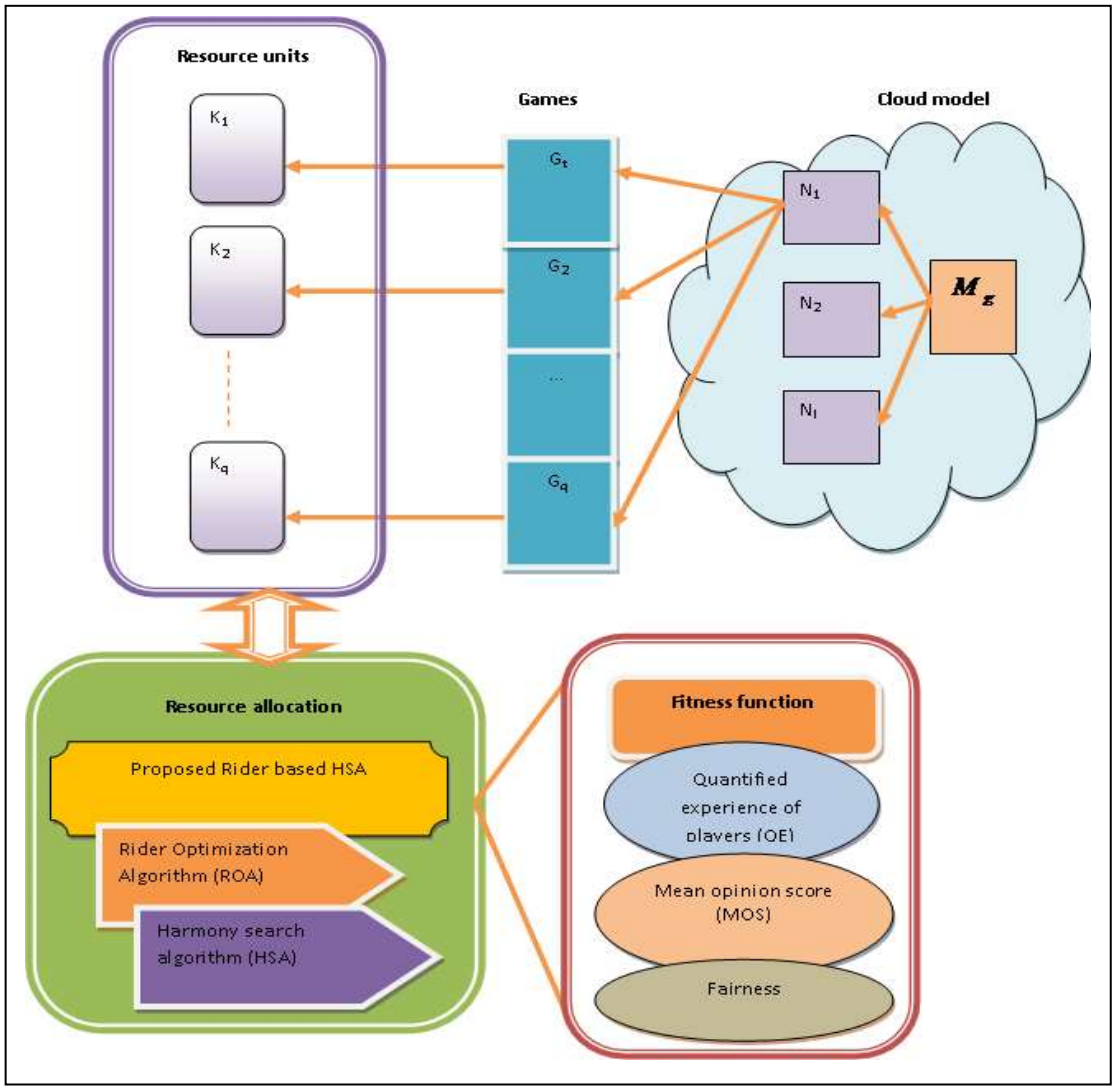

Figure 2. 2 Schéma fonctionnel du modèle d'allocation des ressources utilisant le RHSA

2.3.2.1 Illustration de la solution

La modélisation d'une solution est impérative pour déterminer la solution optimale de résolution des problèmes d'optimisation. La méthode RHSA est conçue pour nous aider à sélectionner la bonne solution en choisissant les jeux et machines virtuelles optimaux parmi les solutions possibles. L'optimisation détermine la valeur optimale parmi les solutions qui composent l'ensemble de solutions fourni en tant que valeur arbitraire. Pour l'allocation des ressources, l'ensemble de solutions forme un ensemble optimal de jeux. Considérons un scénario de sept jeux comme illustré à la figure 3.3. Premièrement, le vecteur solution est aléatoire. Ensuite, selon la fonction de fitness, le vecteur de solution est assemblé à partir des VM optimales.

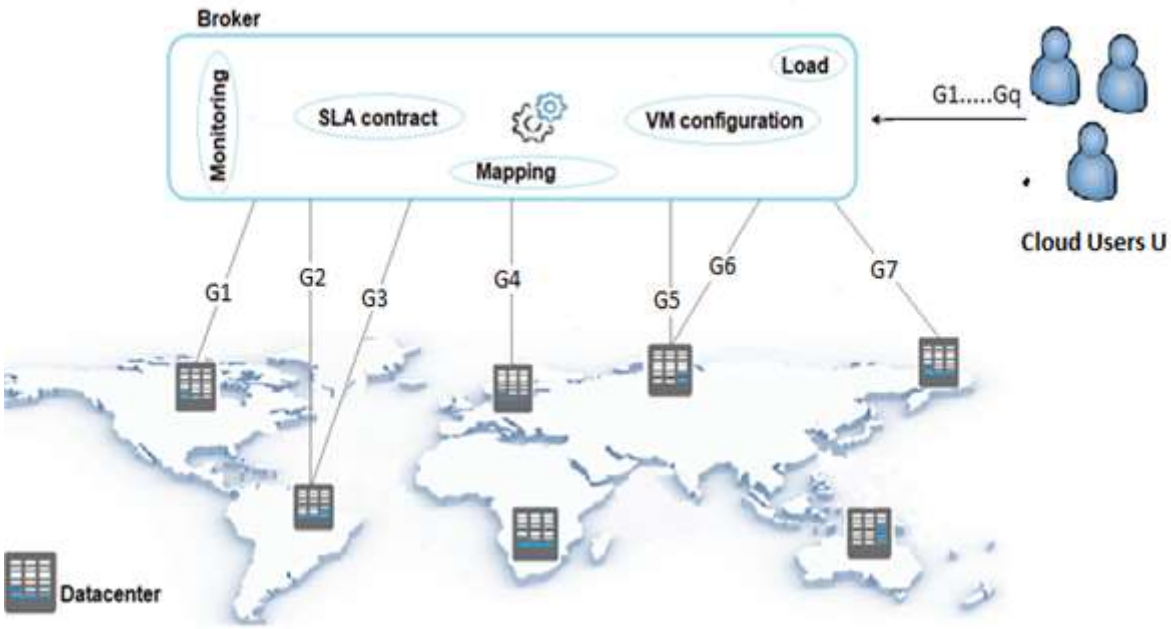

Figure 2. 3 Illustration de la solution pour l'allocation des ressources

Notre but est de proposer un modèle d'allocation qui traite de manière optimale les requêtes des utilisateurs. Nos propositions permettent de sélectionner les VMs appropriées au sein des datacenters (DC) d'un fournisseur de services *cloud* pour l'exécution de la requête de l'utilisateur tout en garantissant la QoE des utilisateurs.

SLA Constract module : Il permet de définir le contrat SLA de chaque requête. Ce module reçoit en entrée l'ensemble des requêtes et se charge d'étudier leurs différentes contraintes. En sortie, des contrats SLA sont établis pour chaque requête afin de définir les machines virtuelles pour le traitement de l'information.

Configuration VM : Il se charge de la configuration des ressources virtuelles en fonction des besoins de la requête. Il reçoit comme entrée les contraintes SLA et tente de trouver la bonne configuration de la machine virtuelle pour chaque requête. Il produit en sortie une liste de VM avec leurs caractéristiques.

Mapping module : Ce module est chargé de trouver le bon emplacement (DC) de la machine virtuelle configurée et de faire le mapping entre chaque requête et le Datacenter adéquat. Il reçoit en entrée les contrats SLA et la configuration des VM des sous-requêtes et fournit en sortie un vecteur de placement des VM.

Monitoring module : il interagit de façon continue avec tous les modules en temps réel afin d'éviter la violation du contrat SLA et veiller à la qualité de service.

Load manager module : Ce module nous permet d'évaluer la charge des DCs et la charge générée par les requêtes.

2.3.2.2 Modèle multi-objectif

La fonction de fitness est évaluée pour déterminer la meilleure solution parmi l'ensemble des solutions. On présente notre formule comme suit,

$$F = p(1 - QE) + pMOS + pJ^T \tag{2.3}$$

où, QE représente l'expérience des joueurs, MOS signifie le score d'opinion moyen, et J^T l'équité. F est une fonction à maximiser dans laquelle QE doit être minimale, MOS et l'équité J^T doivent être maximales. $p = 1/3$ Poids de chaque critère

Le MOS [67]est formulé par,

$$MOS = \frac{1}{l}\sum_{k=1}^{l}(B_k * L_k + V_k * L_k) \tag{2.4}$$

où, B_k symbolise le débit binaire du jeu, V_k la fréquence d'images vidéo du jeu s'exécutant dans la k^{th} VM, et L_k les paramètres des ressources.

Les paramètres de ressources pour améliorer la QoS sont exprimés comme suit,

$$L_k = \frac{1}{4}\left(\frac{X^R + X^Y + X^C + X^P}{max(X^R, X^Y, X^C, X^P)}\right) \tag{2.5}$$

où, X^R est le nombre de processeurs, X^Y la quantité de mémoire utilisée, X^C la bande passante et X^P symbolise le nombre MIPS utilisés pour l'allocation des ressources.

En intégrant le délai, le nombre d'images par seconde et la résolution, on obtient la perte d'expérience de jeu [65] du joueur, qui est l'objectif de chaque joueur par

$$QE = \mu_1 U - \mu_2 W^l - \mu_3 N_{Qual} \tag{2.6}$$

où, μ_1, μ_2 et μ_3 indique des paramètres constants, U symbolise le délai, W^l le nombre d'images par seconde (FPS), et N_{Qual} la qualité vidéo du jeu.

La perte d'expérience de jeu QE doit être minimale

Les VM utilisant les jeux sont créées et détruites de manière dynamique, et il devrait y avoir un délai de clonage parmi les utilisateurs p qui représente le délai d'initialisation du service. En accumulant les jeux dans le centre de données, la vitesse d'écriture sur le disque dur est

modélisée par O. Si un joueur choisit un jeu avec une taille de fichier x_k, le délai total [65] est modélisé sur la base du temps d'initialisation de la VM et s'exprime comme suit,

$$U = \sum_{k=1}^{l} \frac{x_k}{O} + T_k \tag{2.7}$$

où, x_k est la taille du fichier pour le jeu dans la k^{th} VM, O symbolise la vitesse d'écriture du disque dur, et T_k signifie le temps pris pour l'initialisation de la VM.

Le nombre d'images par seconde FPS [65] est une mesure clé de la qualité d'expérience pour les utilisateurs de jeux vidéo. En général, le FPS est déterminé à l'aide de l'unité de traitement graphique (GPU), de la mémoire vive (RAM) et de l'unité centrale de traitement (CPU) du serveur physique.

Dans le *cloud gaming*, le FPS est exprimé comme suit,

$$W^l = \sum_{k=1}^{l} \frac{\omega_1}{1+e^{\omega_2\left[\frac{1}{5}\sum_{k=1}^{l}\vartheta_k + L_k\right]+\omega_3}} L_k \tag{2.8}$$

où, ω_1, ω_2 et ω_3 indiquent les paramètres d'approximation.

La résolution ou qualité vidéo du jeu [65]est exprimée par,

$$N = \sum_{k=1}^{l} \left(\frac{\omega}{v} log_2 \left(1 + \frac{\ell N_k}{\omega_o + \sum_{k=1}^{\ell} \ell N_k} \right) \right) \tag{2.9}$$

où ω, ℓ et ω_o sont des constantes, et N_k représente la résolution de la vidéo du jeu dans la j^{th} VM.

L'équité est exprimée par,

$$J^T = \frac{1}{l \times t} \sum_{k=1}^{l} \left(\sum_{s=1}^{t} Q_s \right) * UPL \tag{2.10}$$

où, UPL symbolise le niveau de préférence de l'utilisateur et Q signifie les scores de qualité d'expérience donnés par les joueurs.

2.3.2.3 Description de l'algorithme RHSA

L'allocation des ressources se fait à l'aide d'un nouvel algorithme d'optimisation, à savoir l'algorithme RHSA dérivé de la combinaison de l'algorithme ROA [102] et de l'algorithme HSA[103].

Les étapes de la méthode RHSA sont décrites ci-dessous :

Étape 1 : Initialisation des coureurs et des autres paramètres algorithmiques

L'initialisation est effectuée à l'aide de quatre groupes de coureurs *attaker* l'attaquant, *overtaker* le dépasseur, *follower* le suiveur, *bypass rider* le coureur de détour. Les initialisations des positions sont effectuées de manière aléatoire de la manière suivante,

$$D_n = \{D_n(u,v)\}; 1 \le u \le H, 1 \le v \le I \tag{2.11}$$

où, H représente le nombre de coureurs, et $D_n(u,v)$ exprime la position du u^{th} coureur dans la v^{th} dimension au n^{th} instant, et I symbolise la dimension totale.

Étape 2 : Evaluation de la fonction fitness :
La solution de la fonction fitness est évaluée à l'aide de l'équation (3.3)

Étape 3 : Déterminer le leader

La fonction de fitness est considérée comme la partie déterminante de la découverte d'un coureur leader. Le coureur près de la destination est supposé avoir la valeur fitness la plus élevée, et ce coureur est le leader.

Étape 4 : Mettre à jour la position des coureurs

La position de chaque coureur dans un ensemble est mise à jour pour découvrir le coureur de tête et donc le leader. Ainsi, la position du coureur est mise à jour en tenant compte des caractéristiques de chaque coureur.

Ci-dessous, la mise à jour de la position de chaque coureur selon le ROA [102] :

Le *follower* le suiveur a tendance à mettre à jour sa position en utilisant la position du coureur leader pour atteindre la cible plus rapidement et s'exprime comme suit,

$$D_{n+1}^{f}(u,r) = D^{S}(S,r) + \left[Cos\left(Z_{u,r}^{n} * D^{S}(S,r) * y_u^r\right)\right] \tag{2.12}$$

où, r est le sélecteur de coordonnées, D^S indique la position du coureur de tête, S représente l'indice du coureur de tête, $Z_{u,r}^n$ représente l'angle de direction du u^{th} coureur en r^{th} coordonnées, et y_u^n est la distance.

La position du *overtaker* le dépasseur est utilisée pour maximiser la fonction fitness,

$$D_{n+1}^o(u,r) = D_n(u,r) + [A_n(u) * D^S(S,r)] \quad (2.13)$$

où, $A_n(u)$ symbolisent l'indicateur de direction du u^{th} cavalier à l'heure n .

Attaker l'attaquant présente un potentiel pour s'emparer de la position de leader en utilisant la règle de mise à jour du leader. Elle est exprimée comme suit :

$$D_{n+1}^a(u,v) = D^S(S,v) + \left[CosZ_{u,v}^n * D^S(S,v)\right] + y_u^n \quad (2.14)$$

Le *bypass rider* le coureur de détour utilise un chemin spécifique sans suivre le leader. Ainsi, la règle de mise à jour est

$$D_{n+1}(u,v) = \eta\left[D_n(\alpha,v) * \rho(v) + D_n(\gamma,v) * [1-\rho(v)]\right] \quad (2.15)$$

où, η signifie un nombre aléatoire parmi 1 à H , α signifie un nombre arbitraire qui va de 1 à H et ρ exprime un nombre arbitraire parmi 0 et 1.

Considérons $\alpha = u,$ alors l'équation est exprimée comme,

$$D_{n+1}(u,v) = \eta\left[D_n(u,v) * \rho(v) + D_n(\gamma,v) * [1-\rho(v)]\right] \quad (2.16)$$

Chaque unité produite avec la mémoire est calculée pour découvrir si elle doit être ajustée en hauteur. La mise à jour HSA [103] est exprimée comme suit :

$$D_{n+1}(u,v) = D_n(u,v) \pm rand(0,1).E \quad (2.17)$$

où, $rand(0,1)$ est un nombre aléatoire dans [0, 1], E représente la bande passante

$$D_n(u,v) = D_{n+1}(u,v) \pm rand(0,1).E \quad (2.18)$$

Le *bypass rider* le coureur de détour emprunte un chemin familier sans suivre le leader. HSA améliore la position de mise à jour du bypass rider. En substituant l'équation (2.18) dans (2.16),

$$D_{n+1}(u,v) = \eta\left[(D_{n+1}(u,v) \pm rand(0,1).E) * \rho(v) + D_n(\gamma,v) * [1-\rho(v)]\right] \quad (2.19)$$

$$D_{n+1}(u,v) = \eta D_{n+1}(u,v) * \rho(v) + \eta[(D_n(\gamma,v) * [1-\rho(v)] \pm rand(0,1).E) * \rho(v)] \quad (2.20)$$

$$D_{n+1}(u,v) - \eta D_{n+1}(u,v) * \rho(v) = \eta[(D_n(\gamma,v) * [1-\rho(v)] \pm rand(0,1).E) * \rho(v)] \quad (2.21)$$

$$D_{n+1}(u,v)[1-\eta\rho(v)] = \eta[(D_n(\gamma,v) * [1-\rho(v)] \pm rand(0,1).E) * \rho(v)] \quad (2.22)$$

$$D_{n+1}(u,v) = \frac{\eta[(D_n(\gamma,v) * [1-\rho(v)] \pm rand(0,1).E) * \rho(v)]}{1-\eta\rho(v)} \quad (2.23)$$

Étape 5 : Déterminer le taux de réussite :

Après avoir terminé la mise à jour, la fonction fitness de chaque coureur est évaluée. Ainsi, la position du coureur qui est en tête est remplacée par la nouvelle position du coureur obtenue jusqu'à ce que la fonction fitness du nouveau coureur soit plus élevée.

Étape 6 : Temps d'arrêt

La procédure est sans cesse répétée jusqu'au temps d'arrêt, au cours duquel le leader est déterminé. Après avoir terminé la course, le leader est considéré comme le gagnant.

Le pseudo-code de l'algorithme RHSA proposé est illustré dans le tableau 3.1.

Tableau 2. 1 Pseudo-code de l'algorithme RHSA

***Input**: D_n :Random position of rider, n : iteration, $n_{\max}$:maximum iteration*

***Output**: Leading rider D^S*

Begin

Initialize the set of solutions.

Initialize other parameter of rider like gear, brake, accelerator and steering angle

Determine fitness function using equation (2.3)

While $n < N_{OFF}$

For $u = 1\ to\ H$

Update bypass rider's position with equation (2.23)

Update follower position with equation (2.12)

Update overtaker position with equation (2.13)

Update attacker position with equation (2.14)

Rank riders using fitness function with equation (2.3)

Choose the rider with high fitness function

Update gear, brake, accelerator and steering angle

Return D^S

$n = n + 1$

End for

End while

End

2.4 Résultats et discussions

Dans cette section, est évaluée l'efficacité de l'algorithme RHSA sur la base des différentes métriques que nous avons considéré à savoir l'équité, le MOS et la QE.

L'évaluation est faite en considérant différentes tailles de tâches 100, 200 et 300 et aussi en variant la taille de la population de 10 à 50 par pas de 10 arbitrairement. Par la suite, l'approche a été comparée à d'autres approches de la littérature.

Nos différentes expériences ont été exécutées en utilisant le simulateur cloudSim [108] sous PYTHON avec un PC équipé du système d'exploitation Windows 10, d'un processeur Intel i7 et de 12 Go de RAM.

Le nombre de machines physiques considérés est égal à 6, le nombre total de machines virtuelles est équivalent au nombre de tâches. Ici, la tâche à exécuter est la requête de jeux.

2.4.1 Mesures de performances du RHSA

Les différentes métriques à savoir l'équité, l'expérience du joueur et le MOS sont évalués individuellement avec le modèle que nous proposons afin de montrer son efficacité.

Nous présentons ci-dessous les résultats obtenus en faisant varier la taille de tâches et la taille de la population.

2.4.1.1 Evaluation du RHSA avec une taille de tâche 100

2.4.1.1.1 Evaluation du RHSA sur l'équité avec une taille de tâches 100

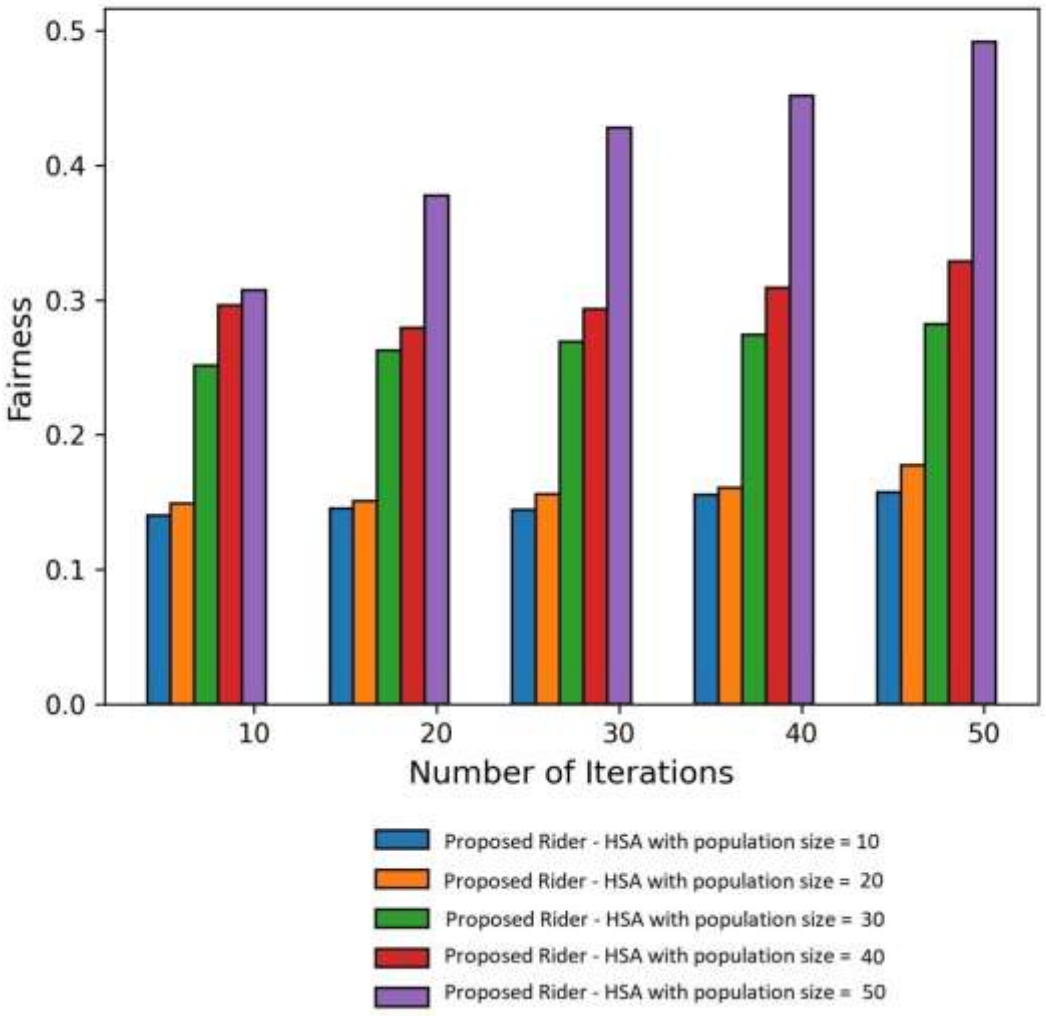

Figure 2. 4 Evaluation du RHSA sur l'équité avec une taille de tâche 100

La figure 2.4 ci-dessus présente l'évaluation de notre algorithme RHSA sur l'équité à partir d'une taille de tâche égale à 100. Nous avons fait varier la taille de la population de 10 à 50. La couleur bleue représente une taille de population égale à 10, la couleur orange une taille de population égale à 20, la couleur verte une taille de population égale à 30, la couleur rouge foncé une taille de la population égale à 40 et la couleur violette une taille de la population égale à 50.

Pour 10 itérations, l'équité évaluée par le RHSA avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.140, 0.149, 0.251, 0.296 et 0.307. Ces résultats montrent que plus la taille de la population augmente, la valeur de l'équité augmente.

De même pour 20,30,40, 50 itérations, nous observons que la valeur de l'équité est améliorée avec l'augmentation de la taille de la population. Nous remarquons que ces valeurs augmentent aussi avec l'augmentation du nombre d'itérations.

Le graphique montre que l'équité du modèle proposé peut être améliorée lorsque le nombre d'itérations augmente. Cela suppose que si nous disposons d'un grand nombre de ressources à allouer de manière optimale, l'augmentation du nombre d'itérations entrainera une amélioration de la valeur de l'équité.

Nos résultats de l'impact de notre algorithme RHSA sur l'équité, montrent que l'amélioration de la valeur de l'équité est directement proportionnelle à la taille de la population et du nombre d'itérations.

2.4.1.1.2 Evaluation du RHSA sur le MOS avec une taille de tâche 100

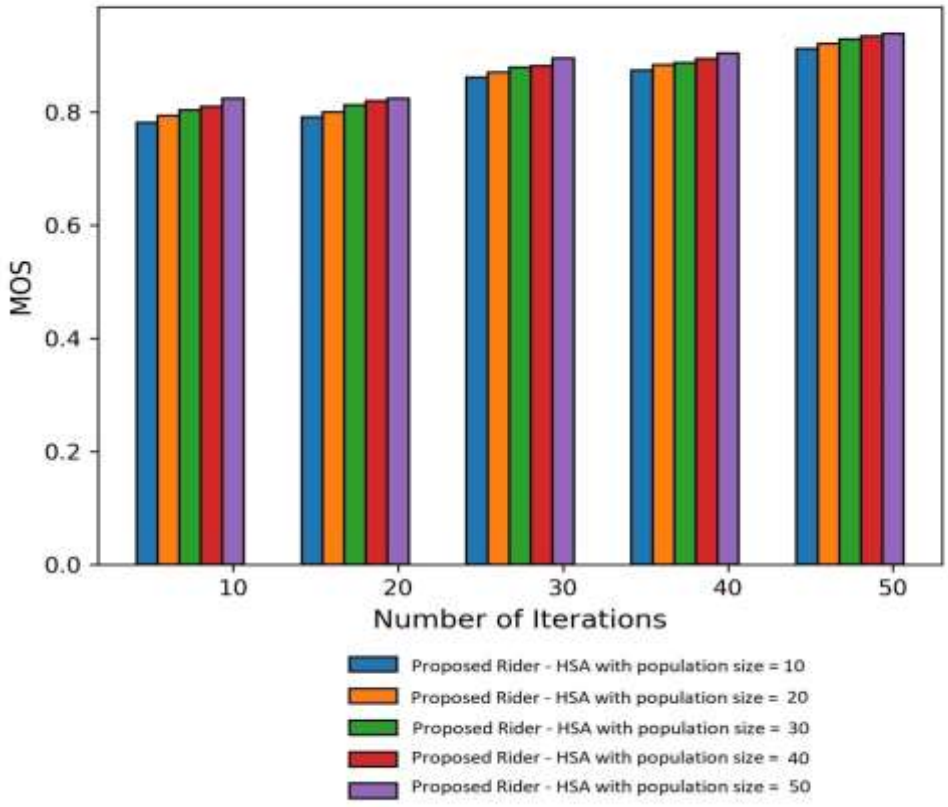

Figure 2. 5 Evaluation du RHSA sur le MOS avec une taille de tâche 100

La figure 2.5 présente l'évaluation du RHSA sur le MOS à partir d'une taille de tâche égale à 100. Nous avons fait encore varier la taille de la population de 10 à 50.

Pour 10 itérations, le MOS évalué avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.781, 0.794, 0.804, 0.810 et 0,825. Ces résultats montrent que plus la taille de la population augmente, la valeur du MOS aussi augmente.

Pour 20, 30, 40, 50 itérations, nous avons les mêmes conclusions que précédemment à savoir que les valeurs du MOS sont améliorées avec l'augmentation de la taille de la population et celle du nombre d'itérations.

Nous arrivons à la même conclusion que précédemment. Le graphique montre que l'augmentation du nombre d'itérations entraîne une augmentation de la valeur MOS. Les résultats ici montrent que l'amélioration de la valeur du MOS est directement proportionnelle à la taille de la population et du nombre d'itérations.

2.4.1.1.3 Evaluation du RHSA sur la QE avec une taille de tâche 100

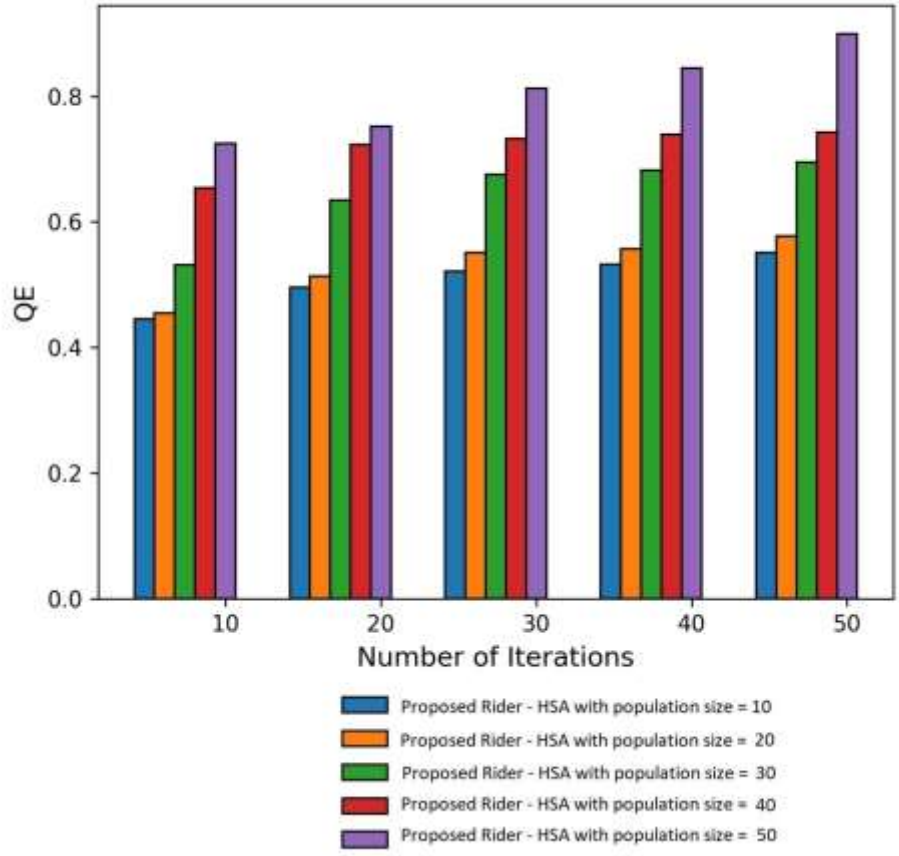

Figure 2. 6 Evaluation du RHSA sur la QE avec une taille de tâche 100

La figure 2.6 montre l'évaluation du RHSA sur l'expérience du joueur QE à partir d'une taille de tâche égale à 100. La taille de la population varie de 10 à 50.

Pour 10 itérations, la QE évaluée avec une population de taille 10, 20, 30, 40 et 50 est de 0.445, 0.455, 0.532, 0.654, 0.723 et 0.725 respectivement. D'après la figure, la valeur de QE augmente en augmentant le nombre d'itérations et la taille de la population.

Pour 20, 30, 40, 50 itérations, nous observons que l'expérience QE est encore améliorée avec l'augmentation de la taille de la population et du nombre d'itérations.

Nos résultats montrent que l'amélioration de la valeur de QE est directement proportionnelle à la taille de la population et du nombre d'itérations.

Dans la section ci-dessous, nous évaluons les performances de notre algorithme RHSA avec une taille de tâche plus grande, deux fois supérieure à la taille précédente.

2.4.1.2 Evaluation du RHSA avec une taille de tâche 200

2.4.1.2.1 Evaluation du RHSA sur l'équité avec une taille de tâche 200

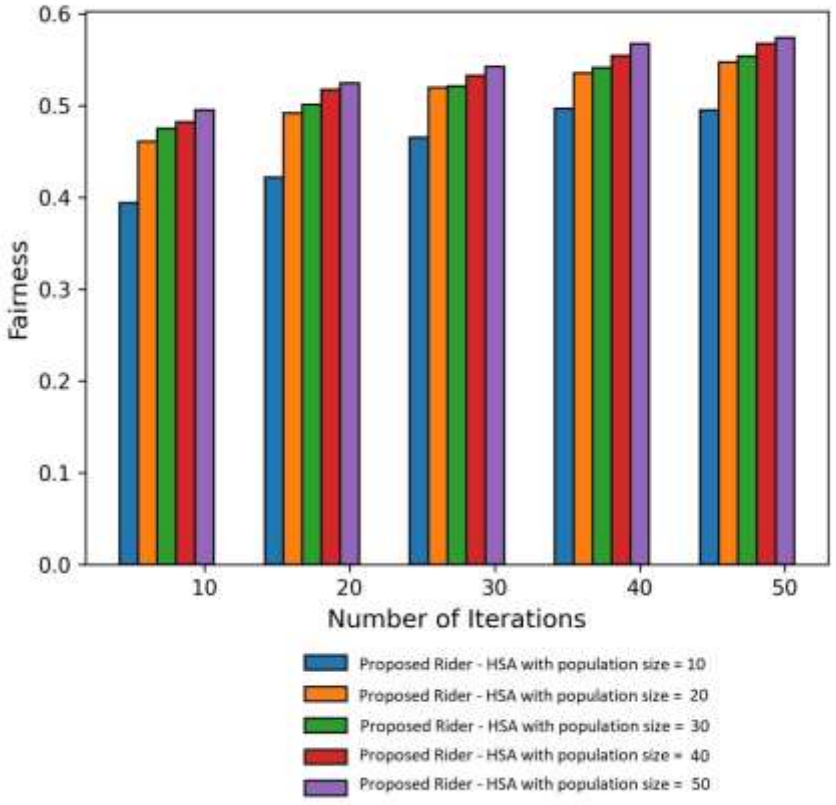

Figure 2. 7 Evaluation du RHSA sur l'équité avec une taille de tâche 200

La figure 2.7 ci-dessus présente l'évaluation du RHSA sur l'équité à partir d'une taille de tâche égale à 200. Nous avons maintenu la variation de la taille de la population de 10 à 50. Nos itérations varient de 10 à 50 par pas de 10.

Pour 10 itérations, l'équité évaluée par le RHSA avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.394, 0.461, 0.475, 0.482, et 0.495. On observe que plus la taille de la population augmente, la valeur de l'équité augmente.

De même pour 20,30,40, 50 itérations, nous observons l'augmentation des valeurs de l'équité avec l'augmentation de la taille de la population.

Cela suppose que si nous disposons d'un grand nombre de ressources à allouer de manière optimale, l'augmentation du nombre d'itérations entrainera une amélioration de la valeur de l'équité.

Avec l'augmentation de la taille de tâches, ici deux fois supérieur que précédemment, nous observons que notre algorithme permet d'avoir de meilleures valeurs. L'algorithme que nous proposons est évolutif.

2.4.1.2.2 Evaluation du RHSA sur le MOS avec une taille de tâche 200

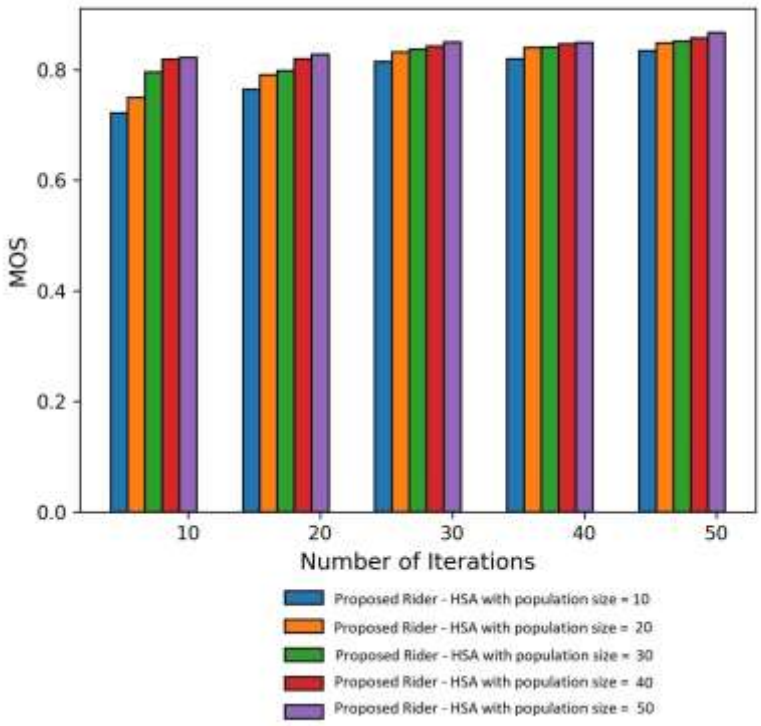

Figure 2. 8 Evaluation du RHSA sur le MOS avec une taille de tâche 200

La figure 2.8 ci-dessus présente l'évaluation du RHSA sur le MOS à partir d'une taille de tâche égale à 200. Nous avons maintenu la variation de la taille de la population de 10 à 50. Nos itérations varient de 10 à 50 par pas de 10.

Pour 10 itérations, le MOS évalué par le RHSA avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.722, 0.750, 0.795, 0.819, et 0.822. Lorsque la taille de la population augmente, la valeur du MOS augmente.

De même pour 20,30,40, 50 itérations, nous observons l'augmentation des valeurs de MOS avec l'augmentation de la taille de la population.

Cela suppose que si nous disposons d'un grand nombre de ressources à allouer de manière optimale, l'augmentation du nombre d'itérations entrainera une amélioration de la valeur MOS.

Avec l'augmentation de la taille de tâches, nous observons que notre algorithme permet d'avoir de meilleures valeurs. L'algorithme que nous proposons est évolutif.

2.4.1.2.3 Evaluation du RHSA sur le QE avec une taille de tâche 200

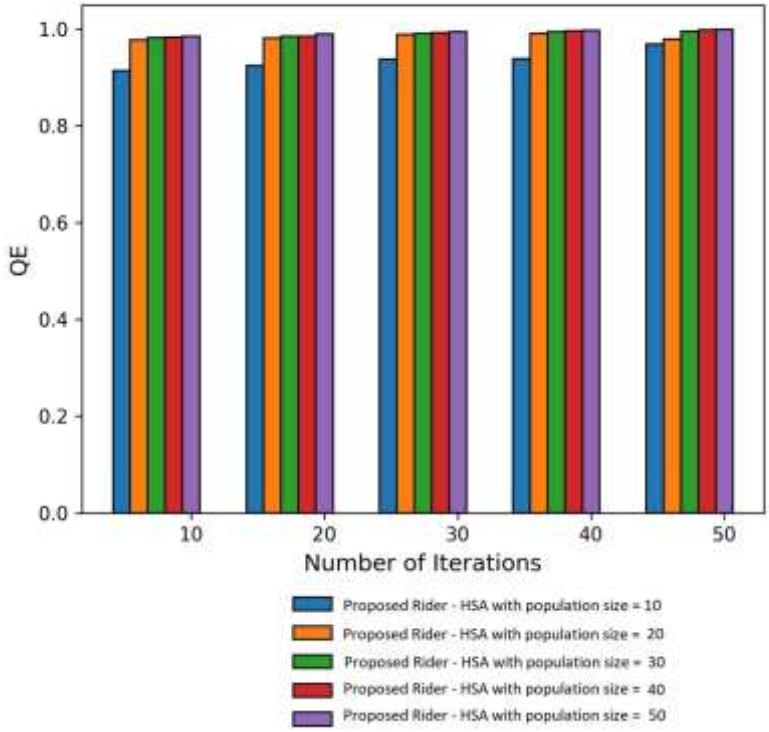

Figure 2. 9 Evaluation du RHSA sur le QE avec une taille de tâche 200

La figure 2.9 est l'évaluation du RHSA sur l'expérience QE à partir d'une taille de tâche égale à 200. Nous avons maintenu la variation de la taille de la population de 10 à 50. Nos itérations vont de 10 à 50.

Pour 10 itérations, la QE évaluée par le RHSA avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.914, 0.977, 0.982, 0.983, et 0.985. Quand la taille de la population augmente, la valeur de QE augmente. De même pour 20,30,40, 50 itérations, nous observons l'augmentation des valeurs de l'équité avec l'augmentation de la taille de la population.

Cela suppose que si nous disposons d'un grand nombre de ressources à allouer de manière optimale, l'augmentation du nombre d'itérations entrainera une amélioration de la valeur de QE.

L'algorithme que nous proposons est évolutif. Il permet d'obtenir de bons résultats avec l'augmentation de la taille des tâches.

Nous poursuivons notre évaluation de la performance du RHSA avec encore une taille plus grande 300.

2.4.1.3 Evaluation du RHSA avec une taille de tâche 300

2.4.1.3.1 Evaluation de la méthode sur l'équité avec une taille de tâche 300

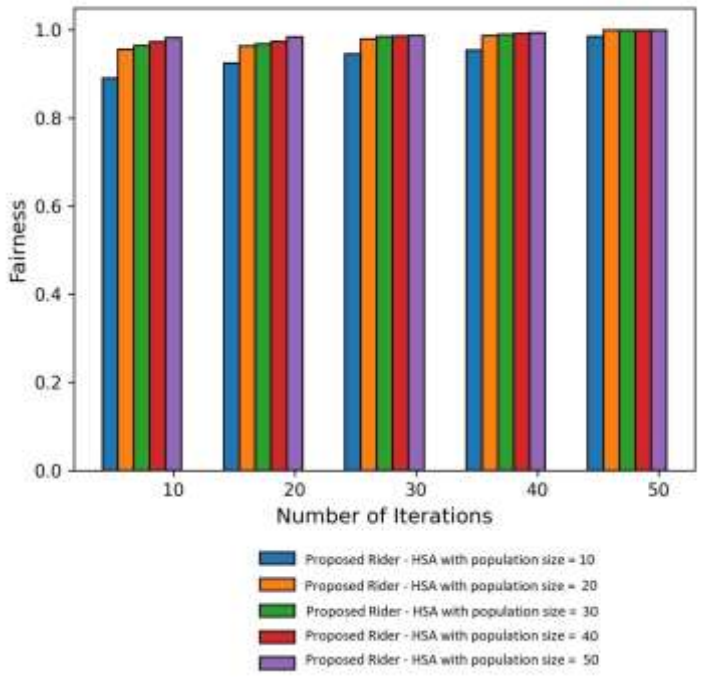

Figure 2. 10 Evaluation de la méthode sur l'équité avec une taille de tâche 300

La figure 2.10 ci-dessus présente l'évaluation du RHSA sur l'équité à partir d'une taille de tâche égale à 300. Nous avons une variation de la taille de la population de 10 à 50 et nos itérations varient de 10 à 50.

Pour 10 itérations, l'équité évaluée par le RHSA avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.891, 0.956, 0.965, 0.972, et 0.982. On observe que plus la taille de la population augmente, la valeur de l'équité augmente.

De même pour 20,30,40, 50 itérations, nous observons l'augmentation des valeurs de l'équité avec l'augmentation de la taille de la population.

Cela suppose que si nous disposons d'un grand nombre de ressources à allouer de manière optimale, l'augmentation du nombre d'itérations entrainera une amélioration de la valeur de l'équité.

Avec l'augmentation de la taille de tâches, nous observons que notre algorithme permet d'avoir encore de meilleures valeurs. L'algorithme que nous proposons est évolutif.

2.4.1.3.2 Evaluation de la méthode sur le MOS avec une taille de tâche 300

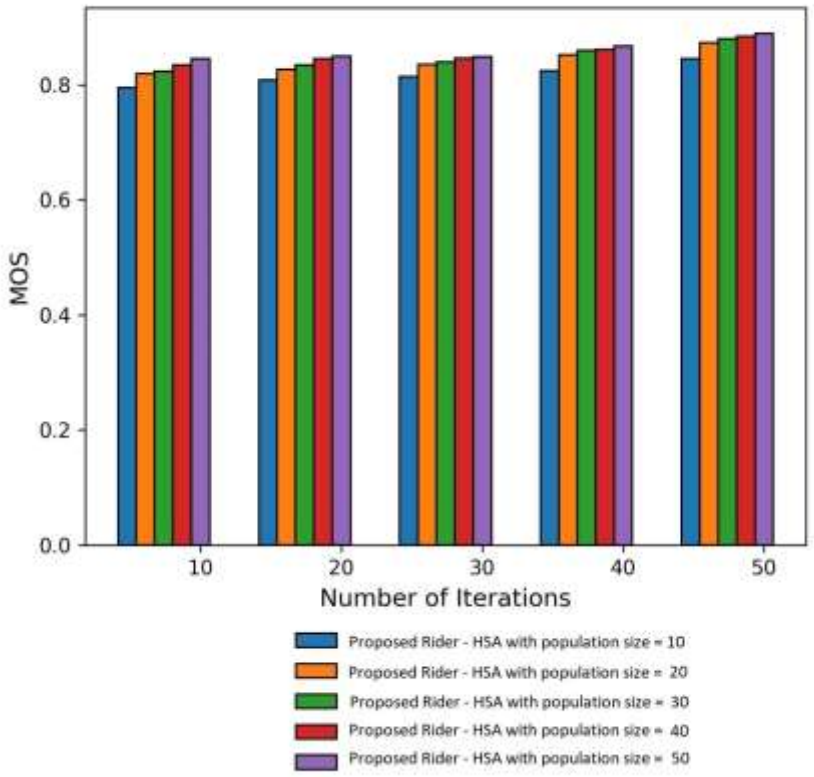

Figure 2. 11 Evaluation de la méthode sur le MOS avec une taille de tâche 300

La figure 2.11 présente l'évaluation du RHSA sur l'équité à partir d'une taille de tâche égale à 300. Nous avons une variation de la taille de la population de 10 à 50 et nos itérations varient de 10 à 50.

Pour 10 itérations, l'équité évaluée par le RHSA avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.891, 0.956, 0.965, 0.972, et 0.982. On observe que plus la taille de la population augmente, la valeur du MOS augmente.

De même pour 20,30,40, 50 itérations, nous observons l'augmentation des valeurs du MOS avec l'augmentation de la taille de la population.

Cela suppose que si nous disposons d'un grand nombre de ressources à allouer de manière optimale, l'augmentation du nombre d'itérations entrainera une amélioration de la valeur de MOS.

Avec l'augmentation de la taille de tâches, nous observons que notre algorithme permet d'avoir encore de meilleures valeurs MOS. L'algorithme que nous proposons est évolutif.

2.4.1.3.3 Evaluation de la méthode sur la QE avec une taille de tâche 300

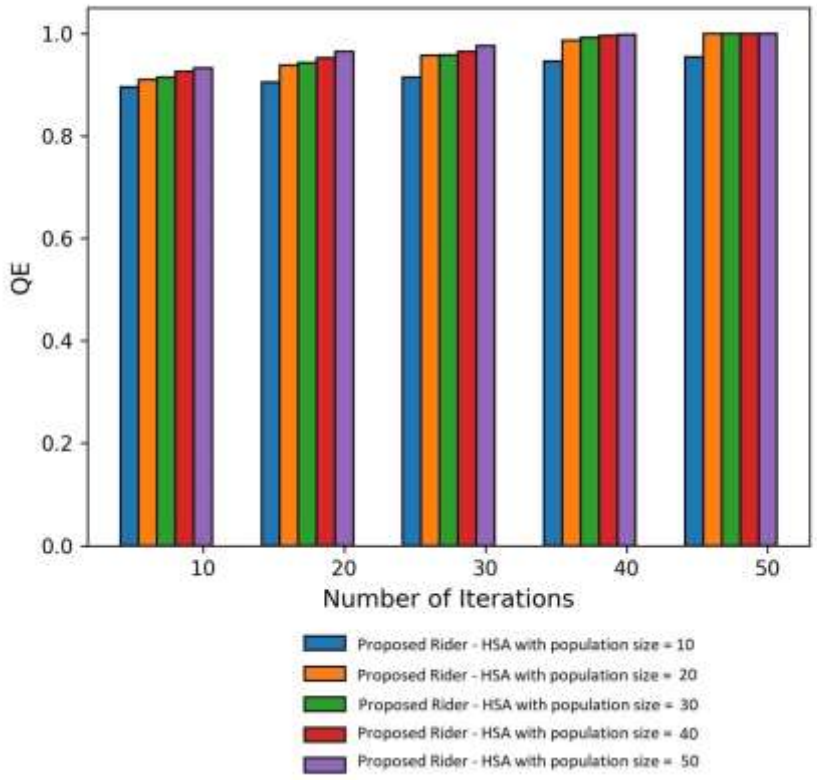

Figure 2. 12 Evaluation de la méthode sur la QE avec une taille de tâche 300

La figure 2.12 est l'évaluation du RHSA sur l'expérience QE à partir d'une taille de tâche égale à 300. Nous utilisons une taille de la population de 10 à 50 et une variation du nombre d'itérations de 10 à 50.

Pour 10 itérations, la QE évaluée par le RHSA avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.895, 0.910, 0.915, 0.925, and 0.932. Quand la taille de la population augmente, la valeur de QE augmente. De même pour 20,30,40, 50 itérations, nous observons l'augmentation des valeurs de l'équité avec l'augmentation de la taille de la population.

L'algorithme que nous proposons est évolutif. Il permet d'obtenir de bons résultats de l'expérience avec de grande taille des tâches.

En sommes, nos résultats montrent que lorsque la taille de la population et le nombre d'itérations augmentent, les différentes métriques du modèle augmentent également. Nos métriques sont directement proportionnelles à la taille de la population et au nombre d'itérations. Notre modèle réagit aussi bien avec l'augmentation du nombre de requêtes à placer. Dans la suite, nous continuons notre évaluation en comparant notre approche avec d'autres approches de la littérature qui traitent du même sujet.

2.4.2 Analyse Comparative

Dans cette section, nous comparons notre approche avec différentes approches de la littérature. Les méthodes utilisées pour la comparaison sont l'algorithme d'optimisation basé sur le jeu potentiel [65], l'algorithme d'allocation proactive des ressources [66], l'algorithme d'allocation des ressources tenant compte de la qualité de l'expérience[67], l'algorithme distribué [69], Yusen Li et al [95].

2.4.2.1 Analyse comparative avec une taille de tâche 100

2.4.2.1.1 Analyse comparative du RHSA sur l'Equité avec une taille de tâche 100

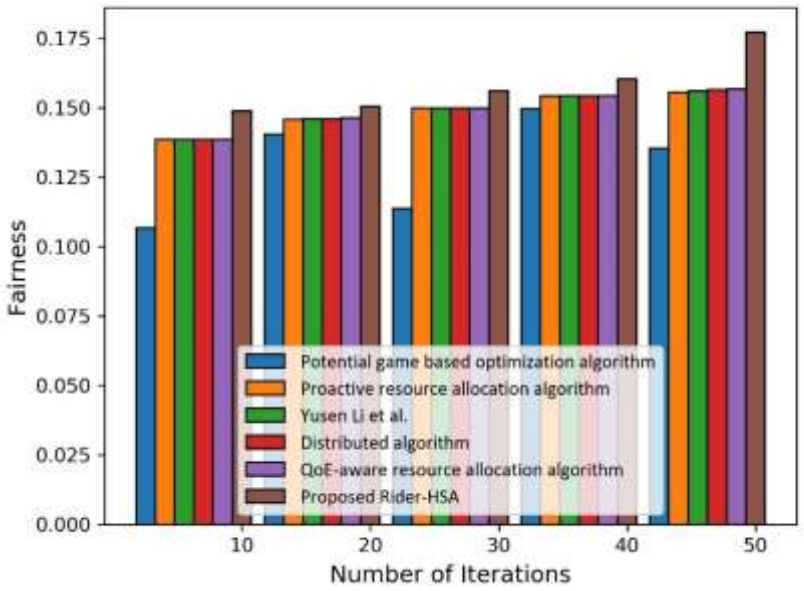

Figure 2. 13 Analyse comparative du RHSA sur l'Equité avec une taille de tâche 100

La figure 2.13 présente l'évaluation des méthodes avec l'équité à partir d'une taille de tâche égale à 100. Nous fixons la taille de la population à 50.
Nous comparons notre contribution de couleur marron avec la méthode *QoE-aware ressource allocation algorithm* de couleur violet, la méthode *distributed algorithm* de couleur rouge foncé, la méthode de *Yusen Li et al* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, et la méthode *potential game based optimization algorithm* de couleur bleue.

Notre contribution RHSA *Rider based HSA* présentent les meilleures valeurs de l'équité avec les différentes variations du nombre d'itération.

2.4.2.1.2 Analyse comparative du RHSA sur MOS avec une taille de tâche 100

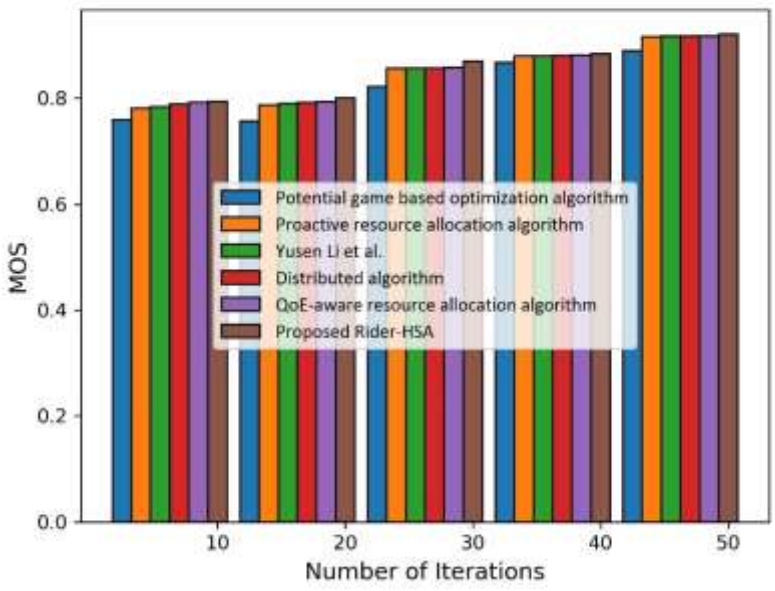

Figure 2. 14 Analyse comparative du RHSA sur MOS avec une taille de tâche 100

La figure 2.14 est l'évaluation des méthodes avec le MOS à partir d'une taille de tâche égale à 100.

Notre méthode *Rider based HSA* présentent les meilleures valeurs de MOS avec les différentes variations du nombre d'itération.

2.4.2.1.3 Analyse comparative du RHSA sur la QE avec une taille de tâche 100

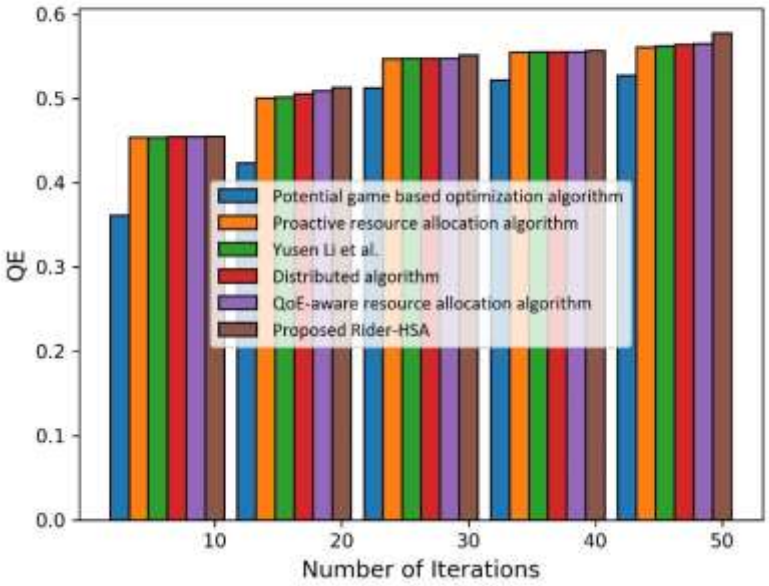

Figure 2. 15 Analyse comparative du RHSA sur la QE avec une taille de tâche 100

La figure 2.15 est l'évaluation des méthodes avec le QE à partir d'une taille de tâche égale à 100. Notre méthode *Rider based HSA* présentent les meilleures valeurs de QE.

Nous poursuivons l'analyse comparative de notre méthode avec le double de la taille précédente.

2.4.2.2 Analyse comparative avec une taille de tâche 200

2.4.2.2.1 Analyse comparative du RHSA sur l'équité avec une taille de tâche 200

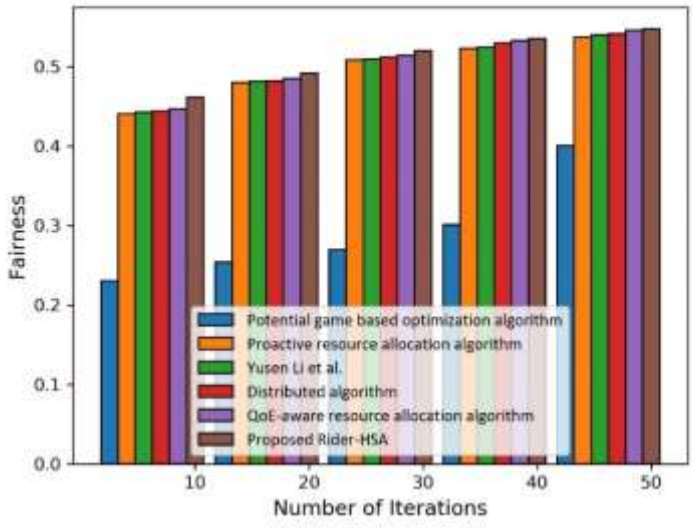

Figure 2. 16 Analyse comparative du RHSA sur l'équité avec une taille de tâche 200

La figure 2.16 est l'évaluation des méthodes avec le l'équité à partir d'une taille de tâche égale à 200. Notre méthode *Rider based HSA* présentent ici les meilleures valeurs de l'équité.

2.4.2.2.2 Analyse comparative du RHSA sur le MOS avec une taille de tâche 200

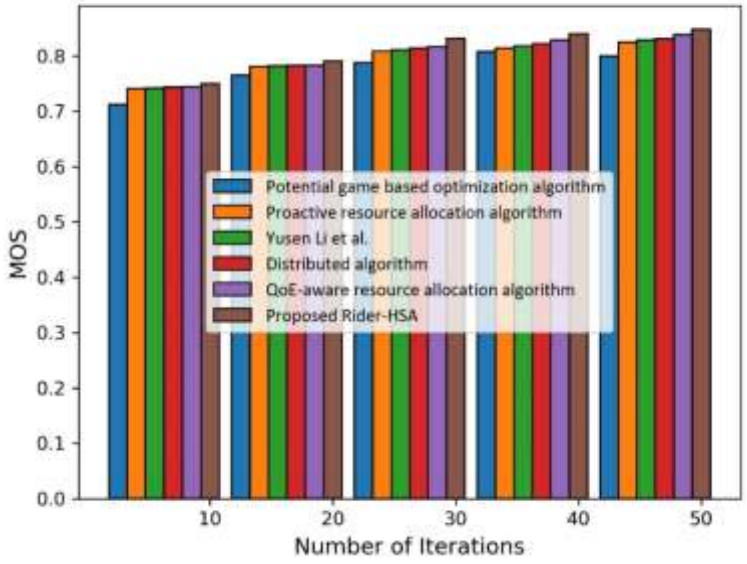

Figure 2. 17 Analyse comparative du RHSA sur le MOS avec une taille de tâche 200

La figure 2.17 est l'évaluation des méthodes avec le MOS à partir d'une taille de tâche égale à 200. Notre méthode *Rider based HSA* présentent les meilleures valeurs de MOS.

2.4.2.2.3 Analyse comparative du RHSA sur la QE avec une taille de tâche 200

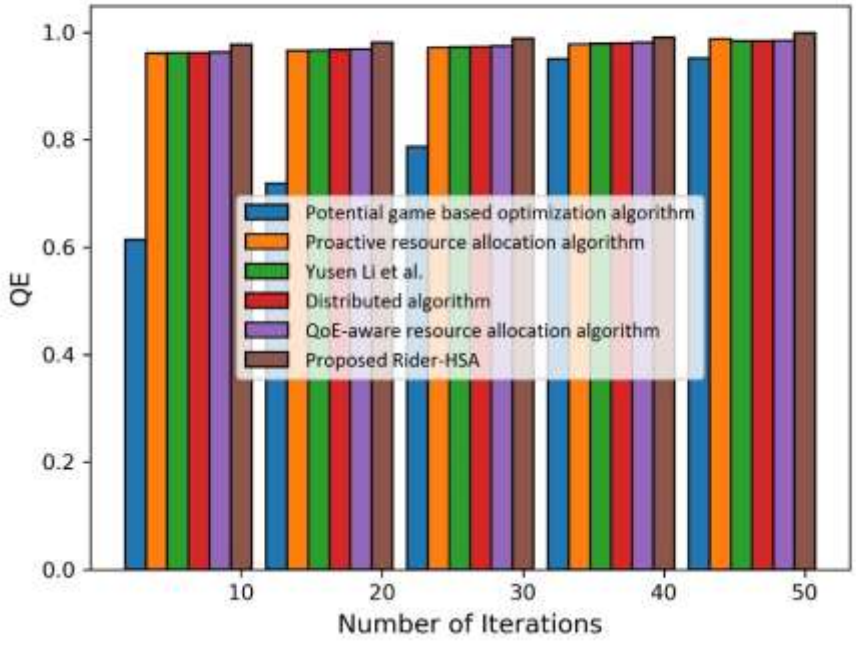

Figure 2. 18 Analyse comparative du RHSA sur la QE avec une taille de tâche 200

La figure 2.18 est l'évaluation des méthodes avec l'expérience QE à partir d'une taille de tâche égale à 200. Notre méthode *Rider based HSA* présentent les meilleures valeurs de QE.

Dans la section ci-dessous, nous utilisons encore une taille plus grande 300 pour continuer l'analyse comparative

2.4.2.3 Analyse comparative avec une taille de tâche 300

2.4.2.3.1 Analyse comparative du RHSA sur l'équité avec une taille de tâche 300

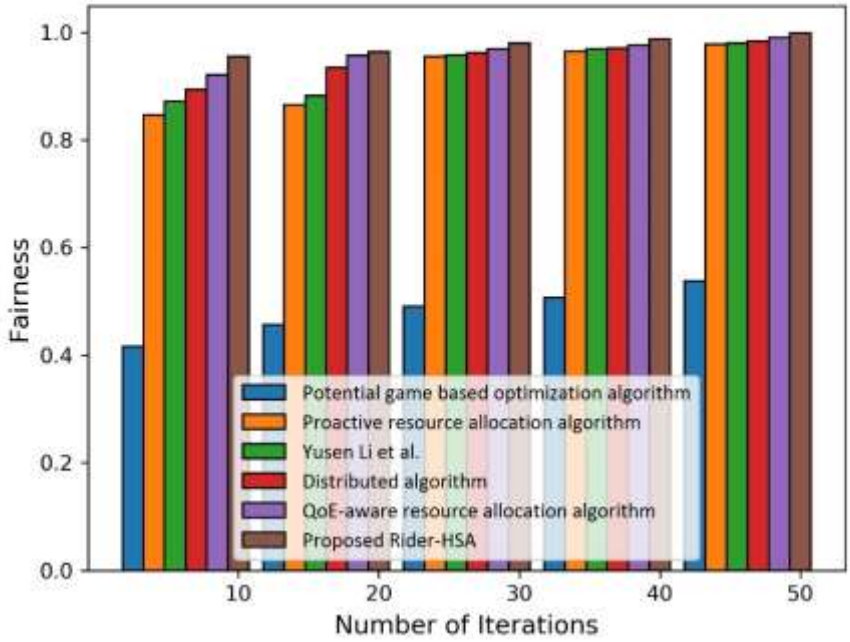

Figure 2. 19 Analyse comparative du RHSA sur l'équité avec une taille de tâche 300

La figure 2.19 est l'évaluation des méthodes sur l'équité à partir d'une taille de tâche égale à 300. Notre méthode *Rider based HSA* présentent les meilleures valeurs de l'équité.

2.4.2.3.2 Analyse comparative du RHSA sur le MOS avec une taille de tâche 300

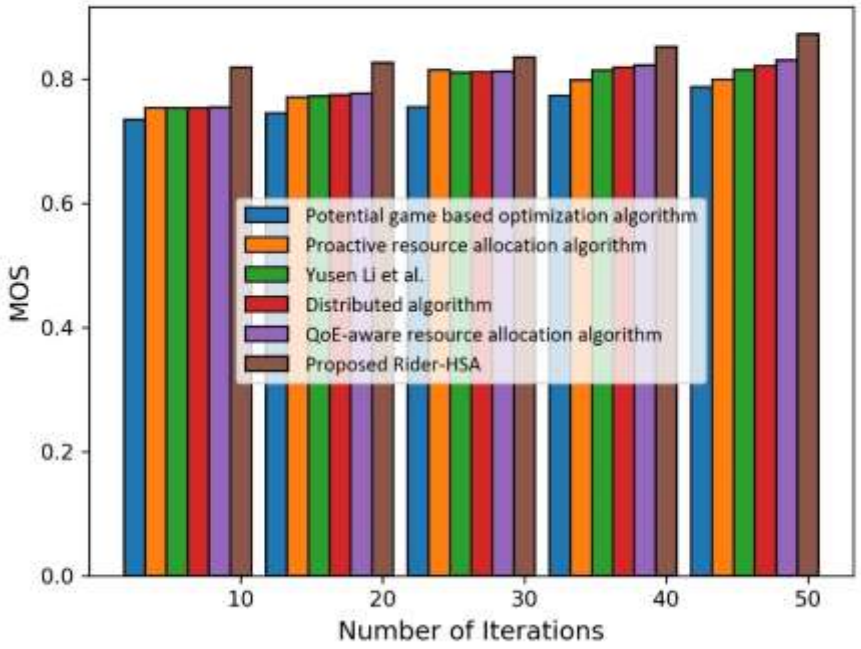

Figure 2. 20 Analyse comparative du RHSA sur le MOS avec une taille de tâche 300

La figure 2.20 est l'évaluation des méthodes sur le MOS à partir d'une taille de tâche égale à 300. Notre méthode *Rider based HSA* présentent les meilleures valeurs de MOS.

2.4.2.3.3 Analyse comparative du RHSA sur la QE avec une taille de tâche 300

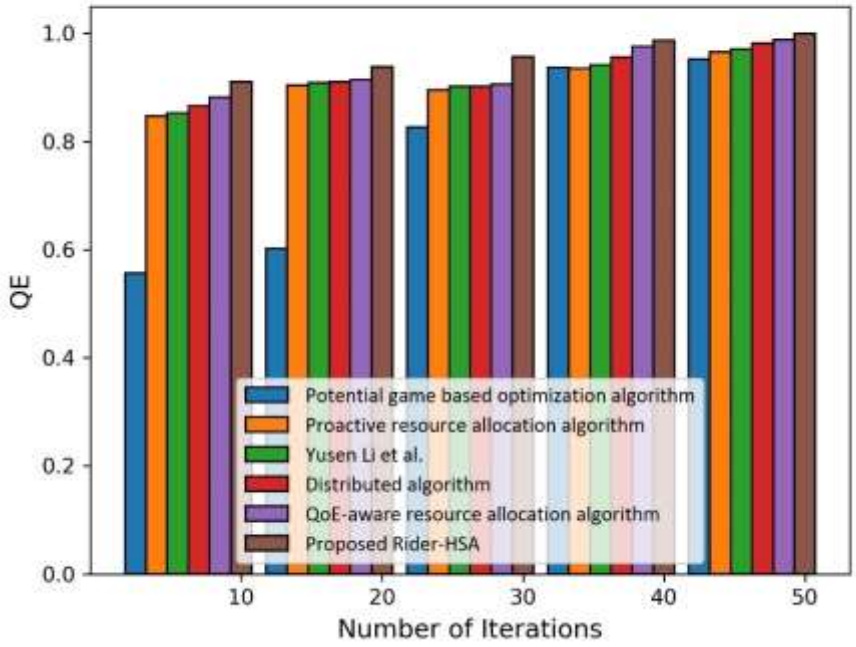

Figure 2. 21 Analyse comparative du RHSA sur la QE avec une taille de tâche 300

La figure 2.21 est l'évaluation des méthodes avec l'expérience QE à partir d'une taille de tâche égale à 300. Notre méthode *Rider based HSA* présentent les meilleures valeurs de QE.

La section ci-dessous est une évaluation de la convergence de notre méthode par rapport à d'autres algorithmes.

2.4.2.4 Analyse Comparative sur la convergence

2.4.2.4.1 Graphe de convergence avec une taille de tâche 100

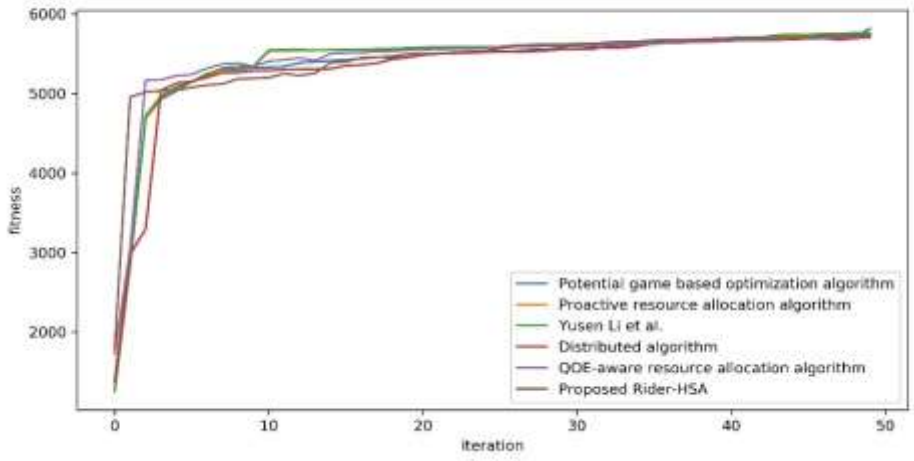

Figure 2. 22 Graphe de convergence avec une taille de tâche 100

La figure 2.22 est l'évaluation de la convergence de notre méthode avec d'autres méthodes pour une taille de tâche 100

Nous observons que la convergence de notre méthode en marron est moins bonne au bout de 10 itérations comparé aux autres méthodes.

Au bout de 20, 30, 40, 50 itérations, notre méthode converge de la même manière que les autres méthodes.

2.4.2.4.2 Graphe de convergence avec une taille de tâche 200

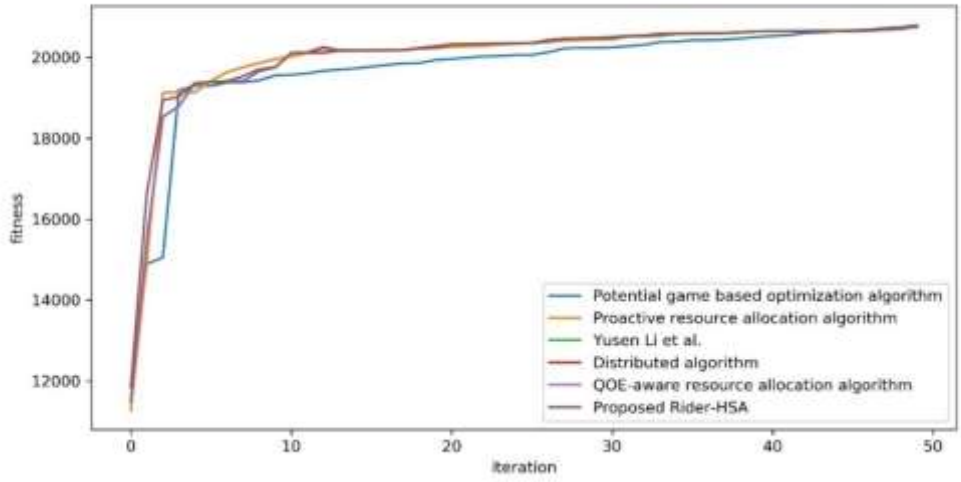

Figure 2. 23 Graphe de convergence avec une taille de tâche 200

La figure 2.23 est l'évaluation de la convergence de notre méthode avec d'autres méthodes pour une taille de tâche 200

Nous observons que la convergence de notre méthode en marron au bout de 10 itérations suit celle des autres méthodes.

2.4.2.4.3 Graphe de convergence avec une taille de tâche 300

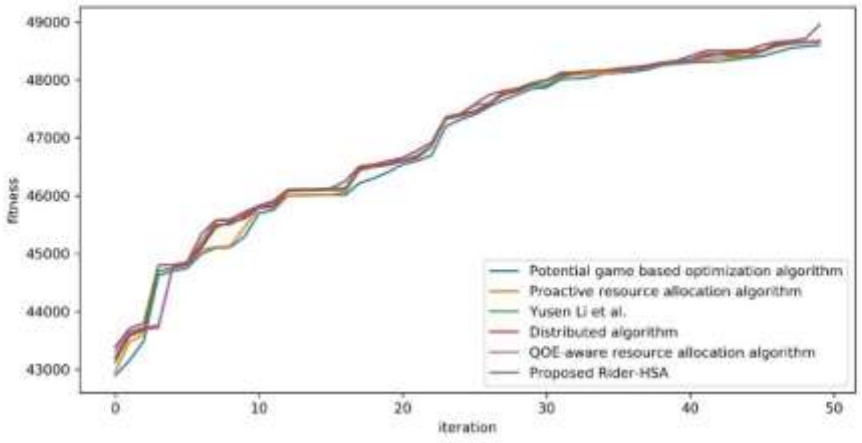

Figure 2. 24 Graphe de convergence avec une taille de tâche 300

La figure 2.24 est l'évaluation de la convergence de notre méthode avec d'autres méthodes pour une taille de tâche 300

Nous observons que la convergence de notre méthode en marron au bout de 10 itérations suit celle des certaines méthodes et offre une meilleure convergence comparée à l'algorithme *potential game based optimization algorithm.*

2.4.3 Discussions

Le tableau 2.2 ci-dessous illustre l'analyse des méthodes en fonction de l'équité, du MOS et de la QE en utilisant des tailles de tâche différentes 100, 200 et 300, respectivement. Si l'on considère une taille de tâche de 100, le RHSA améliore l'équité de 11,299 %, 11,29 %, 11,86 %, 11,864 %, 23,728 % par rapport à l'algorithme d'allocation de ressources basé sur la QoE *QoE-aware resource allocation algorithm*, l'algorithme distribué *Distributed algorithm*, *Yusen Li et al*., l'algorithme d'allocation de ressources proactif *Proactive resource allocation algorithm* ,l'algorithme d'optimisation basé sur le jeu potentiel *Potential game-based optimization algorithm* respectivement.

Le MOS connait une légère amélioration de 0,3% par rapport à l'algorithme d'allocation de ressources basé sur la QoE *QoE-aware resource allocation algorithm*, 0,4% pour l'algorithme distribué *Distributed algorithm*, *Yusen Li et al* et 3.37% l'algorithme d'optimisation basé sur le jeu potentiel *Potential game-based optimization algorithm*. Et, avec l'augmentation du nombre de requêtes de jeux, nous observons une légère progression

Grâce aux différentes analyses, il faut noter que notre algorithme RHSA offre une meilleure performance pour l'allocation des ressources *cloud*. Il offre une meilleure performance avec une taille de tâche réduite par rapport aux autres algorithmes.

Tableau 2. 2 Tableau comparatif 1

Game size	Metrics	Potential game based optimization algorithm	Proactive resource allocation algorithm	Yusen li *et al*	Distributed algorithm	QoE-aware resource allocation algorithm	Proposed Rider-based HSA (RHSA)
100	Fairness	0.135	0.156	0.156	0.157	0.157	**0.177**
	MOS	0.890	0.916	0.917	0.917	0.918	**0.921**
	QE	0.527	0.560	0.562	0.564	0.565	**0.578**
200	Fairness	0.401	0.537	0.540	0.541	0.546	**0.547**
	MOS	0.800	0.825	0.829	0.831	0.838	**0.848**
	QE	0.951	0.988	0.984	0.984	0.985	**0.999**
300	Fairness	0.538	0.978	0.980	0.984	0.990	**0.999**
	MOS	0.788	0.799	0.815	0.822	0.831	**0.873**
	QE	0.952	0.965	0.971	0.981	0.988	**1.000**

2.5 Conclusion

Une nouvelle heuristique a été proposée et détaillée dans ce chapitre. Ce nouvel algorithme hybride appelé *Rider-based HSA* (RHSA) est capable de trouver des solutions optimales pour résoudre le problème de l'allocation des ressources.

Notre proposition RHSA offre des résultats supérieurs par rapport à certaines méthodes d'optimisation de la littérature.

3 CHAPITRE 3 : Proposition d'une approche d'allocation de ressources multi-objectifs basée sur l'énergie

Sommaire

3.1 Introduction

Avec la croissance rapide du volume de données et la construction de centres de données, les problèmes d'efficacité énergétique et de gaspillage d'énergie retiennent de plus en plus l'attention. Cette augmentation de la consommation énergétique des technologies de l'information, a soulevé des inquiétudes [109][110][111]. En effet, l'augmentation de la consommation énergétique a entraîné une augmentation spectaculaire des émissions de CO2 et a affecté directement notre environnement. Par conséquent, la réduction de la consommation d'énergie est un problème important qui doit être résolu car elle réduira non seulement les coûts de l'énergie mais aussi la durabilité environnementale. Cependant, les systèmes *cloud* habituels souffrent d'un provisionnement en ressources inefficace qui entraîne un gaspillage d'énergie.

Une des solutions à ce problème est l'allocation optimale des ressources. Elle peut réduire les coûts d'exploitation et augmenter l'utilisation des ressources. L'objectif principal de l'allocation des ressources est d'optimiser la quantité de machines physiques actives dans l'environnement *cloud* et de créer des charges de travail de machines physiques en cours d'exécution qui sont uniformément réparties pour éviter la congestion. Un système de gestion durable et sensible à l'énergie doit offrir un compromis entre la consommation d'énergie et les performances du service[112].

Pour cette contribution, l'objectif principal est d'améliorer l'efficacité énergétique des systèmes des datacenters *cloud* en proposant une approche de provisionnement de ressources à plusieurs objectifs. Nous proposons un nouvel algorithme d'allocation des ressources basé sur l'architecture Spark pour identifier les solutions optimales pour résoudre les problèmes d'allocation des ressources. Notre algorithme d'optimisation, appelé *Fractional Rider-HSA (*FRHSA*)* est une modification de notre précédente contribution avec le concept du calcul fractionnaire.

3.2 Proposition de l'algorithme FRHSA

Le *cloud* présente plusieurs avantages, au nombre desquels une grande capacité de stockage, une grande capacité de calcul, une diversification des services. En raison du modèle du *cloud*, de plus en plus, les entreprises choisissent de migrer leurs applications ou leurs services vers le *cloud* afin de réduire les coûts. Ainsi, les datacenters hébergent d'énormes services et applications et il est impératif de trouver comment atteindre une allocation optimale des ressources dans les datacenters. Une technique d'allocation appropriée peut améliorer

l'utilisation des ressources et minimiser la consommation d'énergie. Cette technique permettra de réduire les coûts d'exploitation du fournisseur et celui du service au client. Nous proposons ci-dessous notre approche FRHSA basée sur l'architecture Spark.

3.2.1 Optimisation de l'allocation des ressources avec l'algorithme FRHSA

Cette section présente l'approche FRHSA *Fractional Rider-HSA* pour allouer les ressources et aider le fournisseur de services à réduire les coûts d'exploitation. Cette contribution vise à proposer une approche pour l'allocation des ressources distribuées dans le *cloud computing* en utilisant l'architecture spark. L'algorithme FRHSA proposé est la modification de l'algorithme RHSA avec la notion de calcul fractionnaire[113].

La figure 2 présente une illustration de notre proposition pour l'allocation des ressources.

Ici, J représente les ressources (resources units) et H les applications de jeux.

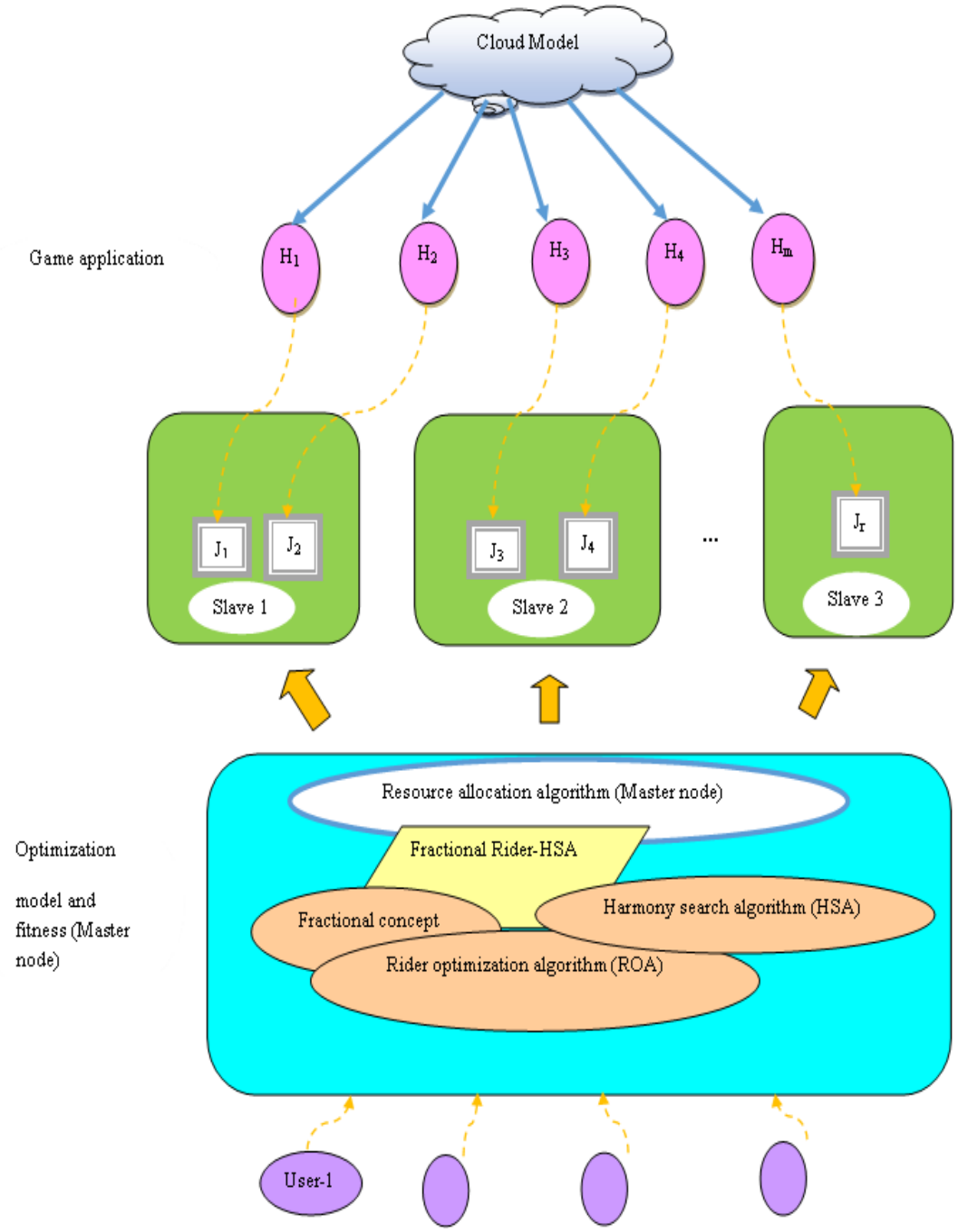

Figure 3. 1 Schéma fonctionnel du FRHSA pour l'allocation de ressources

3.2.2 Modèle multi-objectif

La fonction multi-objectif formulée pour le FRHSA est basée sur cinq paramètres. Nous avons ajouté deux paramètres à savoir l'énergie et un facteur de définition du réseau, à la fonction de fitness précédente. La nouvelle fonction est définie par

$$FO = (1 - QE) + MOS + J^T + N + (1 - E) \quad (3.1)$$

Où, le terme QE fait référence à la perte d'expérience du joueur, MOS le score d'opinion moyen, J^T l'équité, le terme N désigne un facteur de définition du réseau, et le terme E représente la consommation énergétique.

Le facteur de définition du réseau est exprimé par,

$$N = \frac{1}{2}[B + (1 - K)] \tag{3.2}$$

où, le terme B désigne la bande passante, et le terme K le délai.

La bande passante est définie par

$$B = \frac{1}{x \times y} \sum_{d=1}^{x} \sum_{l=1}^{y} A_{dl} \tag{3.3}$$

A_{dl} est la bande passante de l^{th} VM dans le d^{th}PM, x représente le nombre total de PM, et y représente le nombre total de VM. Ensuite, le délai est calculé à l'aide de l'expression ci-dessous,

$$K = \kappa_1 * U \tag{3.4}$$

où, le terme κ_1 désigne les paramètres constants.

L'équation de l'énergie totale [5]consommée est donnée par,

$$E = \sum_{i=1}^{r} E_{Total}\,(i) + \sum_{j=1}^{k} E_{static}\,(j) \tag{3.5}$$

L'énergie totale correspondant à l'exécution de l'application comprend la perte d'énergie sur le terminal et le serveur et est représentée comme,

$$E_{Total}(i) = E_{local,(i)} + E^{i}_{server,(j)} \tag{3.6}$$

où, le terme j se réfère à l'index du serveur où l'application est mappée. Considérons que les serveurs suivent la politique de gestion de l'énergie par temps mort, ce qui signifie qu'après une certaine période d'inactivité, le mode de faible consommation est maintenu. Supposons T que soit le seuil de sortie, et $P_{static,j}$ représente la puissance statique du serveur pendant la période j d'inactivité. Par conséquent, la consommation d'énergie statique du serveur lorsqu'il est inactif est exprimée par,

$$E_{static}(j) = P_{static,j} * T \tag{3.7}$$

On obtient la perte d'expérience de jeu [65] du joueur, qui est l'objectif de chaque joueur par

$$QE = \mu_1 U - \mu_2 W^l - \mu_2 N_{Qual} \tag{3.8}$$

où, μ_1, μ_2 et μ_3 indique des paramètres constants, U symbolise le délai, W^l le nombre d'images par seconde (FPS), et N_{Qual} la qualité vidéo du jeu.

La perte d'expérience de jeu QE doit être minimale alors l'expérience pour les joueurs est maximale

Les autres paramètres de la fonction sont définis dans le chapitre 2 au niveau de la section 2.2.2.2

3.2.3 Description de l'algorithme FRHSA

Le FRHSA est une combinaison du calcul fractionnaire FC [113], du ROA [102] et du HSA [103]. Le calcul fractionnaire est utilisé pour résoudre les équations intégrales et les dérivées. Ici, les équations intégrales et différentielles d'ordre fractionnaire sont résolues par des transformées de Laplace. Les étapes du calcul fractionnaire sont les suivantes : l'étape primaire trouve la transformée de Laplace de l'équation. L'étape secondaire est utilisée pour résoudre la transformée de la fonction inconnue, et l'étape finale utilise la transformée de Laplace inverse pour identifier la meilleure solution. Le FC est utilisé pour améliorer les performances de calcul du RHSA (Rider-HSA).

La combinaison du concept FC, du ROA et du HSA permet d'ajuster les paramètres pour obtenir une solution globalement optimale. Les étapes du FRHSA sont les suivantes :

Étape 1 : Initialisation

L'initialisation de l'algorithme proposé est effectuée sur la base de quatre coureurs, tels que le suiveur, l'attaquant, le coureur de contournement et le dépasseur. L'initialisation de l'emplacement est effectuée de manière arbitraire. L'initialisation du groupe est donnée par,

$$Y_{\hbar} = \{Y_{\hbar}(c,n)\}; 1 \le c \le M, 1 \le n \le N \tag{3.9}$$

où, M se réfère au nombre total des coureurs, et $\mathrm{Y}_{\hbar}(c,n)$ signifie l'emplacement du c^{th} coureur dans la n^{th} dimension à l'instant $\hbar^{th}$, et N représente la dimension totale.

Étape 2 : Identification de la fonction de fitness

Le calcul de la fonction fitness est nécessaire pour rechercher la meilleure solution. Ainsi, la fonction fitness de chaque solution est définie dans la section 4.2.1 pour atteindre la solution optimale. La solution optimale est identifiée à l'itération finale car chaque solution cherche à atteindre le meilleur emplacement.

Étape 3 : Détermination des mises à jour des positions

Dans cette étape, les solutions optimales sont déterminées sur la base du FRHSA et l'équation actualisée de l'algorithme RHSA est exprimée par,

$$Y_{\hbar+1}(c,n) = \frac{\lambda[(Y_{\hbar}(\eta,n) * [1-\gamma(n)] \pm rand(0,1).P) * \gamma(n)]}{1-\lambda\gamma(n)} \quad (3.10)$$

$$Y_{\hbar+1}(c,n) - Y_{\hbar}(c,n) = \frac{\lambda[(Y_{\hbar}(\eta,n) * [1-\gamma(n)] \pm rand(0,1).P) * \gamma(n)]}{1-\lambda\gamma(n)} - Y_{\hbar}(c,n) \quad (3.11)$$

$$d^{\varepsilon}[Y_{\hbar+1}(c,n) - Y_{\hbar}(c,n)] = \frac{\lambda[(Y_{\hbar}(\eta,n) * [1-\gamma(n)] \pm rand(0,1).P) * \gamma(n)]}{1-\lambda\gamma(n)} - Y_{\hbar}(c,n) \quad (3.12)$$

Le modèle FC est utilisé pour améliorer les performances du système et résoudre les problèmes d'optimisation du RHSA.

$$Y_{\hbar+1}(c,n) - \varepsilon Y_{\hbar}(c,n) - \frac{1}{2}\varepsilon Y_{\hbar-1}(c,n) - \frac{1}{6}(1-\varepsilon)Y_{\hbar-2}(c,n) - \frac{1}{24}\varepsilon(1-\varepsilon)(2-\varepsilon)Y_{\hbar-3}(c,n) = \frac{\lambda[(Y_{\hbar}(\eta,n) * [1-\gamma(n)] \pm rand(0,1).P) * \gamma(n)]}{1-\lambda\gamma(n)} - Y_{\hbar}(c,n) \quad (3.13)$$

$$Y_{\hbar+1}(c,n) = \varepsilon Y_{\hbar}(c,n) - Y_{\hbar}(c,n) + \frac{1}{2}\varepsilon Y_{\hbar-1}(c,n) + \frac{1}{6}(1-\varepsilon)Y_{\hbar-2}(c,n) + \frac{1}{24}\varepsilon(1-\varepsilon) \quad (2-\varepsilon)Y_{\hbar-3}(c,n) + \frac{\lambda[(Y_{\hbar}(\eta,n) * [1-\gamma(n)] \pm rand(0,1).P) * \gamma(n)]}{1-\lambda\gamma(n)} \quad (3.14)$$

L'équation finale actualisée du FRHSA est donnée par,

$$Y_{\hbar+1}(c,n) = (\varepsilon-1)Y_{\hbar}(c,n) + \frac{1}{2}\varepsilon Y_{\hbar-1}(c,n) + \frac{1}{6}(1-\varepsilon)Y_{\hbar-2}(c,n) + \frac{1}{24}\varepsilon(1-\varepsilon)(2-\varepsilon)Y_{\hbar-3}(c,n) + \frac{\lambda[(Y_{\hbar}(\eta,n) * [1-\gamma(n)] \pm rand(0,1).P) * \gamma(n)]}{1-\lambda\gamma(n)} \quad (3.15)$$

où, $rand(0,1)$ représente un nombre aléatoire compris entre [0, 1], P signifie une largeur de bande de distance arbitraire.

Étape 4 : évaluer la faisabilité

Sur la base de la valeur de fitness, la faisabilité est à nouveau calculée. Ainsi, si la nouvelle solution est meilleure que la précédente, l'ancienne est remplacée par la nouvelle solution.

Étape 5 : Terminaison

Les étapes ci-dessus sont répétées jusqu'à ce que la meilleure solution soit obtenue pour l'allocation des ressources.

Le pseudo-code de l'approche FRHSA est représenté dans le tableau 3.1

Tableau 3. 1 Pseudo-code de l'algorithme FRHSA

Input:: $\mathrm{Y}_{\hbar}$ *Random position of the Rider,* h *: iteration,* $h_{\max}$ *: maximum iteration*

Output*: Leading rider* Y^{s}

Begin

Initialize the set of solutions.

Initialize other parameters of a rider like gear, brake, accelerator, and steering angle

Determine fitness function using equation (4.1)

While $n < N_{OFF}$

For $u = 1\ to\ N$

Update the Rider-HSA with equation (4.9)

Update the Fractional Rider-HSA with equation (4.14)

Rank riders using fitness function with equation (4.1)

Choose the Rider with a high fitness function

Update gear, brake, accelerator, and steering angle

Return Y^{S}

$n = n + 1$

End for

End while

End

3.3 Résultats et discussions

Cette section traite de l'analyse de performance de notre approche avec différentes tailles de population en changeant le nombre d'itérations. Les tailles des tâches sont 200 et 300.

Les mesures de performance utilisées sont les suivantes : MOS, QE, *Fairness*, consommation énergétique et Délai.

Par la suite, notre approche a été comparée à notre précédente approche et à d'autres approches de la littérature.

Nos différentes expériences ont été exécutées en utilisant le simulateur *cloudSim* [108] sous PYTHON avec un PC contenant 12 GB de RAM, Windows 10 OS, ROM-plus de 100GB, CPU core i7 2.2 GHz.

Le nombre de machines physiques utilisées est égal à 6, le nombre total de machines virtuelles est équivalent au nombre de tâches.

3.3.1 Mesures de performance du FRHSA

3.3.1.1 Evaluation du FRHSA avec une taille de tâche 200

3.3.1.1.1 Evaluation du FRHSA sur le délai avec une taille de tâche 200

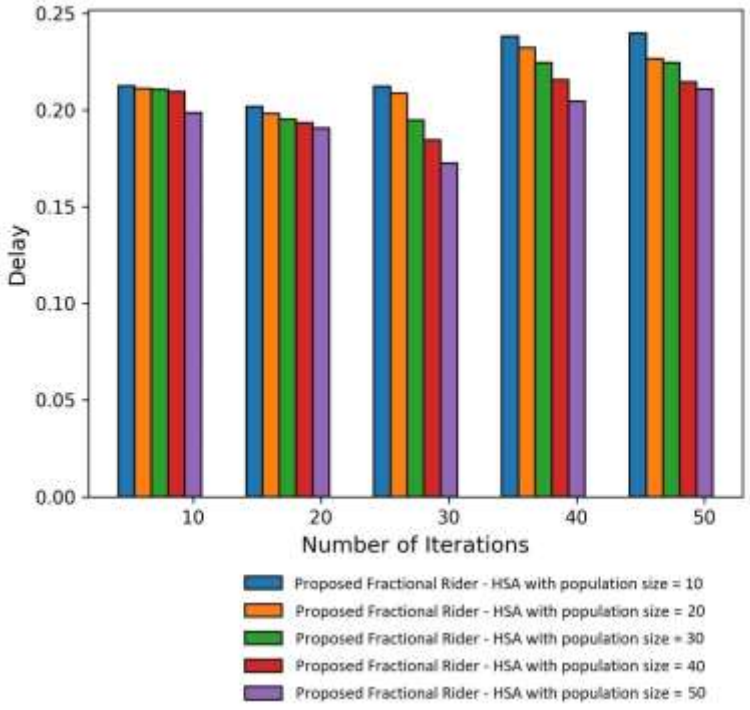

Figure 3. 2 Evaluation du FRHSA sur le délai avec une taille de tâche 200

La figure 3.2 ci-dessus présente l'évaluation de notre algorithme FRHSA sur le délai de traitement à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, le délai évalué par le RHSA avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.2125, 0.2111, 0.2105, 0.2094, et 0.1985. Ces résultats montrent que plus la taille de la population augmente, la valeur du délai baisse.

Nos résultats, de l'impact de notre algorithme FRHSA sur le délai, montrent qu'avec un nombre élevé d'itérations, nous améliorons la valeur de délai.

3.3.1.1.2 Evaluation du FRHSA sur l'Energie avec une taille de tâche 200

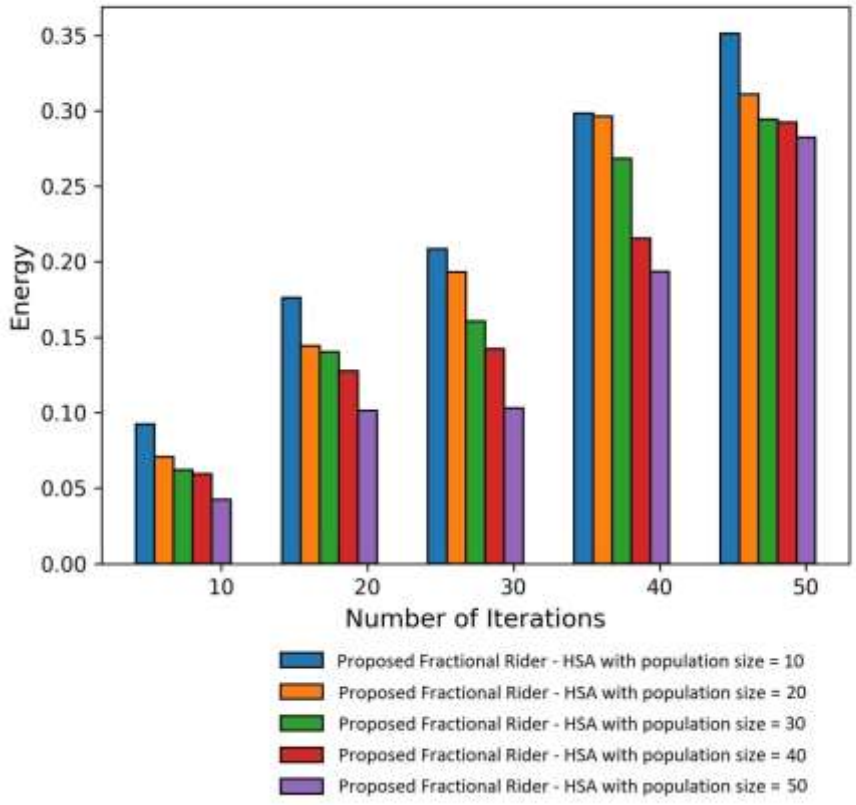

Figure 3. 3 Evaluation du FRHSA sur l'Energie avec une taille de tâche 200

La figure 3.3 présente l'évaluation de notre algorithme FRHSA sur la consommation énergétique à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.

Nos résultats, de l'impact de notre algorithme FRHSA sur la consommation énergétique, montrent qu'avec un nombre élevé d'itérations, nous améliorons la consommation énergétique.

3.3.1.1.3 Evaluation du FRHSA sur l'Equité avec une taille de tâche 200

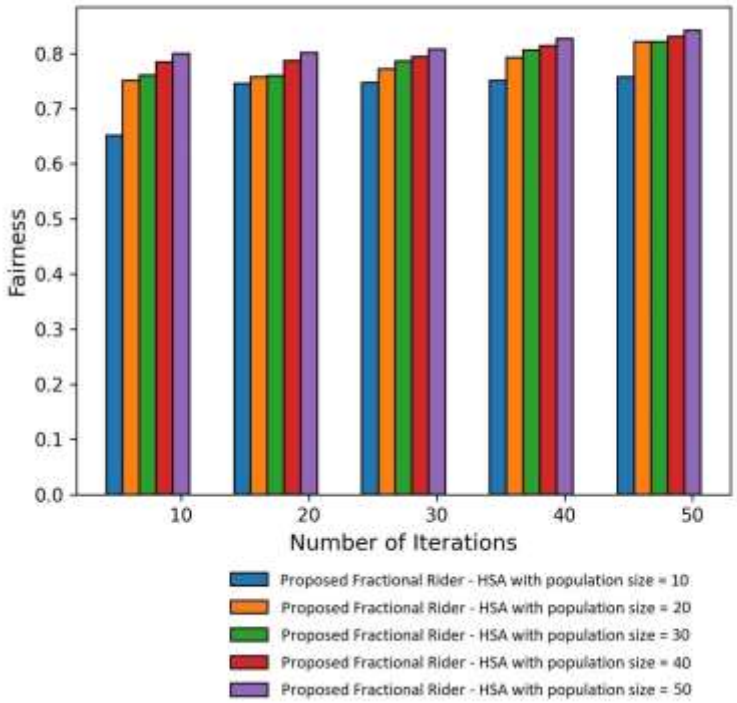

Figure 3. 4 Evaluation du FRHSA sur l'Equité avec une taille de tâche 200

La figure 3.4 présente l'évaluation de notre algorithme FRHSA sur l'équité à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.

Nos résultats, de l'impact de notre algorithme FRHSA sur l'équité, montrent qu'avec un nombre élevé d'itérations, nous améliorons la valeur de l'équité.

3.3.1.1.4 Evaluation du FRHSA sur le MOS avec une taille de tâche 200

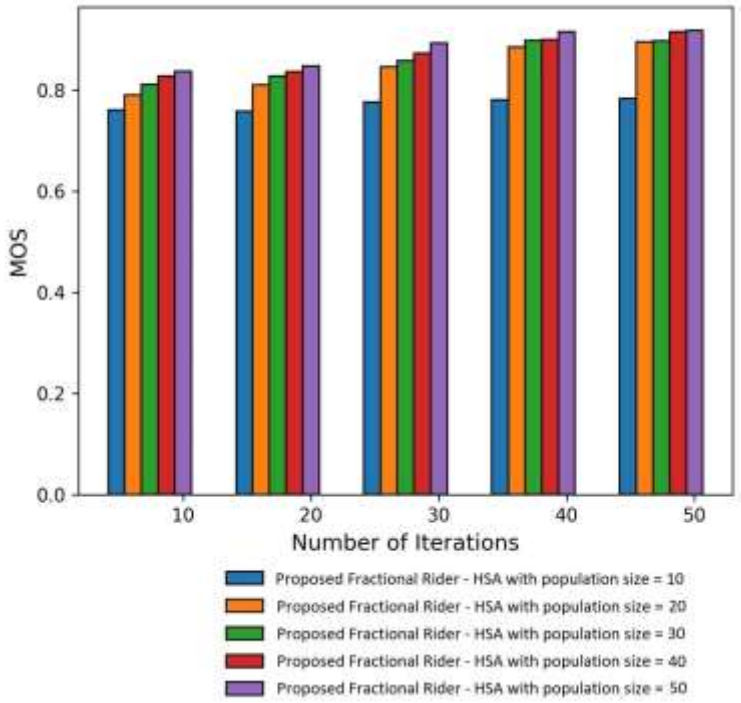

Figure 3. 5 Evaluation du FRHSA sur le MOS avec une taille de tâche 200

La figure 3.5 présente l'évaluation de notre algorithme FRHSA sur le MOS à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.

Nos résultats, de l'impact de notre algorithme FRHSA sur la MOS, montrent qu'avec un nombre élevé d'itérations, nous améliorons la valeur du MOS.

3.3.1.1.5 Evaluation du FRHSA sur la QE avec une taille de tâche 200

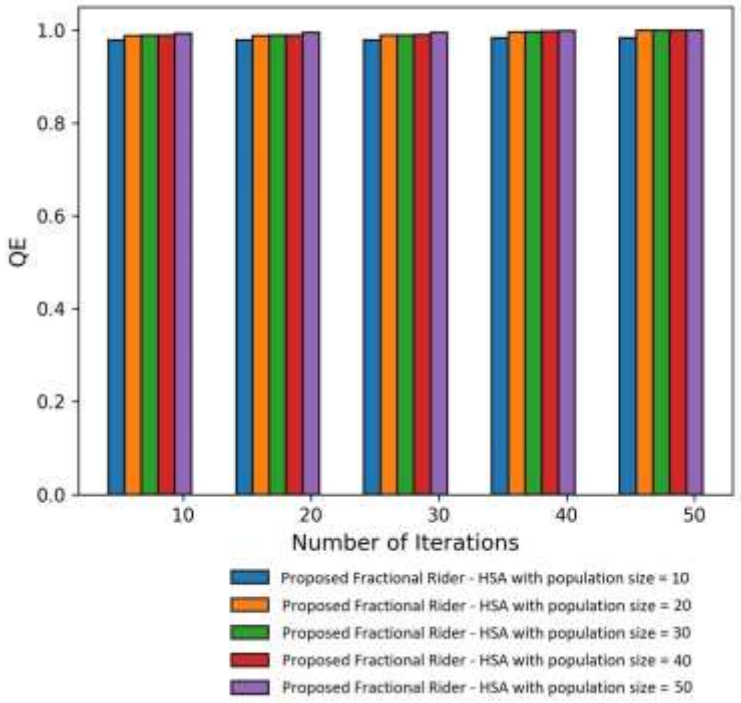

Figure 3. 6 Evaluation du FRHSA sur la QE avec une taille de tâche 200

La figure 3.6 présente l'évaluation de notre algorithme FRHSA sur l'expérience QE à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50. Nos résultats, de l'impact de notre algorithme FRHSA sur QE, montrent qu'avec un nombre élevé d'itérations 50, nous améliorons la valeur du QE.

Nous poursuivons l'analyse comparative de la méthode FRHSA avec une augmentation de la taille des tâches.

3.3.1.2 Evaluation du FRHSA avec une taille de tâche 300

3.3.1.2.1 Evaluation du FRHSA sur le délai avec une taille de tâche 300

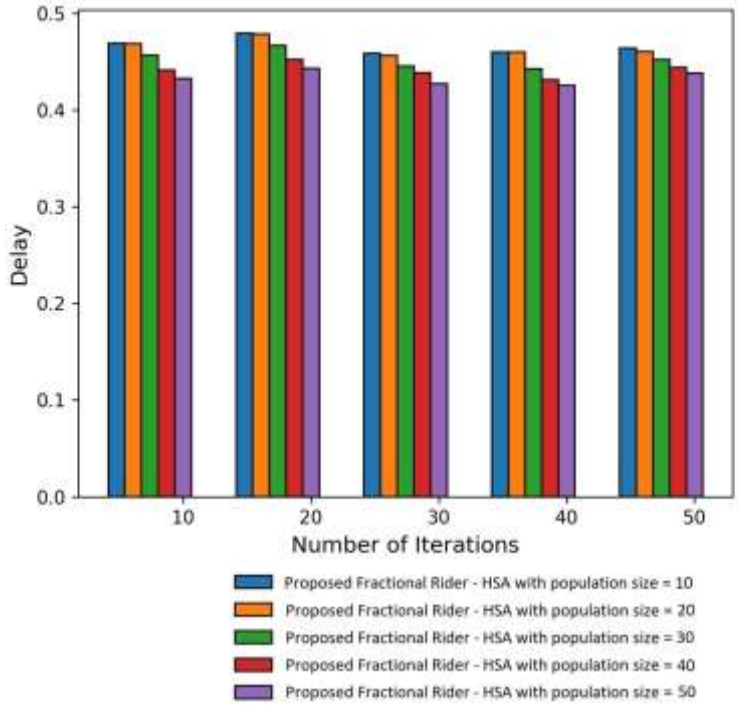

Figure 3. 7 Evaluation du FRHSA sur le délai avec une taille de tâche 300

La figure 3.7 montre l'évaluation de notre algorithme FRHSA sur l'expérience le délai à partir d'une taille de tâche égale à 300. Nous avons fait varier la taille de la population de 10 à 50. Nos résultats, de l'impact de notre algorithme FRHSA sur le délai, montrent qu'avec un nombre élevé d'itérations 50, nous améliorons la valeur du délai.

3.3.1.2.2 Évaluation du FRHSA sur l'Energie avec une taille de tâche 300

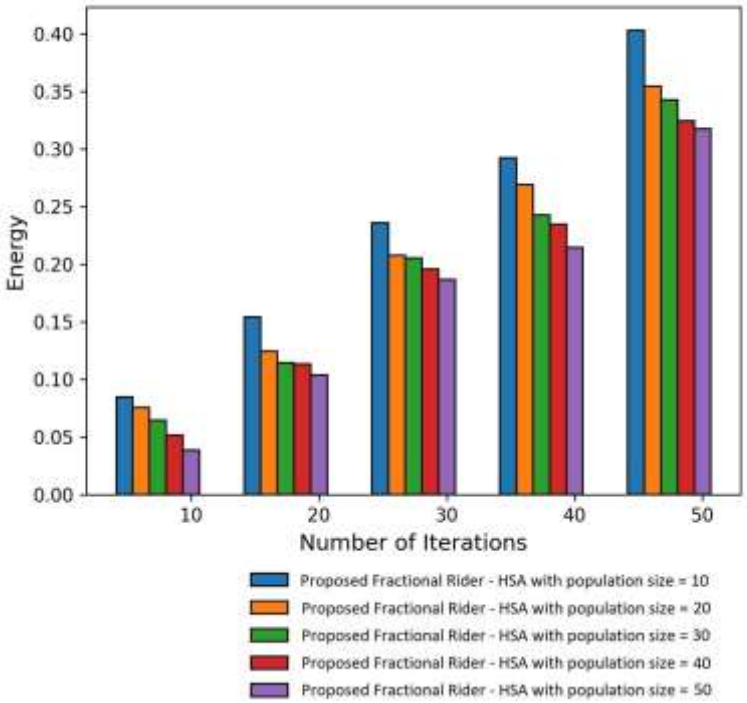

Figure 3. 8 Évaluation du FRHSA sur l'Energie avec une taille de tâche 300

La figure 3.8 montre l'évaluation de notre algorithme FRHSA sur la consommation énergétique à partir d'une taille de tâche égale à 300. Nous avons fait varier la taille de la population de 10 à 50.

Nos résultats, de l'impact de notre algorithme FRHSA sur la consommation énergétique, montrent qu'avec un nombre élevé d'itérations 50, nous arrivons à réduire la consommation énergétique au sein des DCs.

3.3.1.2.3 Evaluation du FRHSA sur l'équité avec une taille de tâche 300

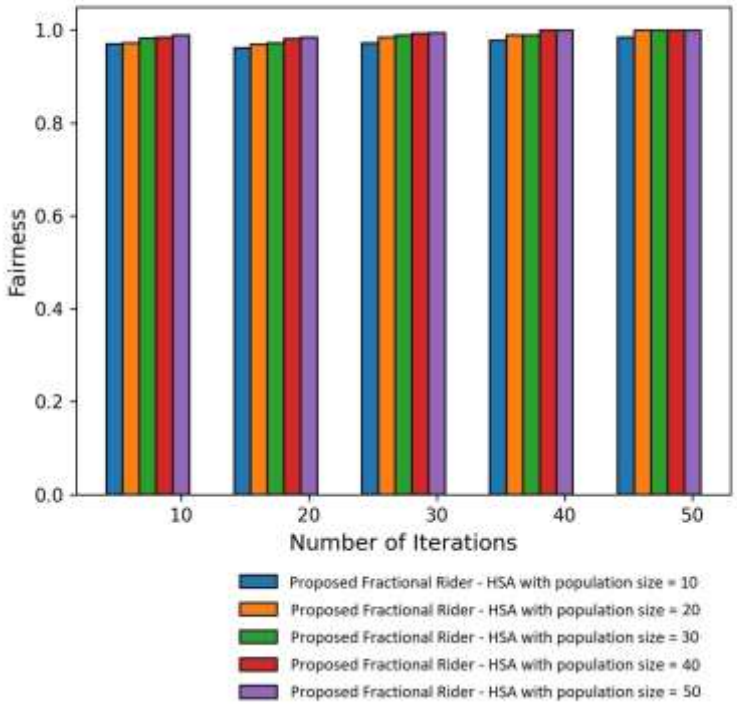

Figure 3. 9 Evaluation du FRHSA sur l'équité avec une taille de tâche 300

La figure 3.9 montre l'évaluation du FRHSA sur l'équité à partir d'une taille de tâche égale à 300. Nos résultats, de l'impact de notre algorithme FRHSA sur l'équité, montrent que lorsque le nombre d'itérations augmente, nous améliorons la valeur de l'équité.

3.3.1.2.4 Evaluation du FRHSA sur le MOS avec une taille de tâche 300

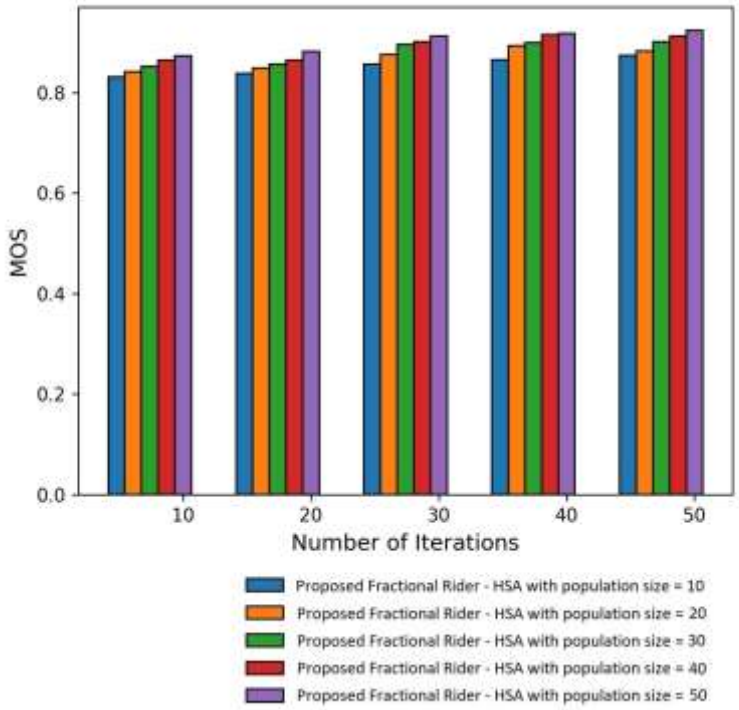

Figure 3. 10 Evaluation du FRHSA sur le MOS avec une taille de tâche 300

La figure 3.10 montre l'évaluation du FRHSA sur le MOS à partir d'une taille de tâche égale à 300. Nos résultats, de l'impact de notre algorithme FRHSA sur le MOS , montrent que lorsque le nombre d'itérations augmente, nous améliorons la valeur du MOS.

3.3.1.2.5 Evaluation du FRHSA sur le QE avec une taille de tâche 300

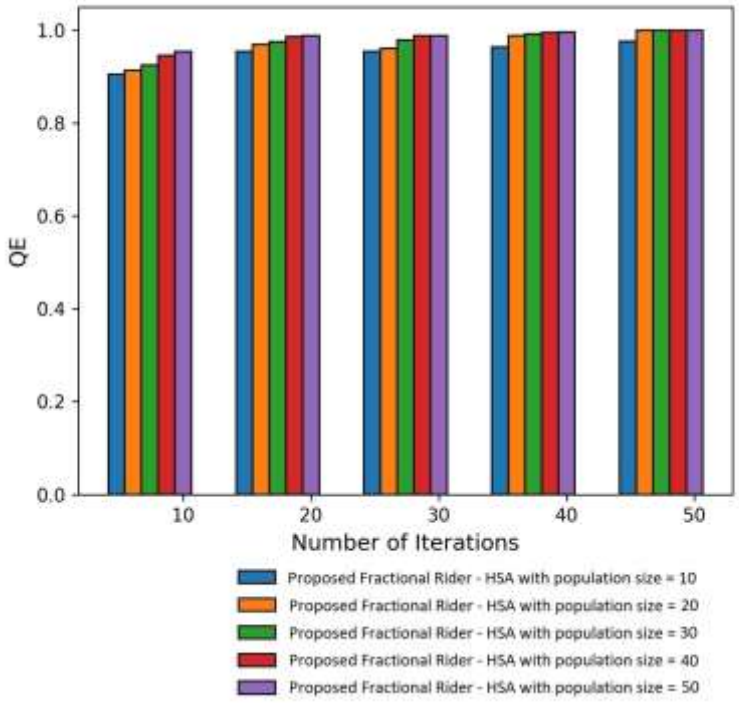

Figure 3. 11 Evaluation du FRHSA sur le QE avec une taille de tâche 300

La figure 3.11 montre l'évaluation du FRHSA sur l'expérience QE à partir d'une taille de tâche égale à 300. Nos résultats montrent que lorsque le nombre d'itérations augmente, nous améliorons la valeur de QE.

Nos résultats montrent que lorsque la taille de la population et le nombre d'itérations augmentent, les différentes métriques du modèle s'améliorent.
Le nouvel algorithme FRHSA offre de meilleures performances que notre précédente méthode.
Le FRHSA converge plus rapidement que le RHSA.

3.3.2 Analyse comparative

La méthode FRHSA est analysée avec d'autres méthodes, telles que l'algorithme d'allocation des ressources en fonction de la QoE [67], l'algorithme d'allocation proactive des ressources[66], l'algorithme d'optimisation basé sur le jeu potentiel [65] et notre première contribution [114].
Cette analyse est effectuée en utilisant des paramètres tels que MOS, QE, équité, l'énergie consommée et le délai.

3.3.2.1 Analyse comparative avec une taille de tâche 200

3.3.2.1.1 Analyse comparative du FRHSA en fonction du délai avec une taille de tâche 200

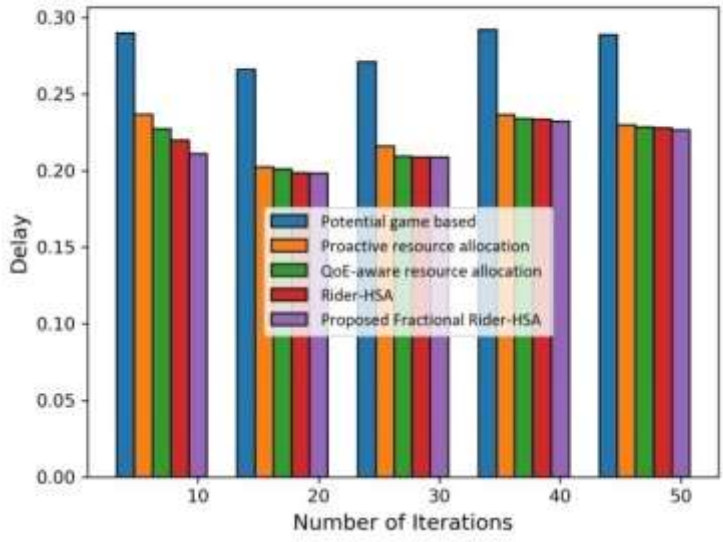

Figure 3. 12 Analyse comparative du FRHSA en fonction du délai avec une taille de tâche 200

La figure 3.12 présente l'évaluation des méthodes avec le délai à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50.

Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet de réduire à chaque fois le délai de traitement avec les différentes variations du nombre d'itération.

3.3.2.1.2 Analyse comparative du FRHSA en fonction de l'énergie avec une taille de tâche 200

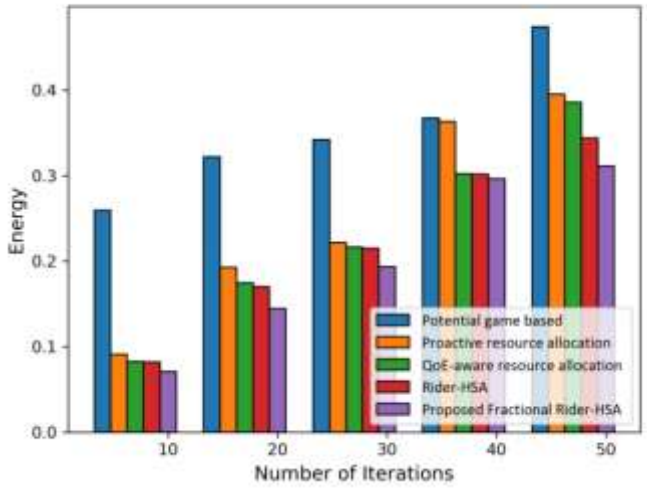

Figure 3. 13 Analyse comparative du FRHSA en fonction de l'énergie avec une taille de tâche 200

La figure 3.13 présente la comparaison de notre méthode avec d'autres méthodes sur le critère de la consommation énergétique à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50.

Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet de réduire à chaque fois la consommation énergétique.

3.3.2.1.3 Analyse comparative du FRHSA en fonction de l'équité avec une taille de tâche 200

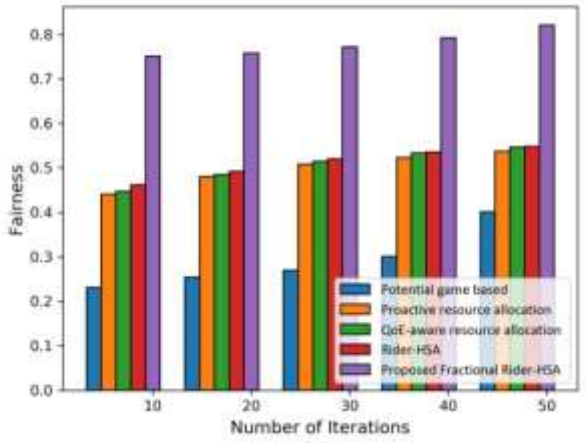

Figure 3. 14 Analyse comparative du FRHSA en fonction de l'équité avec une taille de tâche 200

La figure 3.14 présente la comparaison de notre méthode avec d'autres méthodes sur le critère équité à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet d'améliorer à chaque itération la valeur de l'équité.

3.3.2.1.4 Analyse comparative du FRHSA en fonction de QE avec une taille de tâche 200

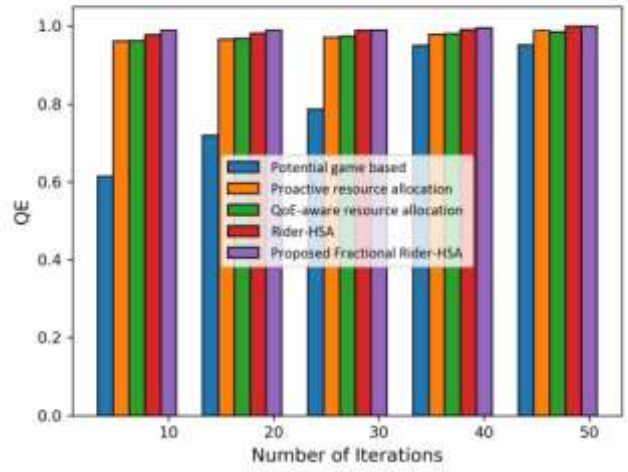

Figure 3. 15 Analyse comparative du FRHSA en fonction de QE avec une taille de tâche 200

La figure 3.15 présente la comparaison de notre méthode FRHSA avec d'autres méthodes sur le critère expérience du joueur QE à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50.

Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet pour chaque itération d'augmenter la valeur de l'expérience.

3.3.2.1.5 Analyse comparative du FRHSA en fonction de MOS avec une taille de tâche 200

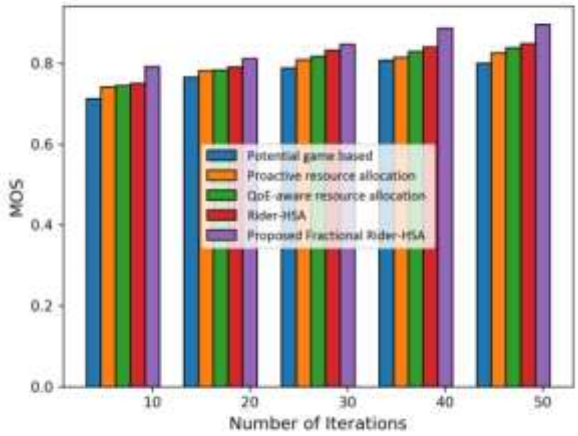

Figure 3. 16 Analyse comparative du FRHSA en fonction de MOS avec une taille de tâche 200

La figure 3.16 présente la comparaison de notre méthode FRHSA avec d'autres méthodes sur le MOS à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet pour chaque itération d'augmenter la valeur du MOS.

Nous poursuivons dans la section ci-dessous l'évaluation de performance de notre algorithme en augmentant la taille des tâches de 100.

3.3.2.2 Analyse comparative avec une taille de tâche 300

3.3.2.2.1 Analyse comparative du FRHSA en fonction du délai avec une taille de tâche 300

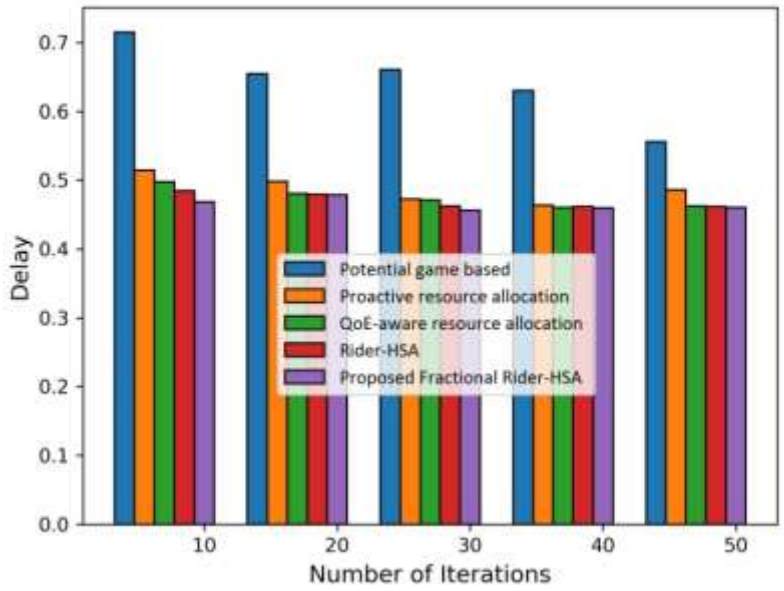

Figure 3. 17 Analyse comparative du FRHSA en fonction du délai avec une taille de tâche 300

La figure 3.17 est la comparaison de notre méthode FRHSA avec d'autres méthodes sur le délai à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet de réduire pour chaque itération la valeur du délai de traitement.

3.3.2.2.2 Analyse comparative du FRHSA en fonction de l'énergie avec une taille de tâche 300

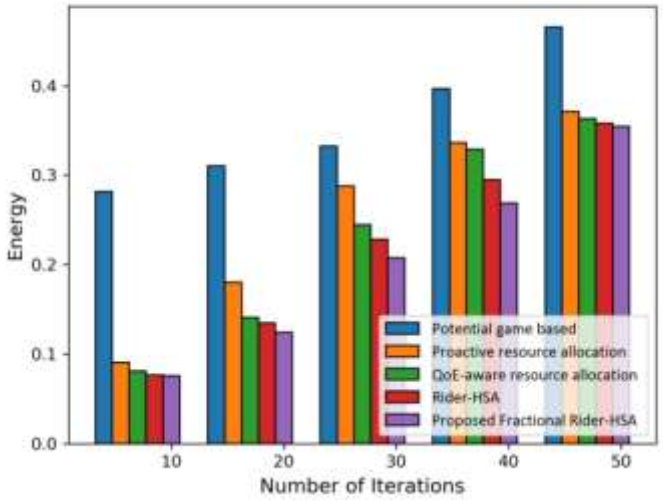

Figure 3. 18 Analyse comparative du FRHSA en fonction de l'énergie avec une taille de tâche 300

La figure 3.18 est la comparaison de notre méthode FRHSA avec d'autres méthodes sur l'énergie à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet pour chaque itération de réduire la consommation la consommation énergétique.

3.3.2.2.3 Analyse comparative du FRHSA en fonction de l'équité avec une taille de tâche 300

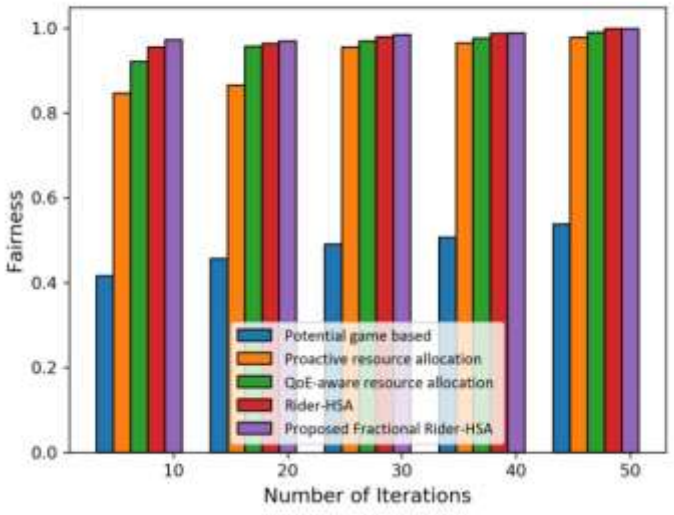

Figure 3. 19 Analyse comparative du FRHSA en fonction de l'équité avec une taille de tâche 300

La figure 3.19 est la comparaison de notre méthode FRHSA avec d'autres méthodes sur l'équité à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet à chaque itération d'améliorer la valeur de l'équité.

3.3.2.2.4 Analyse comparative du FRHSA en fonction de MOS avec une taille de tâche 300

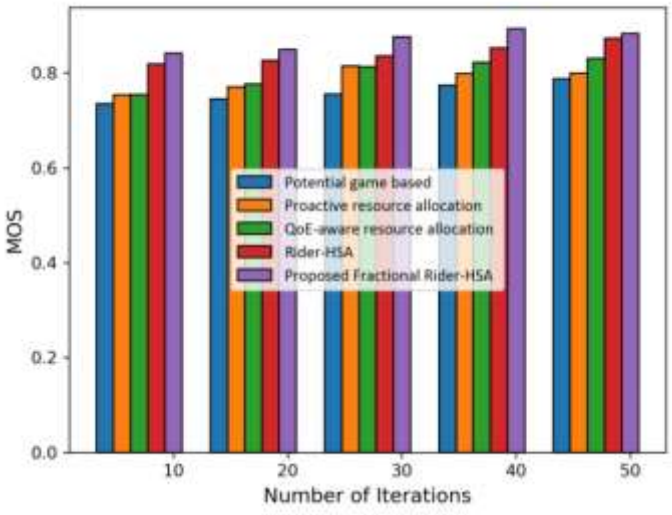

Figure 3. 20 Analyse comparative du FRHSA en fonction de MOS avec une taille de tâche 300

La figure 3.20 est la comparaison de notre méthode FRHSA avec d'autres méthodes sur le MOS à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet à chacune des itérations d'améliorer la valeur de MOS.

3.3.2.2.5 Analyse comparative du FRHSA en fonction de QE avec une taille de tâche 300

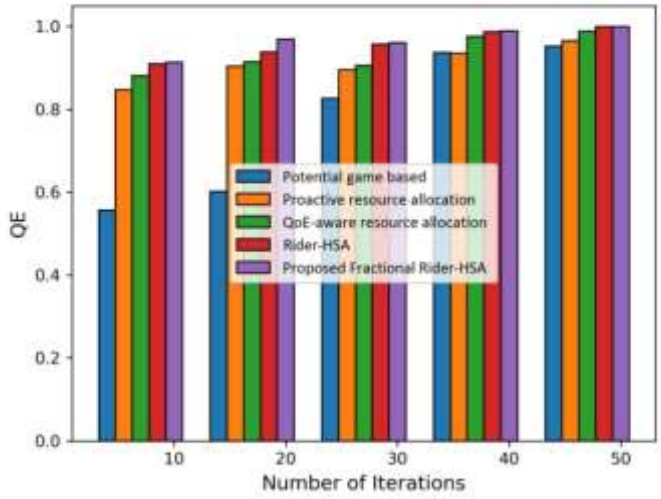

Figure 3. 21 Analyse comparative du FRHSA en fonction de QE avec une taille de tâche 300

La figure 3.21 est la comparaison de notre méthode FRHSA avec d'autres méthodes sur l'expérience à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50.

Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet à chacune des itérations d'améliorer l'expérience du joueur.

Nous poursuivons notre évaluation comparative en augmentant encore la taille de tâches à 500.

3.3.2.3 Analyse comparative avec une taille de tâche 500

3.3.2.3.1 Analyse comparative du FRHSA en fonction du délai avec une taille de tâche 500

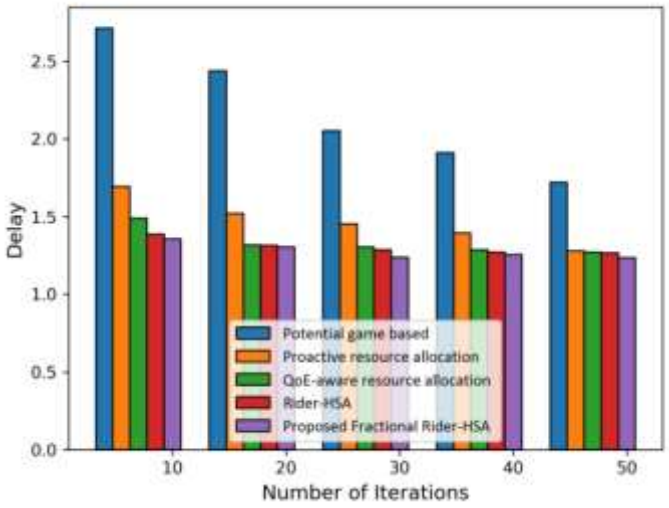

Figure 3. 22 Analyse comparative du FRHSA en fonction du délai avec une taille de tâche 500

La figure 3.22 est la comparaison de notre méthode FRHSA avec d'autres méthodes sur le délai à partir d'une taille de tâche égale à 500. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Notre contribution FRHSA *Fractional Rider based HSA* permet à chacune des itérations de réduire le délai de traitement.

3.3.2.3.2 Analyse comparative du FRHSA en fonction de l'énergie avec une taille de tâche 500

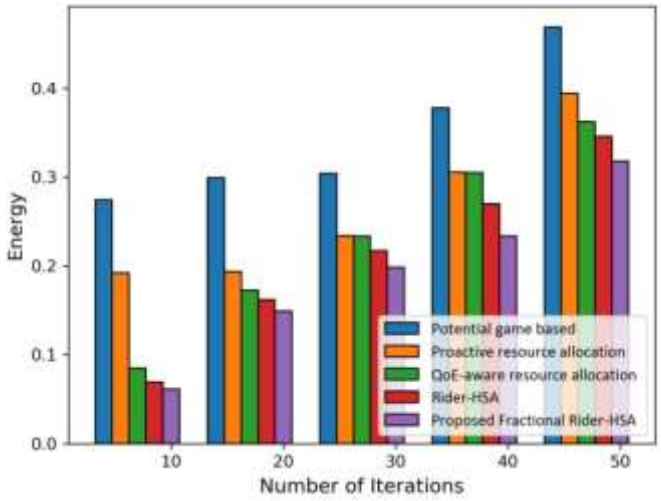

Figure 3. 23 Analyse comparative du FRHSA en fonction de l'énergie avec une taille de tâche 500

La figure 3.23 est la comparaison de notre méthode FRHSA avec d'autres méthodes sur le délai à partir d'une taille de tâche égale à 500. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Le FRHSA *Fractional Rider based HSA* permet à chacune des itérations de réduire la valeur de l'énergie.

3.3.2.3.3 Analyse comparative du FRHSA en fonction de l'équité avec une taille de tâche 500

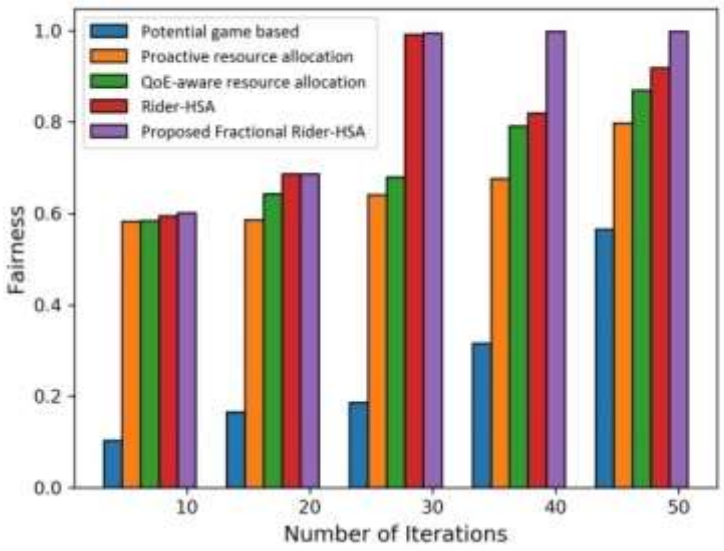

Figure 3. 24 Analyse comparative du FRHSA en fonction de l'équité avec une taille de tâche 500

La figure 3.24 est la comparaison de FRHSA avec d'autres méthodes sur l'équité à partir d'une taille de tâche égale à 500. Nous fixons la taille de la population à 50.
Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Le FRHSA *Fractional Rider based HSA* permet à chacune des itérations d'augmenter la valeur de l'équité.

3.3.2.3.4 Analyse comparative du FRHSA en fonction de MOS avec une taille de tâche 500

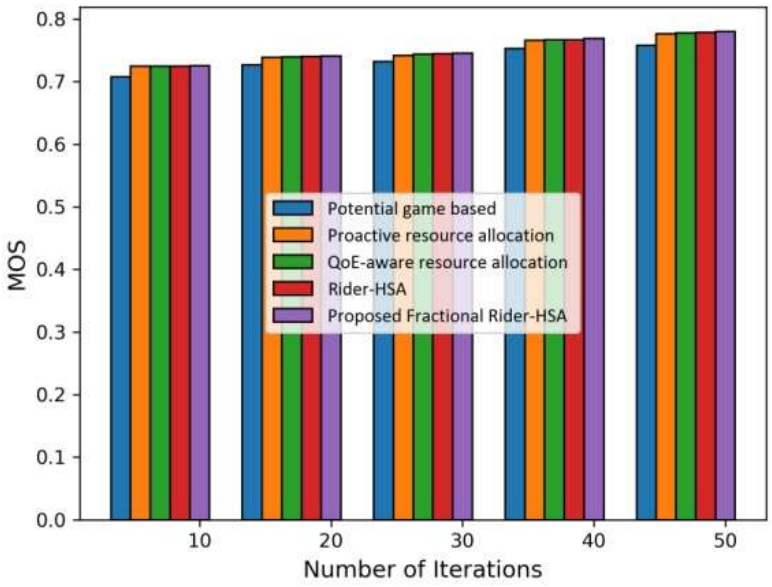

Figure 3. 25 Analyse comparative du FRHSA en fonction de MOS avec une taille de tâche 500

La figure 3.25 est la comparaison de FRHSA avec d'autres méthodes sur le MOS à partir d'une taille de tâche égale à 500. Nous fixons la taille de la population à 50.
Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Le FRHSA *Fractional Rider based HSA* permet à chacune des itérations d'augmenter la valeur de MOS.

3.3.2.3.5 Analyse comparative du FRHSA en fonction de QE avec une taille de tâche 500

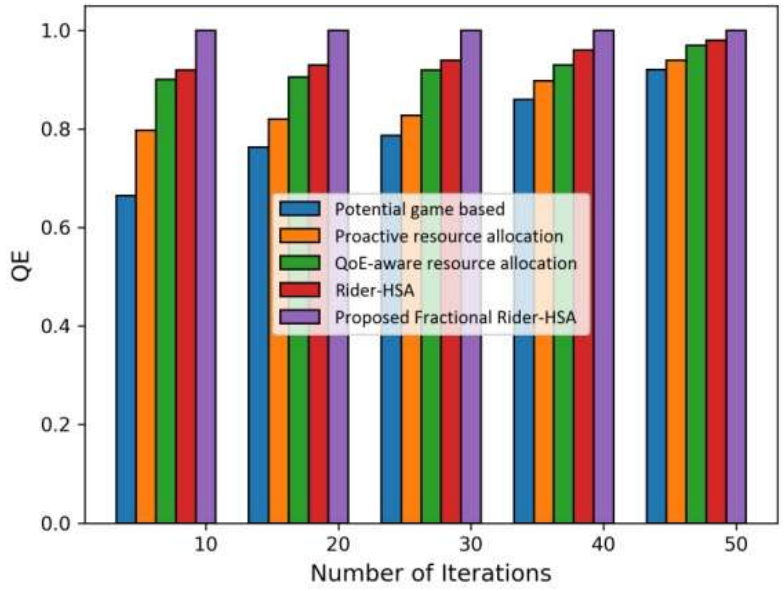

Figure 3. 26 Analyse comparative du FRHSA en fonction de QE avec une taille de tâche 500

La figure 3.26 est la comparaison de FRHSA avec d'autres méthodes sur l'expérience à partir d'une taille de tâche égale à 500. Nous fixons la taille de la population à 50.
Nous comparons notre contribution de couleur violet avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre précédente méthode RHSA *rider-based HSA* de couleur rouge foncé.

Le FRHSA *Fractional Rider based HSA* permet à chacune des itérations d'augmenter la valeur de l'expérience.

3.3.3 Discussions

Le tableau 3.2 présente l'efficacité de notre approche par rapport à d'autres approches de *cloud gaming* en utilisant les paramètres MOS, QE, équité, énergie et délai. Aussi, nous observons que le concept FC améliore la performance de notre RHSA. Le FRHSA permet de minimiser l'énergie de 34,4% par rapport à l'algorithme d'optimisation basé sur le jeu potentiel.
Pour la taille du jeu de 300 et 500, nous avons un gain d'énergie de 23.89 % et de 32.20% respectivement.

Le FRHSA améliore les métriques par rapport à notre précédent algorithme.

Tableau 3. 2 Tableau comparatif 2

Jeu de taille	Métriques	Potential game based optimization algorithm	Proactive resource allocation algorithm	QoE-aware resource allocation algorithm	Rider-HSA	Proposed Fractional Rider-HSA
200	MOS	0.7997	0.8250	0.8380	0.8483	**0.8961**
	QE	0.9513	0.9877	0.9853	0.9994	**0.9997**
	Fairness	0.4008	0.5371	0.5459	0.5475	**0.8210**
	Energy	0.4742	0.3949	0.3855	0.3439	**0.3109**
	Delay	0.2889	0.2298	0.2286	0.2279	**0.2266**
300	MOS	0.7877	0.7993	0.8310	0.8732	**0.8831**
	QE	0.9523	0.9651	0.9877	0.9997	**0.9998**
	Fairness	0.5381	0.9777	0.9900	0.9990	**0.9991**
	Energy	0.4658	0.3711	0.3632	0.3578	**0.3545**
	Delay	0.5560	0.4862	0.4619	0.4618	**0.4601**
500	MOS	0.7578	0.7765	0.7777	0.7787	**0.7803**
	QE	0.9205	0.9386	0.9691	0.9795	**0.9999**
	Fairness	0.5647	0.7976	0.8698	0.9183	**0.9984**
	Energy	0.4689	0.3941	0.3621	0.3459	**0.3179**
	Delay	1.7211	1.2798	1.2704	1.2653	**1.2353**

3.4 Conclusion

Ce chapitre présente l'algorithme d'allocation des ressources qui tient compte de la consommation d'énergie. Notre méthode permet l'allocation des ressources en minimisant la consommation de l'énergie au sein de datacenters. Cette méthode appelée FRHSA pour l'allocation des ressources, augmente l'efficacité du système de *cloud computing* avec un MOS maximal de 0,8961, un QE maximal de 0,998, une équité maximale de 0,9991, une consommation d'énergie minimale de 0,3109 et un délai minimal de 0,2266. Il améliore l'efficacité du modèle de notre première contribution.

4 CHAPITRE 4 : Proposition d'une approche d'allocation prédictive des ressources *cloud*

Sommaire

4.1 Introduction

Les services *Cloud gaming* possède en réalité un comportement dynamique au fil de temps, comme le montre la figure 4.1 ci-dessous. Par exemple, pour le cas du jeu WoW (World of Warcraft), les avatars se déplacent d'une zone virtuelle à une autre. Ces mouvements interzones rendent la charge d'une zone bien déterminée variable en fonction du temps. Afin de gérer intelligemment les ressources *Cloud*, réduire les coûts et les délais correspondants, la stratégie dynamique d'allocation des VMs s'avère la plus efficace.

L'allocation dynamique des ressources peut être effectuée de deux manières, à savoir avec une approche réactive et avec une approche proactive [115].
Dans le cadre d'une approche réactive, on définit des seuils de sous-utilisation et de surutilisation des ressources. Lorsque la charge de travail atteint la valeur seuil, le processus de redimensionnement automatique prend des mesures en fonction de l'état actuel des ressources, comme la suppression de machines virtuelles pour un état de sous-utilisation ou l'ajout de machines virtuelles aux services *cloud* pour un état de surutilisation. Le principal inconvénient de ce processus est que le processus de redimensionnement automatique a du mal à effectuer les opérations de redimensionnement en cas de fluctuations soudaines de la charge de travail.
Une approche proactive permet quant à elle d'effectuer des opérations de redimensionnement à l'avance [116]. Le *cloud* prévoit la charge de travail future et alloue les ressources aux services en fonction de la valeur prévue.

Dans un état de sur-approvisionnement, davantage de ressources seront allouées aux applications si nécessaire. Selon les accords de niveau de service (SLA), il s'agit d'un avantage pour les utilisateurs, mais pour les fournisseurs, c'est un coût inutile qui entraîne une consommation d'énergie élevée et augmente ainsi les coûts d'exploitation. En cas de sous-dimensionnement, moins de ressources seront allouées aux applications en nuage, ce qui entraînera des violations des accords de niveau de service, une baisse de la qualité d'expérience et, en fin de compte, une perte des consommateurs et des revenus.
Une approche efficace et proactive doit donc prévoir avec précision les ressources futures nécessaires pour atteindre la qualité d'expérience. La mesure la plus importante des modèles de prédiction de la charge de travail du système est la précision, qui est mesurée par la différence entre les résultats prédits et réels [117]. En général, plus la valeur prédite est proche de la valeur réelle, meilleur est le modèle.

Actuellement, il existe de nombreuses techniques et méthodes appliquées à la prédiction de la charge de travail des systèmes informatiques, telles que l'approche d'ensemble e-learning [118], les modèles ARIMAR (Auto Regressive and Moving Average) [119] , les réseaux de neuronnes récurrents (RNN) [120], les réseaux LSTM (Long Short Term Memory) [121]. Cependant, décider de la quantité exacte de ressources avec des approches proactives pendant le temps d'exécution des services n'est pas une tâche triviale.

Afin de traiter des problèmes ci-mentionnés et améliorer notre précédent modèle d'allocation, nous proposons dans ce chapitre une méthode d'allocation prédictive basée sur le réseau LSTM.

4.2 Méthode proposée pour la prédiction de la charge de travail

Cette section présente le réseau *Fractional Rider Deep LSTM* (FRDL) pour la prédiction de la charge de travail dans le *cloud gaming*. La charge de travail de chaque allocation de ressources est prédite par le réseau FRDL et l'allocation des ressources est faite par le FRHSA. Le modèle *Fractional Rider Deep LSTM* est développé en combinant le modèle FC et le réseau Rider Deep LSTM.

L'objectif principal de notre approche est de trouver les ressources optimales pour la charge prédite et ainsi améliorer l'allocation des ressources pour l'utilisateur.

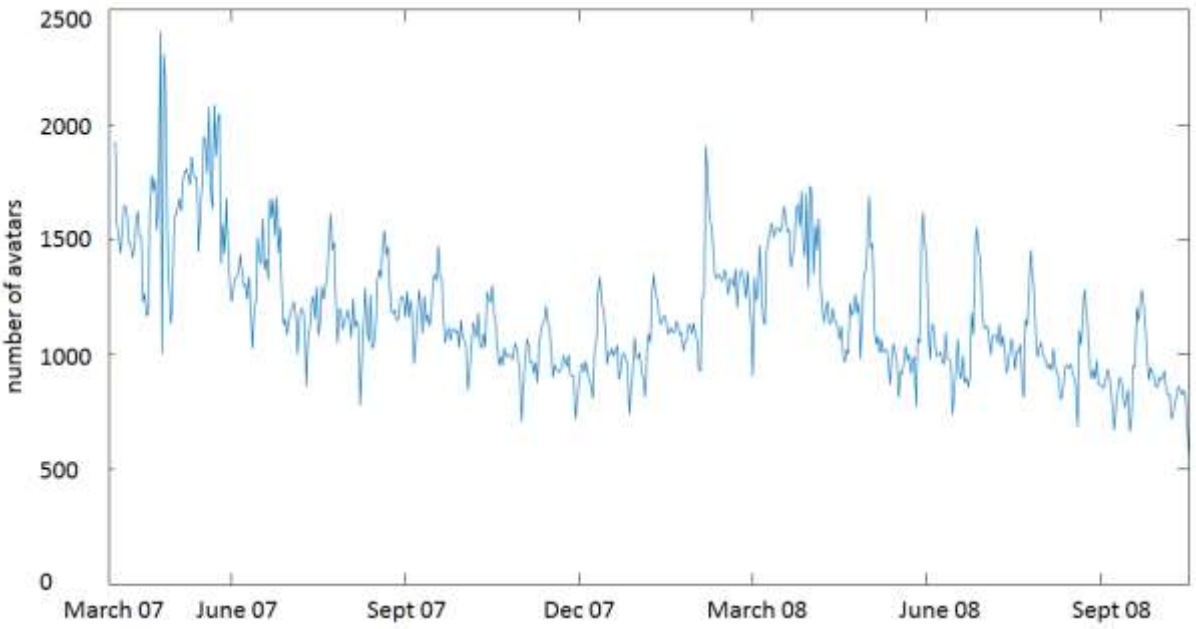

Figure 4. 1 Charge de service MMOG [16]

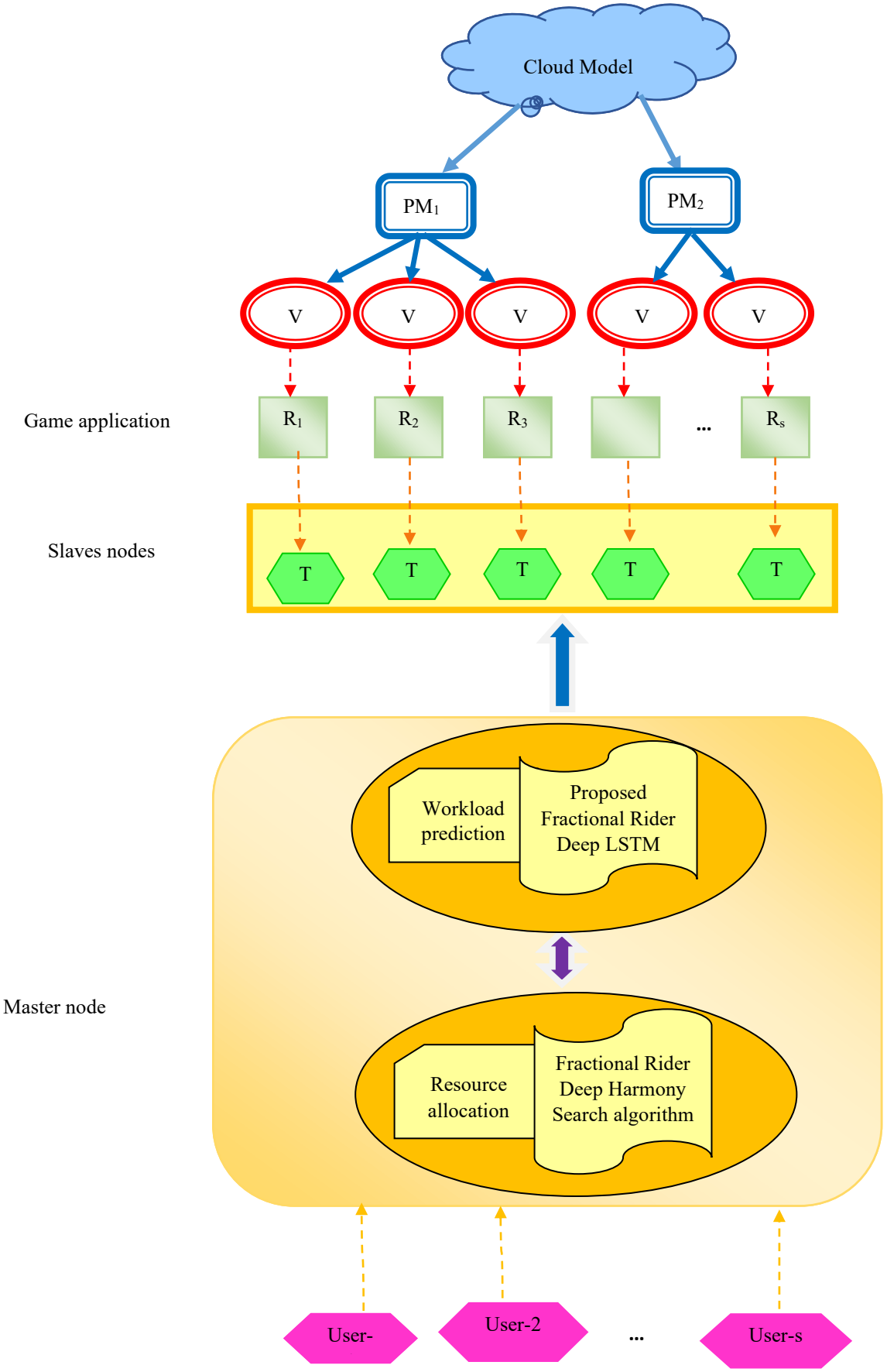

Figure 4. 2 Schéma fonctionnel pour l'approche prédictive

4.2.1 Modèle multi-objectif

Notre fonction objectif est définie comme suit,

$$FO = (1 - QE) + MOS + J^T + N + (1 - E) + L + L_p \tag{4.1}$$

où, L représente la charge et L_p est la charge prédictive. Le terme QE fait référence à la perte d'expérience du joueur, MOS le score d'opinion moyen, J^T l'équité, N désigne un facteur de définition du réseau, et le terme E représente la consommation énergétique.
La charge de la VM est représentée comme,

$$L_{r,i} = \sum_{e=1}^{n} \frac{(V_p + V_x + V_a + V_g + V_f) * A}{max(V_p + V_x + V_a + V_g + V_f) * n} * \frac{1}{N} \tag{4.2.}$$

où, n spécifie le nombre de jeux, N est le facteur de normalisation, V_p indique le nombre d'éléments de traitement dans la VM, V_x indique les unités de mémoire dans la VM, V_a est une composante de la bande passante dans la i^{th} VM, V_g représente l'élément MIPS dans la VM et V_f spécifie la composante de fréquence dans la VM.

$$A = \begin{cases} 1 \; ; if \, e^{th} game \; is \; run \; by \; i^{th} \; VM \\ 0 \; ; otherwise \end{cases} \tag{4.3}$$

La charge de PM est formulée par,

$$L_r = \sum_{i=1}^{f} L_{r,i} \tag{4.4}$$

Les autres paramètres de l'équation ont été déjà définis dans la section 3.1.2. et 4.2.1

4.2.2 Modèle FRDL pour la prédiction de la charge de travail

Le modèle FRDL est proposé pour prédire la charge de travail. Le FC[113] est utilisé pour résoudre les problèmes d'équation intégrale et d'équation dérivée. Les équations différentielles et l'intégrale d'ordre fractionnaire sont traitées à l'aide de transformées de Laplace. Le processus FC est illustré comme suit : la première étape détermine la transformée de Laplace. La deuxième phase est utilisée pour résoudre la transformation des fonctions indéfinies, et la dernière étape utilise la transformée de Laplace inverse pour identifier une solution optimale. Le Rider Deep

LSTM [122] possède quatre couches et améliore la prédiction des défauts en temps réel. Par conséquent, le FC et le Rider Deep LSTM sont combinés pour une meilleure performance.
En se basant sur le Rider Deep LSTM, la règle de mise à jour de l'overtaker est exprimée comme suit,

$$\varphi_d^{rider} = \varphi_{d-1} + [S_{d-1}^G * \varphi^V] \tag{4.5}$$

$$\varphi_d^{rider} - \phi_{d-1} = S_{d-1}^G * \varphi^V \tag{4.6}$$

Pour FC,

$$S^{\lambda}\left[\varphi_d^{rider}\right] = S_{d-1}^G * \varphi^V \tag{4.7}$$

$d + 1 = d$,

$$\varphi_d^{rider} - \lambda\varphi_{d-1}^{rider} - \frac{1}{2}\lambda\varphi_{d-2}^{rider} - \frac{1}{6}(1-\lambda)\varphi_{d-3}^{rider} - \frac{1}{24}(1-\lambda)(2-\lambda)\varphi_{t-4}^{rider} = S_{d-1}^G * \varphi^V (4.8)$$

$$\varphi_d^{rider} = \lambda\varphi_{d-1}^{rider} + \frac{1}{2}\lambda\varphi_{d-2}^{rider} + \frac{1}{6}(1-\lambda)\varphi_{d-3}^{rider} + \frac{1}{24}\lambda(1-\lambda)(2-\lambda)\varphi_{d-4}^{rider} + (S_{d-1}^G * \varphi^V) (4.9)$$

où, λ nombre aléatoire compris entre [0,1], S_{d-1}^G indicateur de direction o ϕ_{d-1} et ϕ^V le leader

4.3 Résultats et discussions

Nous évaluons l'efficacité de notre approche prédictive en nous basant sur le délai, la consommation énergétique, l'équité, le MOS et la QE. L'évaluation est faite en considérant différentes tailles de jeu 200 et 300 et en variant la taille de la population de 10 à 50. Par la suite, notre approche a été comparée à d'autres approches de la littérature.

Nos différentes expériences ont été exécutées en utilisant le simulateur *cloudSim* [108] sous PYTHON avec un PC contenant 12GB de RAM, Windows 10 OS, ROM-plus de 100GB, CPU core i7 2.2 GHz.
Le nombre de machines physiques utilisées est égal à 6, le nombre total de machines virtuelles est équivalent au nombre de tâches

4.3.1 Mesures de performance

4.3.1.1 Evaluation de l'approche prédictive avec une taille de tâche 200

4.3.1.1.1 Evaluation de l'approche prédictive sur le délai avec une taille de tâche 200

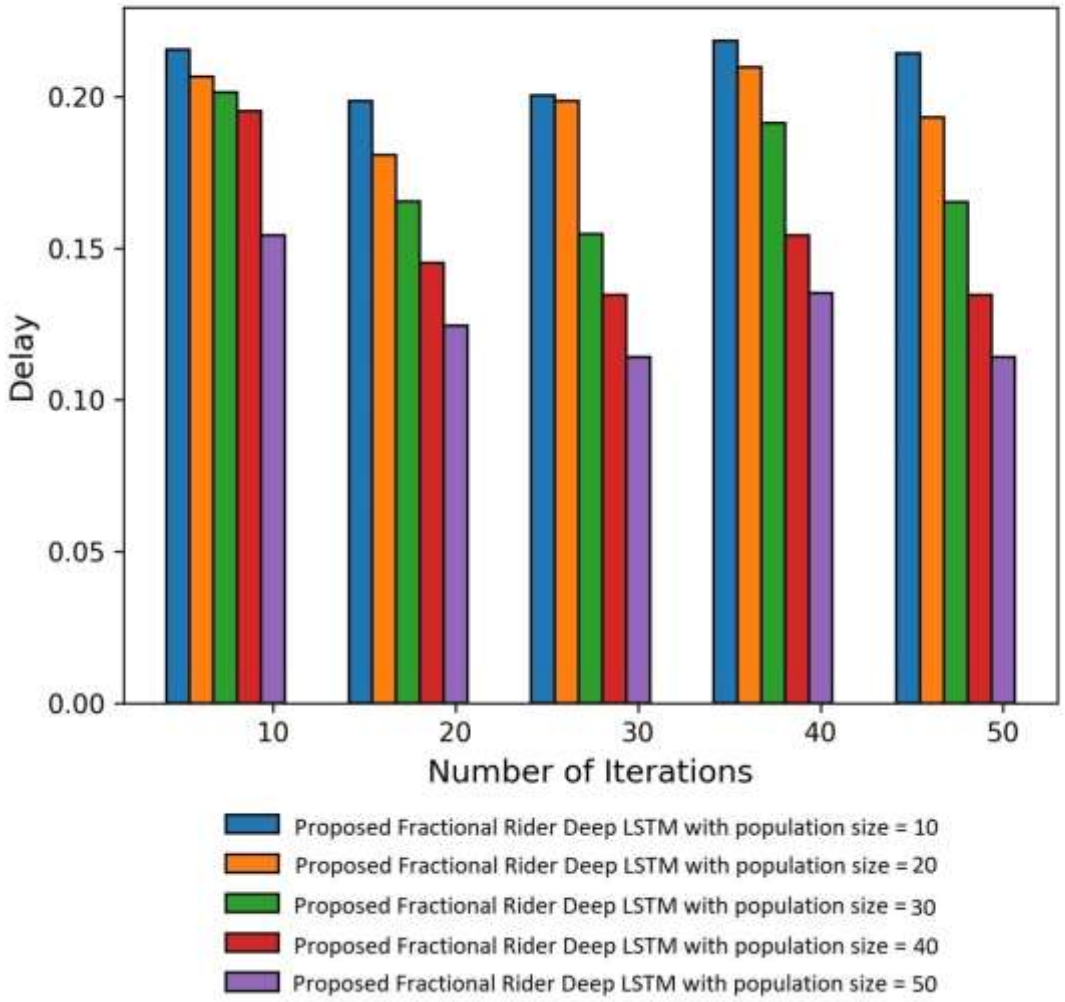

Figure 4. 3 Evaluation de l'approche prédictive sur le délai avec une taille de tâche 200

La figure 4.3 ci-dessus présente l'évaluation de notre approche prédictive basé sur le LSTM sur le délai de traitement à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, le délai évalué par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.216, 0.207, 0.201, 0.195, et 0.154. Ces résultats montrent que plus la taille de la population augmente, la valeur du délai baisse.
Nos résultats, de l'impact de notre approche prédictive sur le délai, montrent qu'avec un nombre élevé d'itérations, nous améliorons la valeur de délai.

4.3.1.1.2 Evaluation de l'approche prédictive sur l'énergie avec une taille de tâche 200

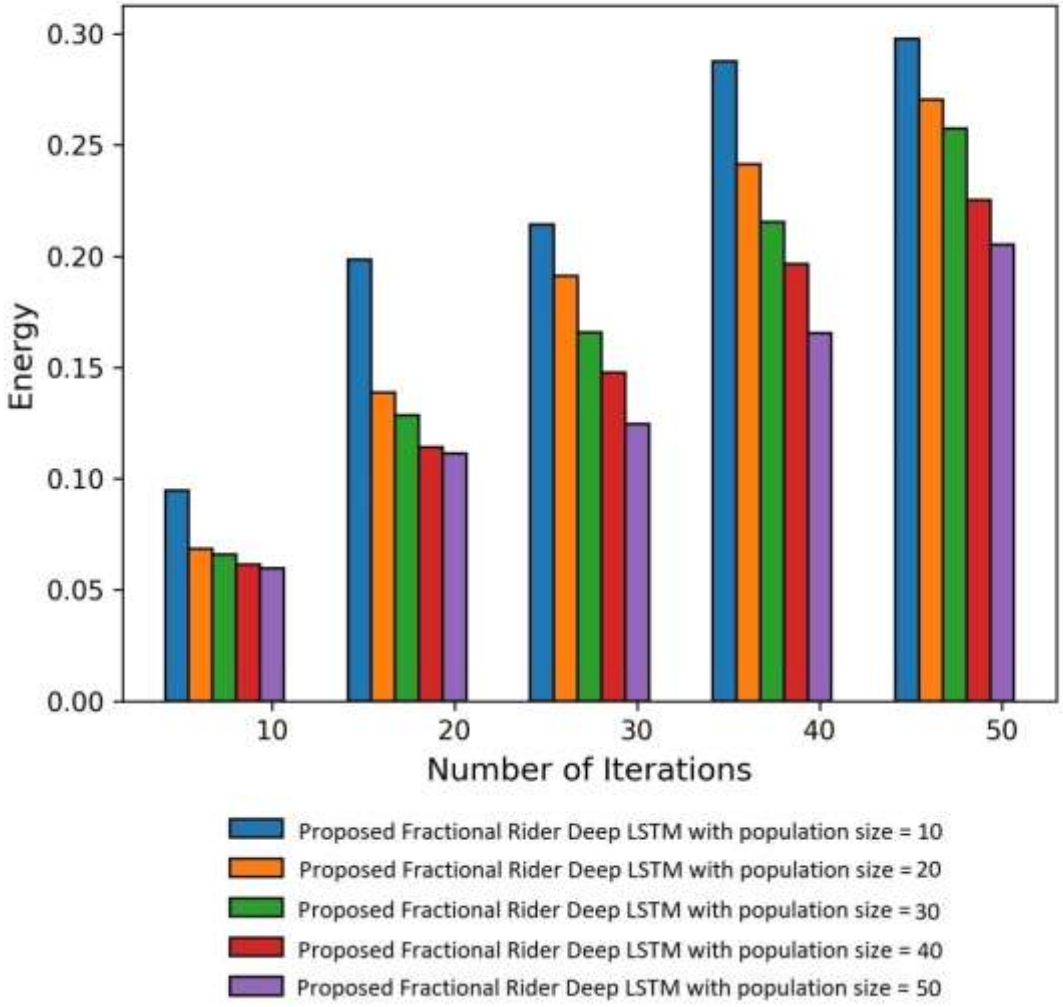

Figure 4. 4 Evaluation de l'approche prédictive sur l'énergie avec une taille de tâche 200

La figure 4.4 présente l'évaluation de notre approche prédictive basé sur le LSTM sur le délai de traitement à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, le délai évalué par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.095, 0.068, 0.066, 0.061, et 0.060. Ces résultats montrent que plus la taille de la population augmente, la valeur de l'énergie baisse.

Nos résultats, de l'impact de notre approche prédictive sur le délai, montrent qu'à chacune des itérations, notre algorithme améliore la consommation énergétique.

4.3.1.1.3 Evaluation de l'approche prédictive sur l'équité avec une taille de tâche 200

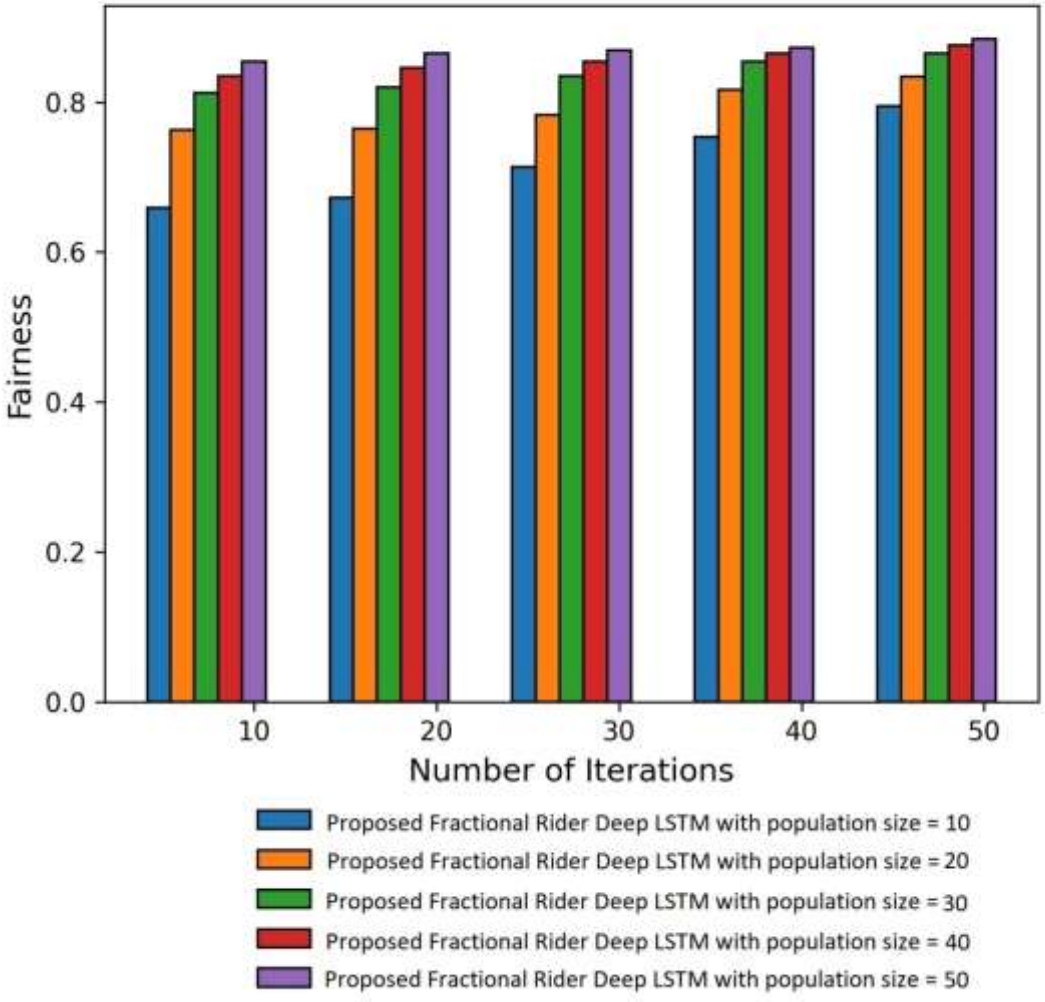

Figure 4. 5 Evaluation de l'approche prédictive sur l'équité avec une taille de tâche 200

La figure 4.5 est l'évaluation de notre approche prédictive basé sur le LSTM sur l'équité à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, le délai évalué par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.659, 0.762, 0.812, 0.835, et 0.854. Ces résultats montrent que plus la taille de la population augmente, la valeur de l'équité augmente.

Nos résultats, de l'impact de notre approche prédictive sur l'équité, montrent qu'à chacune des itérations, notre algorithme améliore la valeur de l'équité.

4.3.1.1.4 Evaluation de l'approche prédictive sur le MOS avec une taille de tâche 200

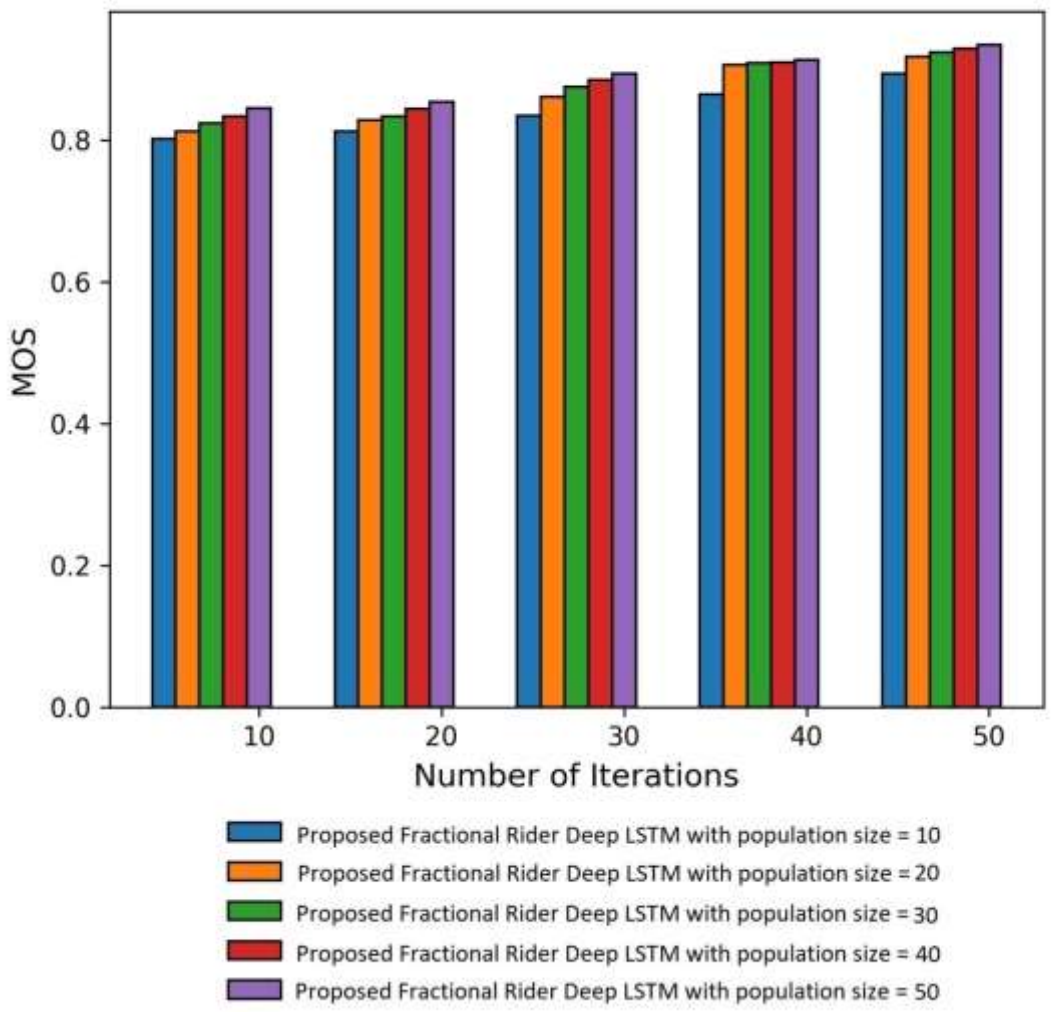

Figure 4. 6 Evaluation de l'approche prédictive sur le MOS avec une taille de tâche 200

La figure 4.6 est l'évaluation de notre approche prédictive basé sur le LSTM sur le MOS à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, le délai évalué par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.802, 0.813, 0.824, 0.835, et 0.846. Ces résultats montrent que plus la taille de la population augmente, la valeur MOS augmente.

Nos résultats montrent qu'à chacune des itérations, notre approche améliore la valeur du MOS

4.3.1.1.5 Evaluation de l'approche prédictive sur le QE avec une taille de tâche 200

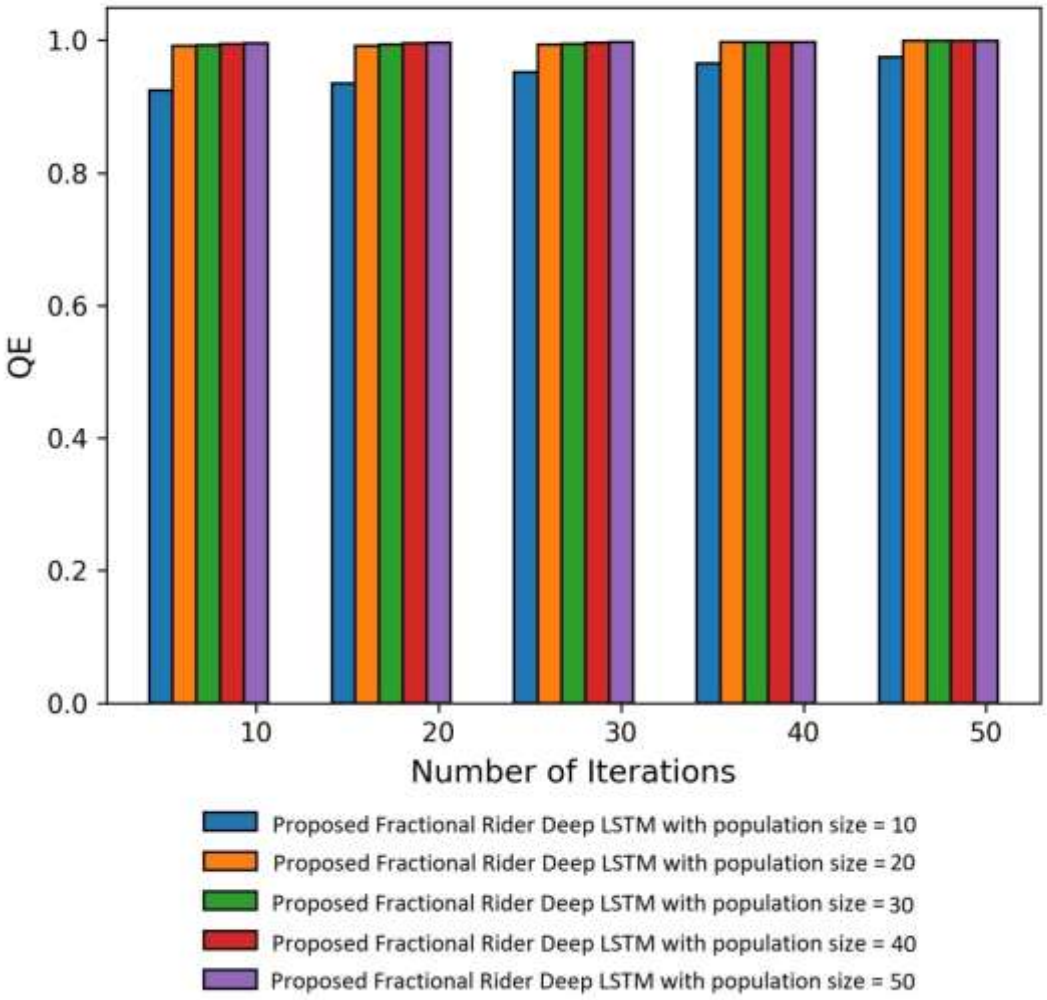

Figure 4. 7 Evaluation de l'approche prédictive sur le QE avec une taille de tâche 200

La figure 4.7 est l'évaluation de notre approche prédictive basé sur le LSTM sur l'expérience QE à partir d'une taille de tâche égale à 200. Nous avons fait varier la taille de la population de 10 à 50.
Pour 10 itérations, la QE évaluée par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.925, 0.992, 0.992, 0.995, et 0.996. Ces résultats montrent que plus la taille de la population augmente, la valeur QE ici augmente légèrement.

Nos résultats montrent qu'à chacune des itérations, notre approche améliore la valeur du QE.

La section ci-dessus montre l'évaluation de la performance de notre approche prédictive avec une taille plus grande 300.

4.3.1.2 Evaluation de l'approche prédictive avec une taille de tâche 300

4.3.1.2.1 Evaluation de l'approche prédictive sur le délai avec une taille de tâche 300

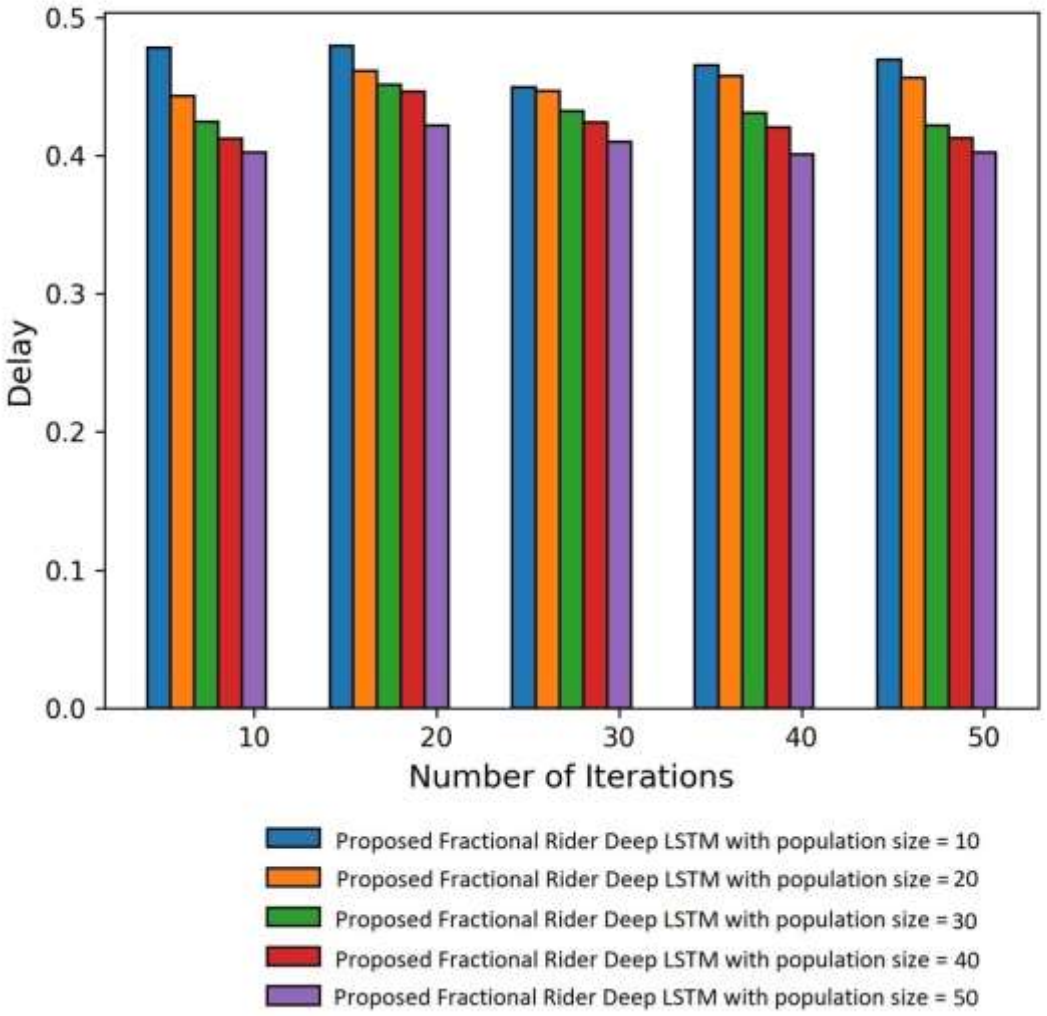

Figure 4. 8 Evaluation de l'approche prédictive sur le délai avec une taille de tâche 300

La figure 4.8 ci-dessus présente l'évaluation de notre approche prédictive basé sur le LSTM sur le délai de traitement à partir d'une taille de tâche égale à 300. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, le délai évalué par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.478, 0.443, 0.425, 0.412, et 0.402. Ces résultats montrent que plus la taille de la population augmente, la valeur du délai baisse.

Nos résultats, ici, montrent qu'avec un nombre élevé d'itérations, nous améliorons la valeur de délai.

4.3.1.2.2 Evaluation de l'approche prédictive sur l'énergie avec une taille de tâche 300

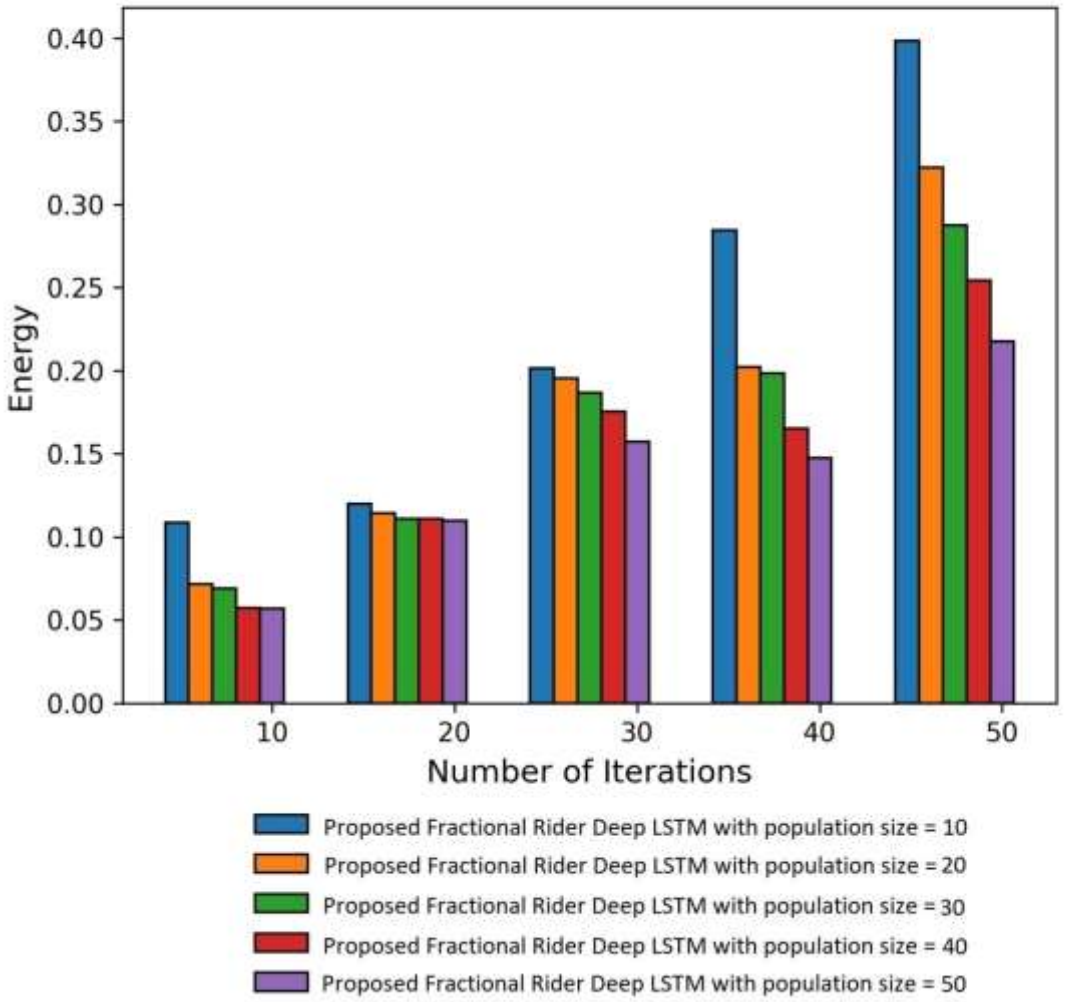

Figure 4. 9 Evaluation de l'approche prédictive sur l'énergie avec une taille de tâche 300

La figure 4.9 est l'évaluation de notre approche prédictive basé sur le LSTM sur l'énergie à partir d'une taille de tâche égale à 300. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, le délai évalué par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.109, 0.072, 0.069, 0.057, et 0.057.

Nos résultats, ici, montrent qu'avec un nombre élevé d'itérations, nous réduisons la consommation énergétique.

4.3.1.2.3 Evaluation de l'approche prédictive sur l'équité avec une taille de tâche 300

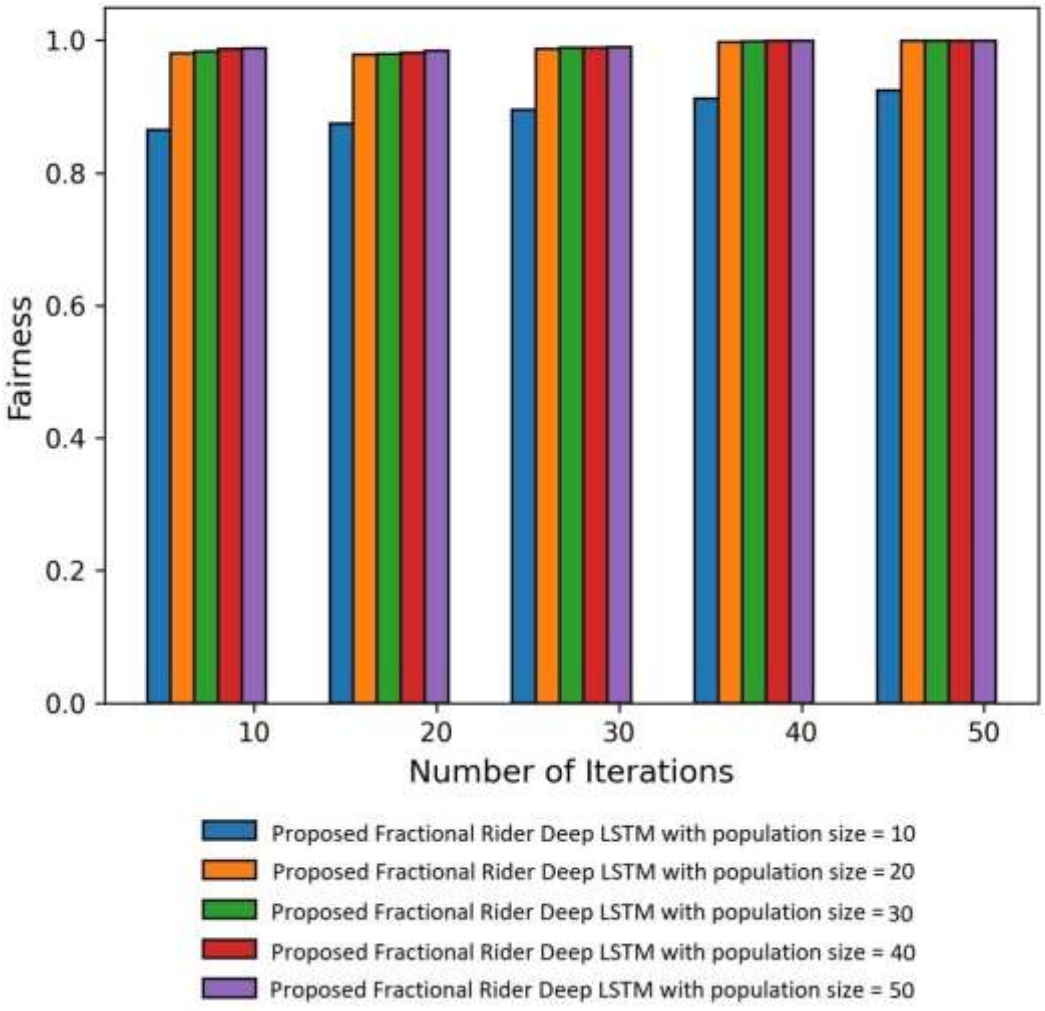

Figure 4. 10 Evaluation de l'approche prédictive sur l'équité avec une taille de tâche 300

La figure 4.10 est l'évaluation de notre approche prédictive basé sur le LSTM sur l'équité à partir d'une taille de tâche égale à 300. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, l'équité évalué par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.865, 0.981, 0.983, 0.988, et 0.989

Nos résultats, ici, montrent qu'avec un nombre élevé d'itérations, nous améliorons la valeur de l'équité.

4.3.1.2.4 Evaluation de l'approche prédictive sur le MOS avec une taille de tâche 300

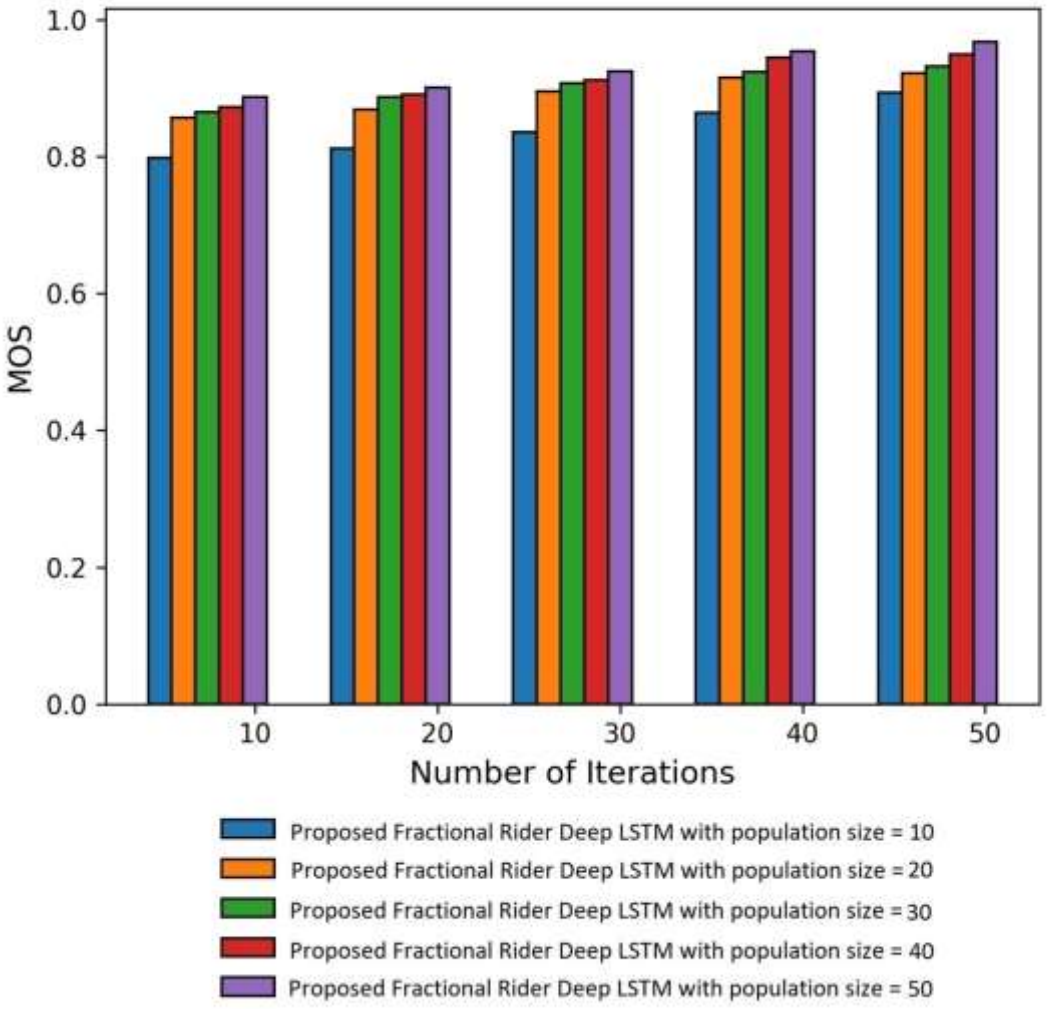

Figure 4. 11 Evaluation de l'approche prédictive sur le MOS avec une taille de tâche 300

La figure 4.11 est l'évaluation de notre approche prédictive basé sur le LSTM sur le MOS à partir d'une taille de tâche égale à 300. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, le MOS évalué par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.799, 0.857, 0.865, 0.872, et 0.888.

Nos résultats, ici, montrent qu'avec un nombre élevé d'itérations, nous améliorons la valeur du MOS

4.3.1.2.5 Evaluation de l'approche prédictive sur la QE avec une taille de tâche 300

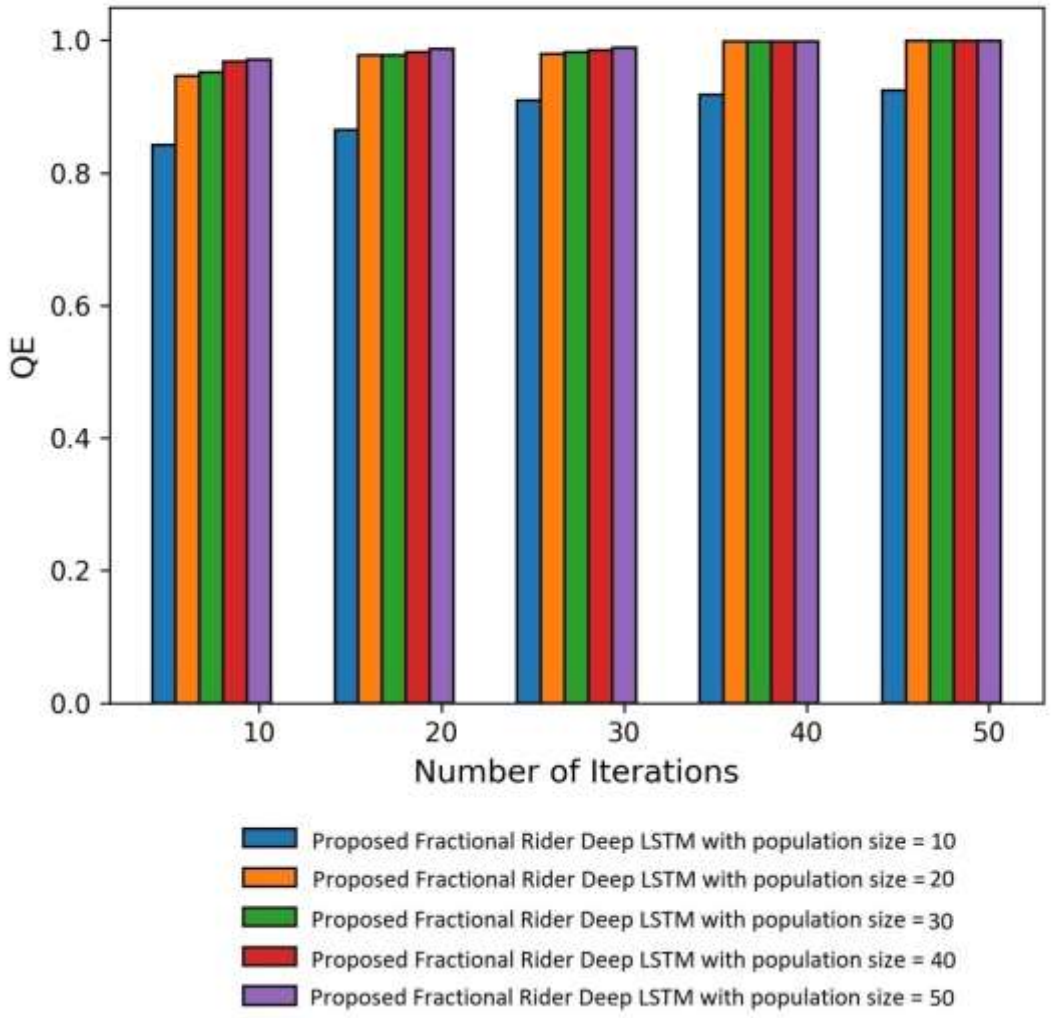

Figure 4. 12 Evaluation de l'approche prédictive sur la QE avec une taille de tâche 300

La figure 4.12 est l'évaluation de notre approche prédictive basé sur le LSTM sur l'expérience du joueur à partir d'une taille de tâche égale à 300. Nous avons fait varier la taille de la population de 10 à 50.

Pour 10 itérations, l'expérience évaluée par l'approche prédictive avec une population de taille 10, 20, 30, 40 et 50 est respectivement de 0.842, 0.947, 0.952, 0.969, et 0.971.

Nos résultats, ici, montrent qu'avec un nombre élevé d'itérations, nous améliorons l'expérience du joueur

Nous observons en somme que lorsque la taille de la population et le nombre d'itérations augmentent, les différentes métriques du modèle s'améliorent. Nos métriques sont directement proportionnelles à la taille de la population et au nombre d'itérations et supporte le passage à l'échelle.
L'allocation prédictive offre de meilleures performances par rapport à nos précédentes contributions.

4.3.2 Analyse comparative

Notre approche prédictive basée sur LSTM est analysée avec d'autres méthodes, telles que l'algorithme d'allocation des ressources en fonction de la QoE [67], l'algorithme d'allocation proactive des ressources [66], l'algorithme d'optimisation basé sur le jeu potentiel [65], notre première contribution [114] et notre deuxième contribution [123]
Cette analyse est effectuée en utilisant des paramètres tels que MOS, QE, équité, l'énergie consommée et le délai.

4.3.2.1 Analyse comparative avec une taille de tâche 200

4.3.2.1.1 Analyse comparative de l'approche prédictive en fonction de l'équité avec une taille de tâche 200

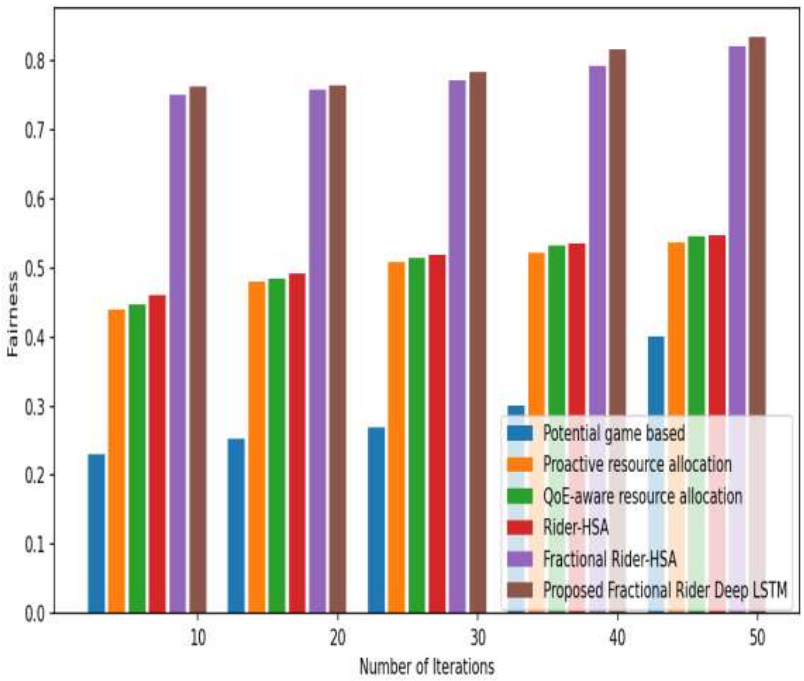

Figure 4. 13 Analyse comparative de l'approche prédictive en fonction de l'équité avec une taille de tâche 200

La figure 4.13 est la comparaison de notre approche prédictive avec d'autres méthodes sur l'équité à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive offre les meilleurs résultats à chaque itération.

4.3.2.1.2 Analyse comparative de l'approche LSTM en fonction de MOS avec une taille de tâche 200

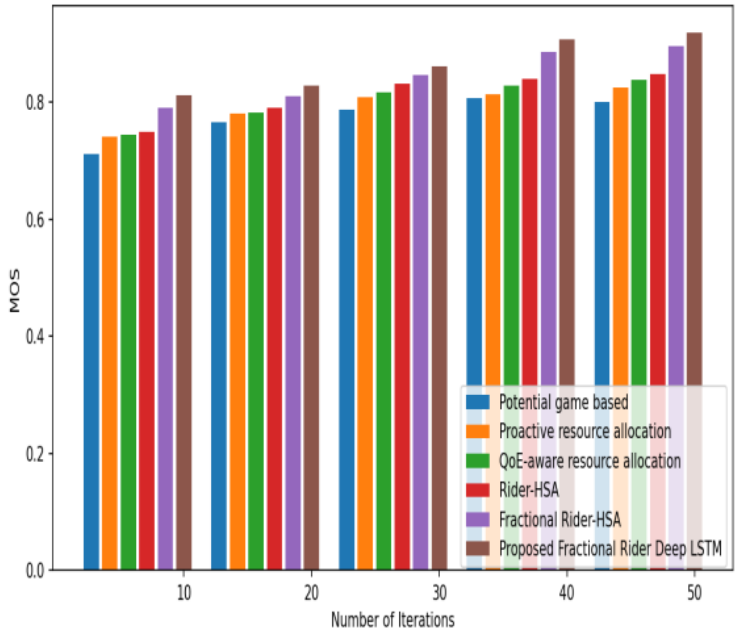

Figure 4. 14 Analyse comparative de l'approche LSTM en fonction de MOS avec une taille de tâche 200

La figure 4.14 est la comparaison de notre approche prédictive avec d'autres méthodes sur l'équité à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive offre les meilleurs résultats à chaque itération.

4.3.2.1.3 Analyse comparative de l'approche LSTM en fonction de QE avec une taille de tâche 200

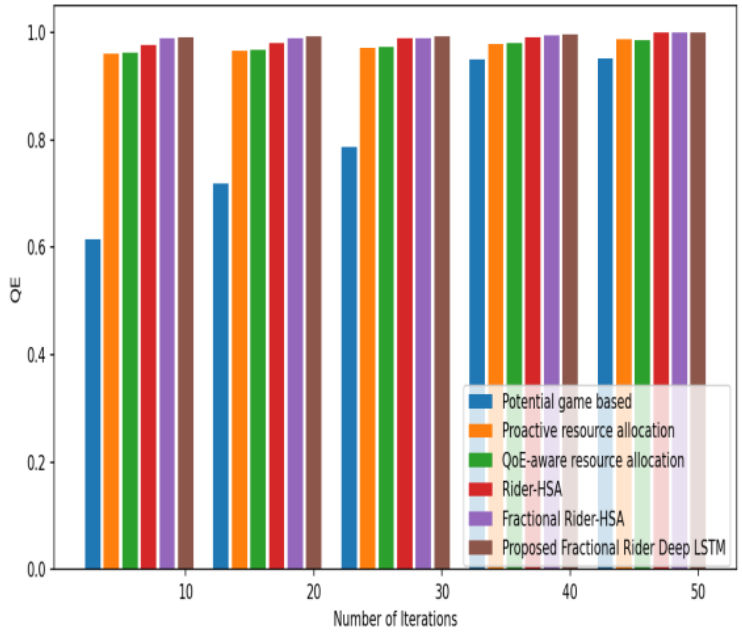

Figure 4. 15 Analyse comparative de l'approche LSTM en fonction de QE avec une taille de tâche 200

La figure 4.15 est la comparaison de notre approche prédictive avec d'autres méthodes sur l'équité à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive offre les meilleurs résultats de l'expérience à chaque itération.

4.3.2.1.4 Analyse comparative de l'approche LSTM en fonction de l'énergie avec une taille de tâche 200

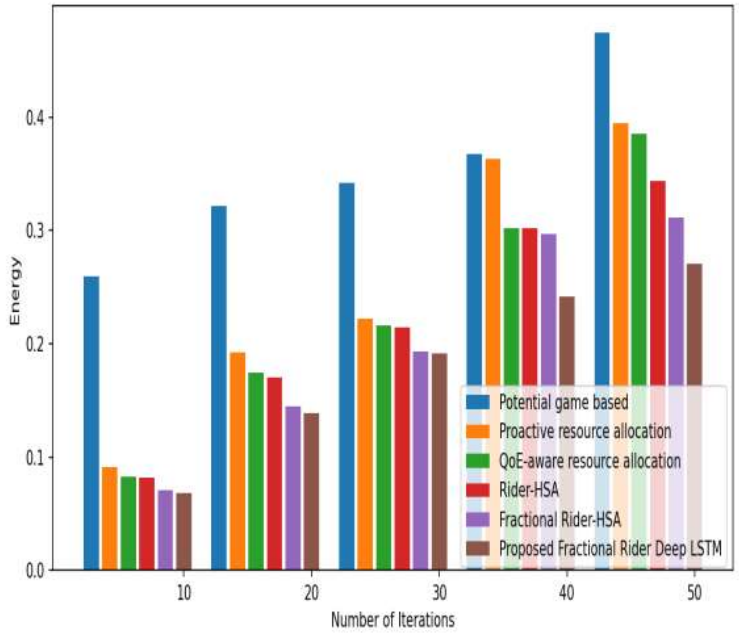

Figure 4. 16. Analyse comparative de l'approche LSTM en fonction de l'énergie avec une taille de tâche 200

La figure 4.16 est la comparaison de notre approche prédictive avec d'autres méthodes sur l'énergie à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive permet de réduire l'énergie à chaque itération.

4.3.2.1.5 Analyse comparative de l'approche LSTM en fonction de délai avec une taille de tâche 200

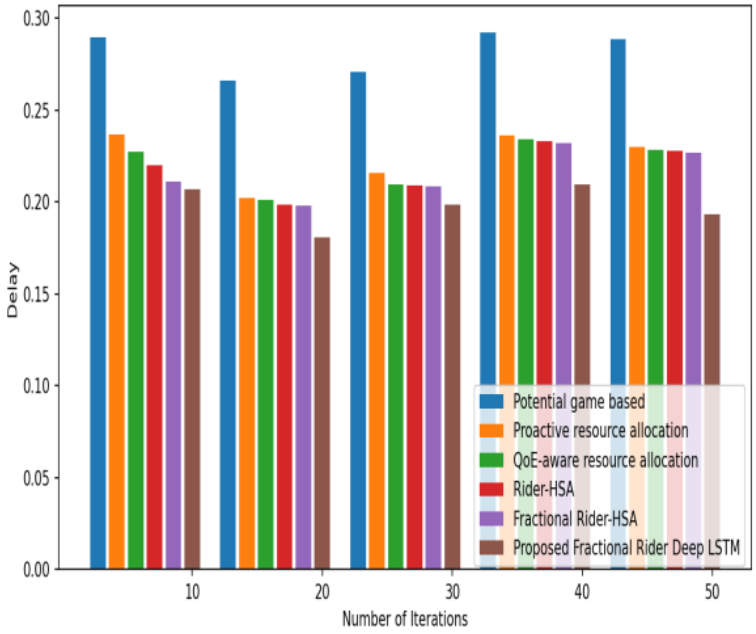

Figure 4. 17 Analyse comparative de l'approche LSTM en fonction de délai avec une taille de tâche 200

La figure 4.17 est la comparaison de notre approche prédictive avec d'autres méthodes sur le délai à partir d'une taille de tâche égale à 200. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive permet de réduire le délai de traitement à chaque itération.

Nous poursuivons notre évaluation comparative en augmentant encore la taille de tâches à 300

4.3.2.2 Analyse comparative avec une taille de tâche 300

4.3.2.2.1 Analyse comparative de l'approche prédictive en fonction de l'équité avec une taille de tâche 300

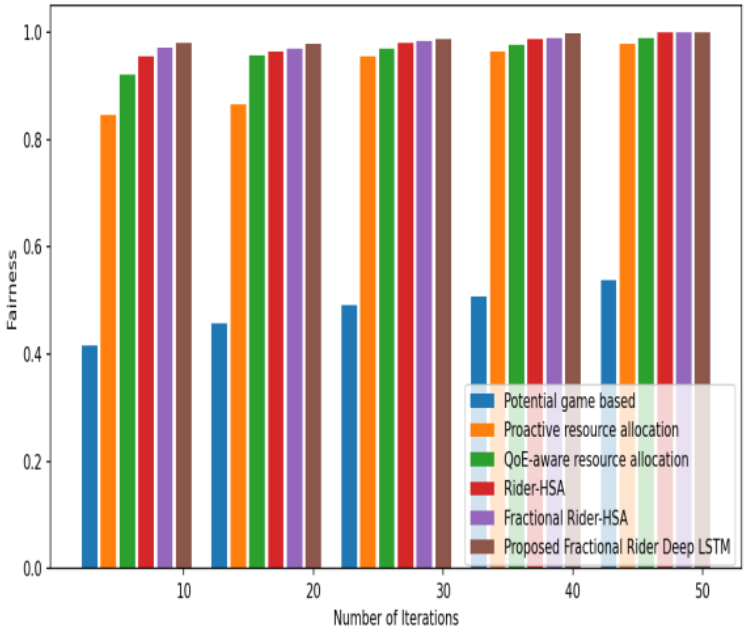

Figure 4. 18 Analyse comparative de l'approche prédictive en fonction de l'équité avec une taille de tâche 300

La figure 4.18 est la comparaison de notre approche prédictive avec d'autres méthodes sur l'équité à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive offre les meilleurs résultats d'équité à chaque itération.

4.3.2.2.2 Analyse comparative de l'approche LSTM en fonction de MOS avec une taille de tâche 300

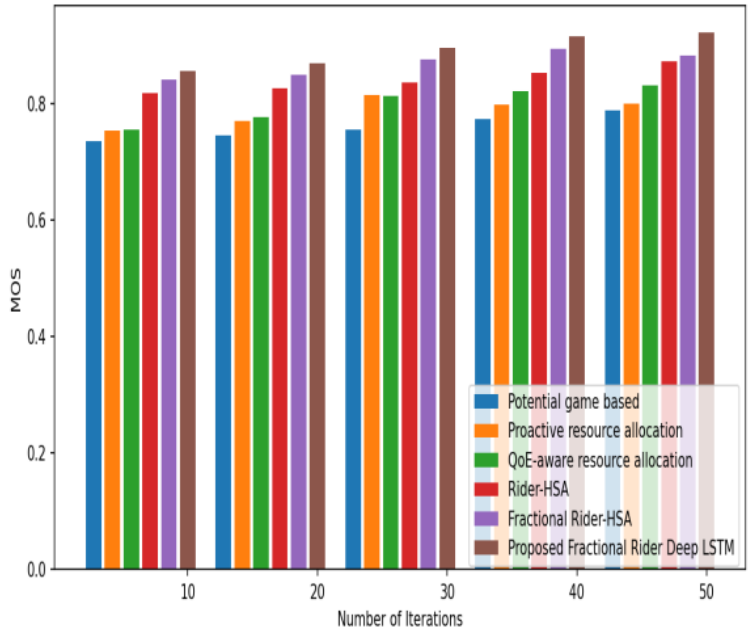

Figure 4. 19 Analyse comparative de l'approche LSTM en fonction de MOS avec une taille de tâche 300

La figure 4.19 est la comparaison de notre approche prédictive avec d'autres méthodes sur le MOS à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive obtient les meilleures valeurs de MOS à chaque itération.

4.3.2.2.3 Analyse comparative de l'approche LSTM en fonction de QE avec une taille de tâche 300

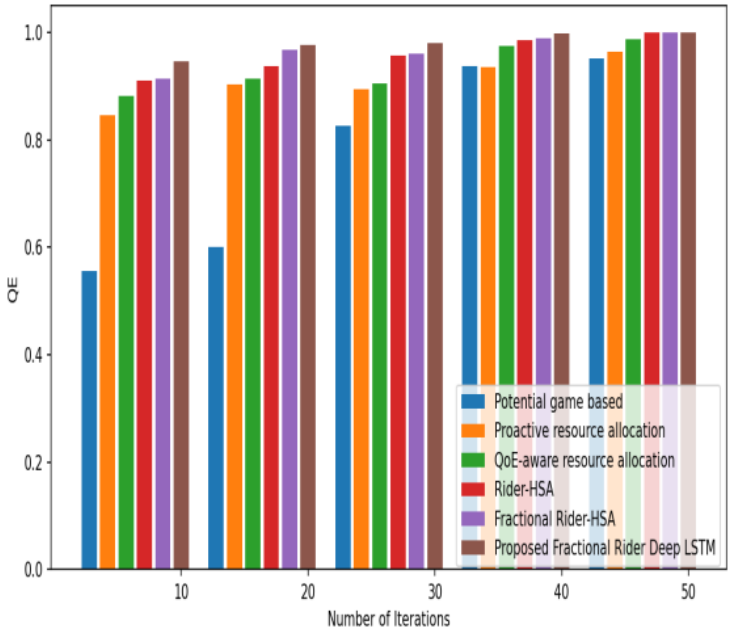

Figure 4. 20 Analyse comparative de l'approche LSTM en fonction de QE avec une taille de tâche 300

La figure 4.20 est la comparaison de notre approche prédictive avec d'autres méthodes sur l'expérience QE à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50.

Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive améliore l'expérience des joueurs à chaque itération.

4.3.2.2.4 Analyse comparative de l'approche LSTM en fonction de l'énergie avec une taille de tâche 300

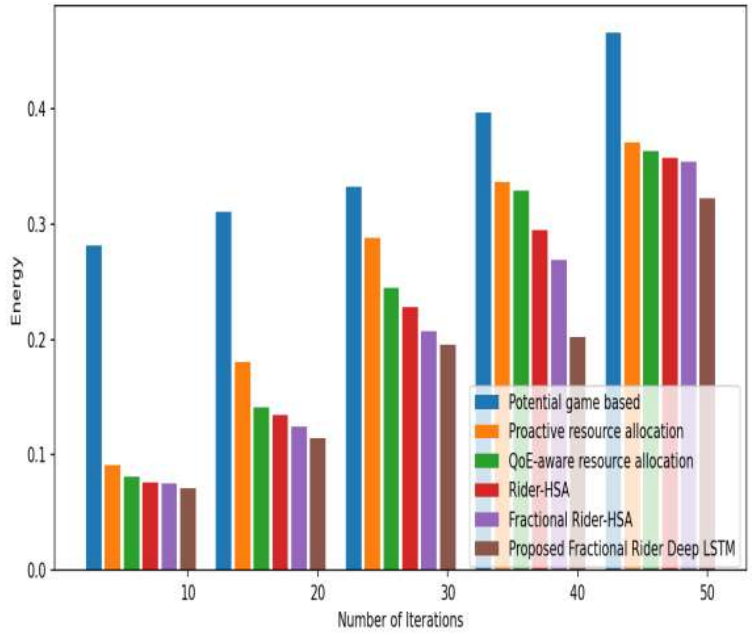

Figure 4. 21. Analyse comparative de l'approche LSTM en fonction de l'énergie avec une taille de tâche 300

La figure 4.21 est la comparaison de notre approche prédictive avec d'autres méthodes sur l'énergie à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive diminue la consommation énergétique des DC à chaque itération.

4.3.2.2.5 Analyse comparative de l'approche LSTM en fonction du délai avec une taille de tâche 300

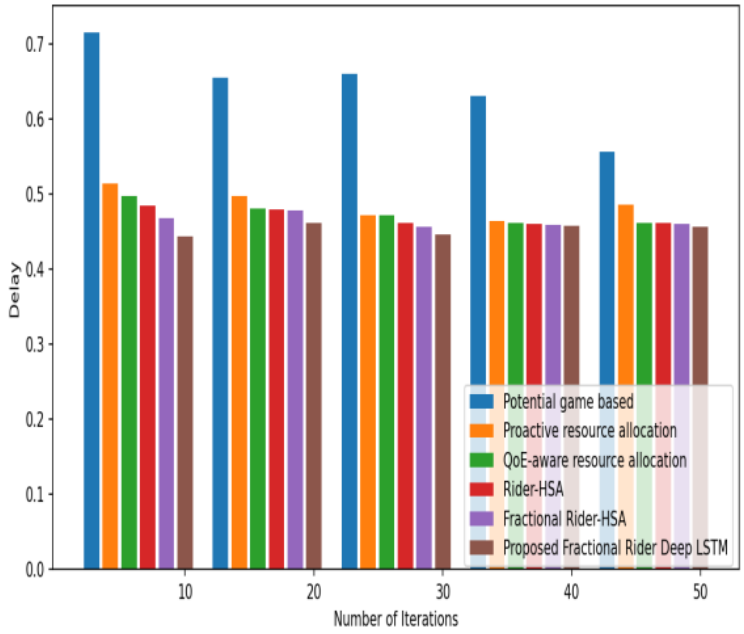

Figure 4. 22 Analyse comparative de l'approche LSTM en fonction du délai avec une taille de tâche 300

La figure 4.22 est la comparaison de notre approche prédictive avec d'autres méthodes sur le délai à partir d'une taille de tâche égale à 300. Nous fixons la taille de la population à 50. Nous comparons notre contribution de couleur marron avec notre précédente contribution FRHSA *Fractional Rider-HSA* en violet, avec la méthode *QoE-aware ressource allocation algorithm* de couleur verte, la méthode *proactive ressource allocation algorithm* de couleur orange, la méthode *potential game based optimization algorithm* de couleur bleue, et notre première contribution le RHSA *rider-based HSA* de couleur rouge foncé.

Notre approche d'allocation prédictive diminue le délai de traitement à chaque itération.

Nos différents résultats montrent qu'une approche prédictive d'allocation améliorent les métriques considérons dans cette thèse.

4.3.3 Discussions

D'après le tableau 4.1, le maximum d'équité, de MOS et de QE est obtenu lorsque la taille du jeu est de 300 et le minimum d'énergie et de retard est obtenu lorsque la taille du jeu est de 200. L'approche prédictive nous offre les meilleures performances dans d'allocation des ressources. Par exemple, nous obtenons avec une taille de tâche de 200 un maximal d'équité (34.41% par rapport au RHSA), un MOS maximal (7.6% par rapport au RHSA), un QE et moins de retard (15% par rapport au RHSA) et moins de consommation d'énergie (21,3% par rapport au RHSA).

*Tableau 4. 1*Tableau comparatif 3

Taille de Jeu	Métrique	Potential game-based optimization algorithm	Proactive resource allocation algorithm	QoE-aware resource allocation algorithm	Contribution 1 RHSA	Contribution 2 FRHSA	Contribution 3 FRDL
200	Equité	0.400	0.537	0.545	**0.547**	**0.820**	**0.834**
	MOS	0.799	0.824	0.837	**0.848**	**0.896**	**0.918**
	QE	0.951	0.987	0.985	**0.995**	**0.997**	**0.999**
	Energie	0.474	0.394	0.385	**0.343**	**0.310**	**0.270**
	Délai	0.288	0.229	0.228	**0.227**	**0.226**	**0.193**
300	Equité	0.538	0.977	0.989	**0.994**	**0.997**	**0.999**
	MOS	0.787	0.799	0.830	**0.873**	**0.883**	**0.921**
	QE	0.952	0.965	0.987	**0.996**	**0.998**	**0.999**
	Energie	0.465	0.371	0.363	**0.357**	**0.354**	**0.322**
	Délai	0.556	0.486	0.461	**0.461**	**0.460**	**0.456**

4.4 Conclusion

Ce chapitre présente une méthode de prédiction de la charge de travail basée sur le réseau LSTM. D’abord, la charge de travail de chaque ressource est estimée à l'aide du réseau LSTM. Ensuite, l'allocation des ressources est effectuée sur la base du FRHSA. Notre technique d’allocation prédictive a montré de bonnes performances avec une équité maximale de 0,999, un MOS de 0,921, un QE de 0,999, une énergie minimale de 0,322 et un retard de 0,456 pour une taille de tâche de 300.

Notre approche d’allocation prédictive améliore l’efficacité de nos précédents modèles.

CONCLUSION GENERALE & PERSPECTIVES

La tendance technologique actuelle est de faire appel à des services délocalisés fonctionnant au sein de *datacenters* distribués dans le monde entier. Ce nouveau paradigme de nom *cloud computing* subit lui aussi une évolution fulgurante due à l'attractivité de ses propriétés intrinsèques. Les services de *cloud computing* font partie aujourd'hui de notre quotidien. Parmi ses services, le *cloud gaming* ou encore le Gaas connait une large expansion en masse budgétaire et en nombre d'utilisateurs, chose qui a entrainé une concurrence accrue entre ses différents fournisseurs. Le *cloud gaming* s'est montré particulièrement dynamique alors que la pandémie de Covid-19 a affecté tous les pans de l'économie mondiale [124].

Les problématiques principales d'un fournisseur de service actuel est la garantie d'une qualité d'expérience acceptable de jeu, le dimensionnement efficace de son infrastructure et la gestion économique et rentable de ses ressources.

Les travaux présentés dans ce manuscrit ont porté sur l'allocation optimale des ressources dans un environnement *cloud computing* pour les jeux en ligne. Notre problématique était de proposer des mécanismes appropriés pour une gestion de ressources équitable, économique et efficace permettant principalement de garantir une qualité d'expérience pour les utilisateurs des services de *cloud gaming*.

Dans le chapitre 1, nous avons présenté le concept du *cloud computing* et plus particulièrement les services de jeux *cloud*, connus sous le nom de *cloud gaming*. Aussi, dans la littérature, plusieurs recherches ont discuté du problème d'optimisation de l'allocation des ressources *cloud* avec pour objectif d'améliorer la qualité d'expérience (QoE) des utilisateurs. Mais la plupart de ces recherches ont utilisé un seul critère d'évaluation ou deux critères d'évaluation. Alors, qu'on pourrait utiliser d'autres critères de QoE pour une meilleure efficacité de l'allocation de ressources basée sur la QoE.

Dans le chapitre 2, notre première contribution a proposé un algorithme d'optimisation hybride nommé RHSA pour une allocation des ressources afin d'améliorer l'efficacité du système de *cloud*. Cet algorithme est le résultat d'une combinaison de deux méta-heuristiques à savoir le ROA *Rider Optimization Algorithm* et le HSA *Harmony Search Algorithm*. La méthode d'optimisation a été développée pour résoudre les problèmes d'allocation de ressources avec pour objectif l'amélioration de la QoE des utilisateurs. Cette méthode considère trois critères, comme le MOS, la perte d'expérience, et le facteur d'équité.

Dans le chapitre 3, pour répondre à la préoccupation de réduire le coût d'exploitation des fournisseurs de jeux en ligne, notre seconde contribution a proposé une approche de provisionnement de ressources qui améliore l'efficacité énergétique des *datacenters cloud*. En effet, une meilleure allocation des ressources pourrait réduire les frais liés à la consommation énergétique et ainsi les coûts d'exploitation. Ce nouvel algorithme d'allocation des ressources FRHSA est une amélioration de notre précédente méthode en utilisant le concept de calcul fractionnaire. Basé sur l'architecture Spark, il permet d'obtenir des solutions optimales en tenant compte du paramètre énergétique.

Dans le chapitre 4, pour assurer une QoE robuste contre la variation de la charge de travail et empêcher la dégradation de QoE jusqu'à un seuil au niveau des approches réactives, notre troisième contribution propose une approche d'allocation prédictive. Cette troisième approche se base sur un modèle de prédiction FRDL de la charge future des services de jeux en ligne. Basé sur le réseau LSTM, notre méthode prédit la charge de travail et nous utilisons l'algorithme FRHSA pour une allocation optimale des ressources. Ici, l'approche prédictive d'allocation des ressources montre de meilleurs résultats face à nos précédentes approches.

En somme, nos différentes approches d'allocation de ressources cloud dans le contexte de jeux en ligne, ont montré de meilleurs résultats comparés à d'autres approches de la littérature qui traitent de la même problématique.

Comme perspectives, explorer d'autres contextes d'allocation de services autre que celui du *cloud gaming*. En effet, nos approches peuvent être adaptées pour améliorer les services multimédias comme le streaming des vidéos en ligne.

Aussi, continuer l'évaluation de la QoE perçue par notre approche d'allocation prédictive. En effet, étudier la fiabilité de notre modèle de prédiction avec des traces de jeux réelles. De plus, le processus de prédiction de la charge de travail peut être exécuté avec d'autres classificateurs d'apprentissage profond pour améliorer la performance de la prédiction des jeux en ligne.

En outre, dans un contexte d'évaluation réaliste, essayer de produire des traces réelles via la plateforme open source telle que celle de *Gaming Anywhere* dédiée pour les services *cloud gaming*, sur laquelle seront configurer les scenarios et développer les scripts de tests.

À l'avenir, le mécanisme d'allocation des ressources pourra être réalisé avec d'autres algorithmes d'optimisation en incluant d'autres paramètres.

Une approche basée sur les jeux sérieux (*Serious game*) est envisagée. En effet, avec l'avènement du covid19, l'apprentissage se fait de plus en plus en ligne. Il serait intéressant de tester nos approches dans ce contexte et proposer une solution d'utilisation des jeux sérieux pour un apprentissage plus ludique.

BIBLIOGRAPHIE

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” *Futur. Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009, doi: 10.1016/J.FUTURE.2008.12.001.

[2] L. Wang *et al.*, “Cloud Computing: a Perspective Study,” *New Gener. Comput. 2010 282*, vol. 28, no. 2, pp. 137–146, Jun. 2010, doi: 10.1007/S00354-008-0081-5.

[3] R. J. Grigg and R. Hexel, “Communication versus computation: A survey of cloud gaming approaches,” *18th Int. Conf. Intell. Games Simulation, GAME-ON 2017*, pp. 10–25, 2017.

[4] LiYusen, DengYunhua, TangXueyan, CaiWentong, LiuXiaoguang, and WangGang, “Cost-Efficient Server Provisioning for Cloud Gaming,” *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 14, no. 3s, Jun. 2018, doi: 10.1145/3190838.

[5] Y. Ge, Y. Zhang, Q. Qiu, and Y. H. Lu, “A game theoretic resource allocation for overall energy minimization in mobile cloud computing system,” *Proc. Int. Symp. Low Power Electron. Des.*, pp. 279–284, 2012, doi: 10.1145/2333660.2333724.

[6] W. Cai *et al.*, “A survey on cloud gaming: Future of computer games,” *IEEE Access*, vol. 4, pp. 7605–7620, 2016, doi: 10.1109/ACCESS.2016.2590500.

[7] S. Garfinkel, “Architects of the information society: 35 years of the Laboratory for Computer Science at MIT,” *MIT Press*, 1999.

[8] I. Foster, “The Grid: Blueprint for a New Computing Infrastructure.,” *Morgan Kaufmann Publ. Inc*, 1999.

[9] M. A. Vouk, “Cloud Computing – Issues, Research and Implementations,” *J. Comput. Inf. Technol.*, vol. 16, no. 4, pp. 235–246, Dec. 2008, doi: 10.2498/CIT.1001391.

[10] Ridha, “Qui dominera le cloud public, un marché de 411 milliards de dollars en 2022 selon Forrester,” 2019.

[11] R. Bragg, “Cloud computing: When computers really rule,” *Tech News World, July 2008. Electron. Mag.*, 2009.

[12] B. de Haaff, “Cloud computing-the jargon is back,” *Cloud Comput. Journal, August.*, 2008.

[13] J. Geelan, “Twenty-one experts define cloud computing,” *Cloud Comput. J.*, vol. 4, pp. 1–5, 2009.

[14] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research

challenges," *J. Internet Serv. Appl. 2010 11*, vol. 1, no. 1, pp. 7–18, Apr. 2010, doi: 10.1007/S13174-010-0007-6.

[15] P. Mell and T. Grance, "The NIST-National Institute of Standars and Technology-Definition of Cloud Computing," *NIST Spec. Publ. 800-145*, p. 7, 2011.

[16] E. DHIB HAMDANI, "Allocation des ressources Cloud et impacts sur la qualité d'expérience des services de Cloud Gaming massivement multi-joueurs," Thèse de doctorat, SUPCOM Tunis, 2018.

[17] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," *Proc. - 10th IEEE Int. Conf. High Perform. Comput. Commun. HPCC 2008*, pp. 5–13, 2008, doi: 10.1109/HPCC.2008.172.

[18] G. A. Santana, "CCNA Cloud CLDFND 210-451 Official Cert Guide: Exam 55 Offic Cert Guide," *Cisco Press*, 2016.

[19] Y. Kouki, "Approche dirigée par les contrats de niveaux de service pour la gestion de l'élasticité du" nuage".," 2013.

[20] H. Talebian *et al.*, "Optimizing virtual machine placement in IaaS data centers: taxonomy, review and open issues," *Clust. Comput. 2019 232*, vol. 23, no. 2, pp. 837–878, Jul. 2019, doi: 10.1007/S10586-019-02954-W.

[21] C. Zhong and X. Yuan, "Intelligent elastic scheduling algorithms for paas cloud platform based on load prediction," *Proc. 2019 IEEE 8th Jt. Int. Inf. Technol. Artif. Intell. Conf. ITAIC 2019*, pp. 1500–1503, May 2019, doi: 10.1109/ITAIC.2019.8785600.

[22] G. L. Stavrinides and H. D. Karatza, "Performance evaluation of a SaaS cloud under different levels of workload computational demand variability and tardiness bounds," *Simul. Model. Pract. Theory*, vol. 91, pp. 1–12, Feb. 2019, doi: 10.1016/J.SIMPAT.2018.11.006.

[23] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. L. Wei, and P. Hong, "An Attribute-Based Controlled Collaborative Access Control Scheme for Public Cloud Storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 11, pp. 2927–2942, Nov. 2019, doi: 10.1109/TIFS.2019.2911166.

[24] K. Bessai, "Gestion optimale de l'allocation des ressources pour l'execution des processus dans le cadre du Cloud. Informatique. Université Paris1 Panthéon-Sorbonne, 2014. Français. tel-01275626," 2014.

[25] D. Griebler, A. Vogel, C. A. F. Maron, A. M. Maliszewski, C. Schepke, and L. G.

Fernandes, “Performance of Data Mining, Media, and Financial Applications under Private Cloud Conditions,” *Proc. - IEEE Symp. Comput. Commun.*, vol. 2018-June, pp. 450–456, Nov. 2018, doi: 10.1109/ISCC.2018.8538759.

[26] NIST, “NIST Cloud Computing Reference Architecture: Recommendations of NIST,” *Natl. Inst. Stand. Technol.*, vol. Special Pu, pp. 1–35, 2011, [Online]. Available: https://pmt-eu.hosted.exlibrisgroup.com/permalink/f/gvehrt/TN_cdi_ieee_primary_6012797.

[27] L. Jin, V. Machiraju, and A. Sahai, “Analysis on Service Level Agreement of Web Services Analysis on Service Level Agreement of Web Services,” *HP Tech Rep. 2002*, 2002.

[28] A. Maarouf, A. Marzouk, and A. Haqiq, “Practical modeling of the SLA life cycle in Cloud Computing,” *Int. Conf. Intell. Syst. Des. Appl. ISDA*, vol. 2016-June, pp. 52–58, Jun. 2016, doi: 10.1109/ISDA.2015.7489170.

[29] M. Armbrust *et al.*, “A view of cloud computing,” *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010, doi: 10.1145/1721654.1721672.

[30] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, “Elasticity in Cloud Computing: State of the Art and Research Challenges,” *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 430–447, Mar. 2018, doi: 10.1109/TSC.2017.2711009.

[31] A. Soni and M. Hasan, “Pricing schemes in cloud computing: A review,” *Int. J. Adv. Comput. Res.*, vol. 7, no. 29, pp. 60–70, 2017, doi: 10.19101/IJACR.2017.729001.

[32] S.-H. Chun and B.-S. Choi, “Service models and pricing schemes for cloud computing,” *Clust. Comput. 2013 172*, vol. 17, no. 2, pp. 529–535, Sep. 2013, doi: 10.1007/S10586-013-0296-1.

[33] S. Yin, A. Hameurlain, and F. Morvan, “SLA Definition for Multi-Tenant DBMS and its Impact on Query Optimization,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 11, pp. 2213–2226, Nov. 2018, doi: 10.1109/TKDE.2018.2817235.

[34] N. M. M. K. Chowdhury and R. Boutaba, “A survey of network virtualization,” *Comput. Networks*, vol. 54, no. 5, pp. 862–876, Apr. 2010, doi: 10.1016/J.COMNET.2009.10.017.

[35] S. Ried et al., “Understanding and quantifying the future of cloud computing,” *Tech. Rep.*, 2011.

[36] G. Cook, “Greenpeace International: How Clean is Your Cloud.,” 2012.

[37] R. Shea, J. Liu, E. Ngai, and Y. Cui, “Cloud gaming: Architecture and performance,” *IEEE Netw.*, vol. 27, no. 4, pp. 16–21, 2013, doi: 10.1109/MNET.2013.6574660.

[38] C. Y. Huang, D. Y. Chen, C. H. Hsu, and K. T. Chen, "Gaminganywhere: An open-source cloud gaming testbed," *MM 2013 - Proc. 2013 ACM Multimed. Conf.*, pp. 827–830, 2013, doi: 10.1145/2502081.2502222.

[39] ClaypoolMark and ClaypoolKajal, "Latency and player actions in online games," *Commun. ACM*, vol. 49, no. 11, pp. 40–45, Nov. 2006, doi: 10.1145/1167838.1167860.

[40] T. Cox, "Online and multiplayer gaming — An overview," *Virtual Real. 2000 54*, vol. 5, no. 4, pp. 215–222, 2000, doi: 10.1007/BF01408520.

[41] C.Glenday, "Paperback / GUINNESS WORLD RECORDS," *Random House Inc., coll.*, p. 241, 2009.

[42] T. Samitha, C. R. Prasanth, P. R. Lekshmi, and K. P. Shanti, "Study of H . 264 / Avc Algorithm and It ' s Implementation In Matlab," vol. 4, no. 1, pp. 53–68, 2014.

[43] K. T. Chen, Y. C. Chang, P. H. Tseng, C. Y. Huang, and C. L. Lei, "Measuring the latency of cloud gaming systems," *MM'11 - Proc. 2011 ACM Multimed. Conf. Co-Located Work.*, pp. 1269–1272, 2011, doi: 10.1145/2072298.2071991.

[44] W. Zhang, X. Li, L. Zhao, X. Yang, T. Liu, and W. Yang, "Service Pricing and Selection for IoT Applications Offloading in the Multi-Mobile Edge Computing Systems," *IEEE Access*, vol. 8, pp. 153862–153871, 2020, doi: 10.1109/ACCESS.2020.3018166.

[45] R. Beauregard and P. Corriveau, "User Experience Quality: A Conceptual Framework for Goal Setting and Measurement," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4561 LNCS, pp. 325–332, 2007, doi: 10.1007/978-3-540-73321-8_38.

[46] E. Dhib, K. Boussetta, N. Zangar, and N. Tabbane, "Modeling Cloud gaming experience for Massively Multiplayer Online Games," *2016 13th IEEE Annu. Consum. Commun. Netw. Conf. CCNC 2016*, pp. 381–386, Mar. 2016, doi: 10.1109/CCNC.2016.7444810.

[47] S. B. et C. Ducourtieux, "Sony rachète la société de 'cloud gaming' Gaikai," 2012. https://www.lemonde.fr/technologies/article/2012/07/02/sony-rachete-la-societe-de-cloud-gaming-gaikai_1727799_651865.html (accessed Sep. 06, 2021).

[48] K. T. Chen, Y. C. Chang, H. J. Hsu, D. Y. Chen, C. Y. Huang, and C. H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Trans. Multimed.*, vol. 16, no. 2, pp. 480–495, Feb. 2014, doi: 10.1109/TMM.2013.2291532.

[49] M. Suznjevic, J. Beyer, L. Skorin-Kapov, S. Moller, and N. Sorsa, "Towards understanding the relationship between game type and network traffic for cloud

gaming," *2014 IEEE Int. Conf. Multimed. Expo Work.*, pp. 1–6, 2014.

[50] C. Y. Huang, K. T. Chen, D. Y. Chen, H. J. Hsu, and C. H. Hsu, "GamingAnywhere: The first open source cloud gaming system," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 10, no. 1 SUPPL, 2014, doi: 10.1145/2537855.

[51] K. Lee, D. Chu, E. Cuervo, A. Wolman, and J. Flinn, "Demo: DeLorean: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," *MobiSys 2014 - Proc. 12th Annu. Int. Conf. Mob. Syst. Appl. Serv.*, p. 347, 2014, doi: 10.1145/2594368.2601474.

[52] and Y. L. Z. Xue, D.Wu, J. He, X. Hei, "Playing high-end video games in the cloud: A measurement study.," *IEEE Trans. Circuits Syst. Video Technol.*, 2015.

[53] Y. Lin and H. Shen, "CloudFog: Leveraging Fog to Extend Cloud Gaming for Thin-Client MMOG with High Quality of Service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 431–445, Feb. 2017, doi: 10.1109/TPDS.2016.2563428.

[54] M. Amiri, A. Sobhani, H. Al Osman, and S. Shirmohammadi, "SDN-Enabled Game-Aware Routing for Cloud Gaming Datacenter Network," *IEEE Access*, vol. 5, pp. 18633–18645, Sep. 2017, doi: 10.1109/ACCESS.2017.2752643.

[55] Y. C. Chang, P. H. Tseng, K. T. Chen, and C. L. Lei, "Understanding the performance of thin-client gaming," *2011 IEEE Int. Work. Tech. Comm. Commun. Qual. Reliab. CQR 2011*, 2011, doi: 10.1109/CQR.2011.5996092.

[56] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "Gaming in the clouds: QoE and the users' perspective," *Math. Comput. Model.*, vol. 57, no. 11–12, pp. 2883–2894, Jun. 2013, doi: 10.1016/J.MCM.2011.12.014.

[57] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "An evaluation of QoE in cloud gaming based on subjective tests," *Proc. - 2011 5th Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput. IMIS 2011*, pp. 330–335, 2011, doi: 10.1109/IMIS.2011.92.

[58] S. Möller, D. Pommer, J. Beyer, and J. Rake-Revelant, "Factors Influencing Gaming QoE: Lessons Learned from the Evaluation of Cloud Gaming Services," pp. 163–167, 2013, doi: 10.21437/pqs.2013-31.

[59] Y. T. Lee, K. T. Chen, Han-I Su, and C. L. Lei, "Are all games equally cloud-gaming-friendly? An electromyographic approach," *Annu. Work. Netw. Syst. Support Games*, 2012, doi: 10.1109/NETGAMES.2012.6404025.

[60] P. Quax, A. Beznosyk, W. Vanmontfort, R. Marx, and W. Lamotte, "An evaluation of the impact of game genre on user experience in cloud gaming," *IEEE Consum.*

Electron. Soc. Int. Games Innov. Conf. IGIC, pp. 216–221, 2013, doi: 10.1109/IGIC.2013.6659141.

[61] M. Claypool and D. Finkel, “The effects of latency on player performance in cloud-based games,” *Annu. Work. Netw. Syst. Support Games*, vol. 2015-January, no. January, Jan. 2015, doi: 10.1109/NETGAMES.2014.7008964.

[62] C. Y. Huang, C. H. Hsu, D. Y. Chen, and K. T. Chen, “Quantifying user satisfaction in mobile cloud games,” *Proc. 6th ACM Mob. Video Work. MoVid 2014*, 2014, doi: 10.1145/2579465.2579468.

[63] S. Wang and S. Dey, “Modeling and characterizing user experience in a cloud server based mobile gaming approach,” *GLOBECOM - IEEE Glob. Telecommun. Conf.*, 2009, doi: 10.1109/GLOCOM.2009.5425784.

[64] WangShaoxuan and DeySujit, “Cloud mobile gaming,” *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 16, no. 1, pp. 10–21, Jul. 2012, doi: 10.1145/2331675.2331679.

[65] D. Guo, Y. Han, W. Cai, X. Wang, and V. C. M. Leung, “QoE-Oriented Resource Optimization for Mobile Cloud Gaming: A Potential Game Approach,” *IEEE Int. Conf. Commun.*, vol. 2019-May, May 2019, doi: 10.1109/ICC.2019.8761510.

[66] F. Haouari, E. Baccour, A. Erbad, A. Mohamed, and M. Guizani, “QoE-Aware Resource Allocation for Crowdsourced Live Streaming: A Machine Learning Approach,” *IEEE Int. Conf. Commun.*, vol. 2019-May, Jun. 2019, doi: 10.1109/ICC.2019.8761591.

[67] I. Slivar, L. Skorin-Kapov, and M. Suznjevic, “QoE-Aware Resource Allocation for Multiple Cloud Gaming Users Sharing a Bottleneck Link,” *Proc. 2019 22nd Conf. Innov. Clouds, Internet Networks Work. ICIN 2019*, pp. 118–123, Apr. 2019, doi: 10.1109/ICIN.2019.8685890.

[68] I. Slivar, L. Skorin-Kapov, and M. Suznjevic, “Cloud gaming qoe models for deriving video encoding adaptation strategies,” *Proc. 7th Int. Conf. Multimed. Syst. MMSys 2016*, pp. 185–196, May 2016, doi: 10.1145/2910017.2910602.

[69] Y. Han, D. Guo, W. Cai, X. Wang, and V. Leung, “Virtual Machine Placement Optimization in Mobile Cloud Gaming through QoE-Oriented Resource Competition,” *IEEE Trans. Cloud Comput.*, pp. 1–1, Jun. 2020, doi: 10.1109/TCC.2020.3002023.

[70] J. Hamilton, “Cooperative expendable micro-slice servers (CEMS): Low cost, low power servers for internet-scale services,” *CIDR 2009 - 4th Bienn. Conf. Innov. Data Syst. Res.*, pp. 1–8, 2009.

[71] M. Guazzone, C. Anglano, and M. Sereno, “A game-theoretic approach to coalition formation in green cloud federations,” *Proc. - 14th IEEE/ACM Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2014*, pp. 618–625, 2014, doi: 10.1109/CCGrid.2014.37.

[72] G. S. Aujla, M. Singh, N. Kumar, and A. Y. Zomaya, “Stackelberg Game for Energy-Aware Resource Allocation to Sustain Data Centers Using RES,” *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 1109–1123, Oct. 2019, doi: 10.1109/TCC.2017.2715817.

[73] D. Fernández-Cerero, A. Jakóbik, A. Fernández-Montes, and J. Kołodziej, “GAME-SCORE: Game-based energy-aware cloud scheduler and simulator for computational clouds,” *Simul. Model. Pract. Theory*, vol. 93, pp. 3–20, May 2019, doi: 10.1016/J.SIMPAT.2018.09.001.

[74] A. Vafamehr and M. E. Khodayar, “Energy-aware cloud computing,” *Electr. J.*, vol. 31, no. 2, pp. 40–49, 2018, doi: 10.1016/j.tej.2018.01.009.

[75] H. Guan, J. Yao, Z. Qi, and R. Wang, “Energy-Efficient SLA Guarantees for Virtualized GPU in Cloud Gaming,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 9, pp. 2434–2443, Sep. 2015, doi: 10.1109/TPDS.2014.2350499.

[76] H. J. Hong, D. Y. Chen, C. Y. Huang, K. T. Chen, and C. H. Hsu, “Placing virtual machines to optimize cloud gaming experience,” *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 42–53, Jan. 2015, doi: 10.1109/TCC.2014.2338295.

[77] M. M. Hassan, M. Abdullah-Al-Wadud, A. Almogren, B. Song, and A. Alamri, “Energy-Aware Resource and Revenue Management in Federated Cloud: A Game-Theoretic Approach,” *IEEE Syst. J.*, vol. 11, no. 2, pp. 951–961, Jun. 2017, doi: 10.1109/JSYST.2015.2472973.

[78] F. Zafari, J. Li, K. K. Leung, D. Towsley, and A. Swami, “A game-theoretic approach to multi-objective resource sharing and allocation in mobile edge clouds,” *Proc. Annu. Int. Conf. Mob. Comput. Networking, MOBICOM*, pp. 9–13, Oct. 2018, doi: 10.1145/3266276.3266277.

[79] Y. Liu, S. Dey, and Y. Lu, “Enhancing Video Encoding for Cloud Gaming Using Rendering Information,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1960–1974, Dec. 2015, doi: 10.1109/TCSVT.2015.2450175.

[80] M. S. Aslanpour, M. Ghobaei-Arani, M. Heydari, and N. Mahmoudi, “LARPA: A learning automata-based resource provisioning approach for massively multiplayer online games in cloud environments,” *Int. J. Commun. Syst.*, vol. 32, no. 14, p. e4090, Sep. 2019, doi: 10.1002/DAC.4090.

[81] M. Ghobaei-Arani, R. Khorsand, and M. Ramezanpour, “An autonomous resource

provisioning framework for massively multiplayer online games in cloud environment," *J. Netw. Comput. Appl.*, vol. 142, pp. 76–97, Sep. 2019, doi: 10.1016/J.JNCA.2019.06.002.

[82] A. Bhojan, S. P. Ng, J. Ng, and W. T. Ooi, "CloudyGame: Enabling cloud gaming on the edge with dynamic asset streaming and shared game instances," *Multimed. Tools Appl. 2020 7943*, vol. 79, no. 43, pp. 32503–32523, Aug. 2020, doi: 10.1007/S11042-020-09612-Z.

[83] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect Information Dynamic Stackelberg Game Based Resource Allocation Using Hidden Markov for Cloud Computing," *IEEE Trans. Serv. Comput.*, vol. 11, no. 1, pp. 78–89, Jan. 2018, doi: 10.1109/TSC.2016.2528246.

[84] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach," *Comput. Networks*, vol. 129, pp. 399–409, Dec. 2017, doi: 10.1016/J.COMNET.2017.03.015.

[85] X. Zhu, C. Jiang, L. Kuang, Z. Zhao, and S. Guo, "Two-Layer Game Based Resource Allocation in Cloud Based Integrated Terrestrial-Satellite Networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 509–522, Jun. 2020, doi: 10.1109/TCCN.2020.2981016.

[86] Z. Zhu, J. Peng, K. Liu, and X. Zhang, "A game-based resource pricing and allocation mechanism for profit maximization in cloud computing," *Soft Comput. 2019 246*, vol. 24, no. 6, pp. 4191–4203, Jul. 2019, doi: 10.1007/S00500-019-04183-0.

[87] S. J. Seyed Aboutorabi and M. H. Rezvani, "An Optimized Meta-heuristic Bees Algorithm for Players' Frame Rate Allocation Problem in Cloud Gaming Environments," *Comput. Games J. 2020 93*, vol. 9, no. 3, pp. 281–304, Apr. 2020, doi: 10.1007/S40869-020-00106-4.

[88] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020, doi: 10.1109/TPDS.2019.2938944.

[89] Y. Deng, Y. Li, R. Seet, X. Tang, and W. Cai, "The Server Allocation Problem for Session-Based Multiplayer Cloud Gaming," *IEEE Trans. Multimed.*, vol. 20, no. 5, pp. 1233–1245, May 2018, doi: 10.1109/TMM.2017.2760621.

[90] Y. Li, X. Tang, and W. Cai, "Play Request Dispatching for Efficient Virtual Machine Usage in Cloud Gaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12,

pp. 2052–2063, Dec. 2015, doi: 10.1109/TCSVT.2015.2450152.

[91] M. Basiri and A. Rasoolzadegan, “Delay-Aware Resource Provisioning for Cost-Efficient Cloud Gaming,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 4, pp. 972–983, Apr. 2016, doi: 10.1109/TCSVT.2016.2632121.

[92] F. Chi, X. Wang, W. Cai, and V. C. M. Leung, “Ad-Hoc Cloudlet Based Cooperative Cloud Gaming,” *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 625–639, Nov. 2015, doi: 10.1109/TCC.2015.2498936.

[93] Y. Zhang, L. Jiao, J. Yan, and X. Lin, “Dynamic Service Placement for Virtual Reality Group Gaming on Mobile Edge Cloudlets,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1881–1897, Aug. 2019, doi: 10.1109/JSAC.2019.2927071.

[94] A. Soltanian, D. Naboulsi, R. Glitho, and H. Elbiaze, “Resource Allocation Mechanism for Media Handling Services in Cloud Multimedia Conferencing,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1167–1181, May 2019, doi: 10.1109/JSAC.2019.2906806.

[95] Y. Li *et al.*, “Towards Minimizing Resource Usage with QoS Guarantee in Cloud Gaming,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 426–440, Feb. 2020, doi: 10.1109/TPDS.2020.3024068.

[96] H. E. Dinaki, S. Shirmohammadi, and M. R. Hashemi, “Boosted Metaheuristic Algorithms for QoE-Aware Server Selection in Multiplayer Cloud Gaming,” *IEEE Access*, vol. 8, pp. 60468–60483, 2020, doi: 10.1109/ACCESS.2020.2983080.

[97] G. Peng, H. Wang, J. Dong, and H. Zhang, “Knowledge-Based Resource Allocation for Collaborative Simulation Development in a Multi-Tenant Cloud Computing Environment,” *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 306–317, Mar. 2018, doi: 10.1109/TSC.2016.2518161.

[98] T. Radzik, “Fractional Combinatorial Optimization,” *Handb. Comb. Optim.*, pp. 429–478, 1998, doi: 10.1007/978-1-4613-0303-9_6.

[99] E.-G. Talbi, “A Taxonomy of Hybrid Metaheuristics,” *J. Heuristics 2002 85*, vol. 8, no. 5, pp. 541–564, Sep. 2002, doi: 10.1023/A:1016540724870.

[100] X. Ehrgott, M., & Gandibleux, “Multiobjective Combinatorial Optimization, Multiple-Criteria Optimization: State of the Art Annotated Bibliographic Surveys.,” 2002.

[101] A. Fréville, “The multidimensional 0–1 knapsack problem: An overview,” *Eur. J. Oper. Res.*, vol. 155, no. 1, pp. 1–21, May 2004, doi: 10.1016/S0377-2217(03)00274-1.

[102] D. Binu and B. S. Kariyappa, “RideNN: A New Rider Optimization Algorithm-Based Neural Network for Fault Diagnosis in Analog Circuits,” *IEEE Trans. Instrum. Meas.*,

vol. 68, no. 1, pp. 2–26, Jan. 2018, doi: 10.1109/TIM.2018.2836058.

[103] P. Chakraborty, G. G. Roy, S. Das, D. Jain, and A. Abraham, “An Improved Harmony Search Algorithm with Differential Mutation Operator,” *Fundam. Informaticae*, vol. 95, no. 4, pp. 401–426, 2009, doi: 10.3233/FI-2009-157.

[104] C. Li, Y. P. Wang, H. Tang, and Y. Luo, “Dynamic multi-objective optimized replica placement and migration strategies for SaaS applications in edge cloud,” *Futur. Gener. Comput. Syst.*, vol. 100, pp. 921–937, Nov. 2019, doi: 10.1016/J.FUTURE.2019.05.003.

[105] G. Wang, Y. Yuan, and W. Guo, “An Improved Rider Optimization Algorithm for Solving Engineering Optimization Problems,” *IEEE Access*, vol. 7, pp. 80570–80576, 2019, doi: 10.1109/ACCESS.2019.2923468.

[106] M. Khasanov, S. Kamel, H. M. Hasanien, and A. Al-Durra, “Rider optimization algorithm for optimal DG allocation in radial distribution network,” *2020 2nd Int. Conf. Smart Power Internet Energy Syst. SPIES 2020*, pp. 138–143, Sep. 2020, doi: 10.1109/SPIES48661.2020.9243103.

[107] A. S. Jadhav, P. B. Patil, and S. Biradar, “Optimal feature selection-based diabetic retinopathy detection using improved rider optimization algorithm enabled with deep learning,” *Evol. Intell. 2020*, pp. 1–18, Apr. 2020, doi: 10.1007/S12065-020-00400-0.

[108] R. Buyya, R. Ranjan, and R. N. Calheiros, “Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities,” *Proc. 2009 Int. Conf. High Perform. Comput. Simulation, HPCS 2009*, pp. 1–11, 2009, doi: 10.1109/HPCSIM.2009.5192685.

[109] A. Shehabi, S. Smith, D. Sartor, R. Brown, and M. Herrlin, “United states data center energy usage report,” 2016, Accessed: Sep. 02, 2021. [Online]. Available: https://escholarship.org/content/qt84p772fc/qt84p772fc.pdf.

[110] M. Avgerinou, P. Bertoldi, L. C.- Energies, and U. 2017, “Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency,” *mdpi.com*, 2017, doi: 10.3390/en10101470.

[111] J. Morley, K. Widdicks, M. H.-E. R. & S. Science, and U. 2018, “Digitalisation, energy and data demand: The impact of Internet traffic on overall and peak electricity consumption,” *Elsevier*, 2018, Accessed: Sep. 02, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214629618301051.

[112] C.-T. Yang, S.-T. Chen, J.-C. Liu, Y.-W. Chan, C.-C. Chen, and V. K. Verma, “An energy-efficient cloud system with novel dynamic resource allocation methods,” *J.*

Supercomput. 2019 758, vol. 75, no. 8, pp. 4408–4429, Mar. 2019, doi: 10.1007/S11227-019-02794-W.

[113] P. R. Bhaladhare and D. C. Jinwala, "A clustering approach using fractional calculus-bacterial foraging optimization algorithm for k-anonymization in privacy preserving data mining," *Censorship, Surveillance, Priv. Concepts, Methodol. Tools, Appl.*, vol. 2, pp. 587–608, 2014, doi: 10.4018/978-1-5225-7113-1.ch031.

[114] K. K. Désiré, E. Dhib, N. Tabbane, and O. Asseu, "QoS and QoE aware multi objective resource allocation algorithm for cloud gaming," *J. High Speed Networks*, vol. 27, no. 2, pp. 121–138, Jan. 2021, doi: 10.3233/JHS-210655.

[115] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *J. Netw. Comput. Appl.*, vol. 82, pp. 93–113, Mar. 2017, doi: 10.1016/J.JNCA.2017.01.016.

[116] K. D. Kumar and E. Umamaheswari, "Prediction methods for effective resource provisioning in cloud computing: A survey," *Multiagent Grid Syst.*, vol. 14, no. 3, pp. 283–305, Sep. 2018, doi: 10.3233/MGS-180292.

[117] M. Amiri, L. Mohammad-Khanli, and R. Mirandola, "An online learning model based on episode mining for workload prediction in cloud," *Futur. Gener. Comput. Syst.*, vol. 87, pp. 83–101, Oct. 2018, doi: 10.1016/j.future.2018.04.044.

[118] N. Singh and S. Rao, "Online Ensemble Learning Approach for Server Workload Prediction in Large Datacenters," in *2012 11th International Conference on Machine Learning and Applications*, Dec. 2012, pp. 68–71, doi: 10.1109/ICMLA.2012.213.

[119] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining kohonen maps with arima time series models to forecast traffic flow," *Transp. Res. Part C Emerg. Technol.*, vol. 4, no. 5, pp. 307–318, Oct. 1996, doi: 10.1016/S0968-090X(97)82903-8.

[120] W. Zhang, B. Li, D. Zhao, F. Gong, and Q. Lu, "Workload Prediction for Cloud Cluster Using a Recurrent Neural Network," in *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*, Oct. 2016, pp. 104–109, doi: 10.1109/IIKI.2016.39.

[121] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Dec. 2017, pp. 1–6, doi: 10.1109/ANTS.2017.8384098.

[122] D. Binu and B. S. Kariyappa, "Rider-deep-lstm network for hybrid distance score-based fault prediction in analog circuits," *IEEE Trans. Ind. Electron.*, vol. 68, no. 10,

pp. 10097–10106, 2020, doi: 10.1109/TIE.2020.3028796.

[123] K. K. Désiré, E. Dhib, N. Tabbane, and O. Asseu, “Energy and Quality Aware Multi-Objective Resource Allocation Algorithm in Cloud,” *https://doi.org/10.1142/S0219649221500520*, p. 2150052, Sep. 2021, doi: 10.1142/S0219649221500520.

[124] M. Mach, “L’industrie du gaming ne connaît pas la crise du Covid-19,” 2021. https://www.journaldunet.com/solutions/dsi/1496881-gaming-une-industrie-qui-ne-connait-pas-la-crise-du-covid-19/.

ANNEXES

ANNEXE 1 : ARTICLE 1

Soumission : Septembre 2020 ; **Publication** : Juillet 2021

Journal of High Speed Networks 27 (2021) 121–138
DOI 10.3233/JHS-210655
IOS Press

QoS and QoE aware multi objective resource allocation algorithm for cloud gaming

Koné Kigninman Désiré [a,c,*], Eya Dhib [b], Nabil Tabbane [b] and Olivier Asseu [a,c]
[a] *Ecole Supérieure Africaine des TIC (ESATIC), LASTIC, Abidjan, Côte d'Ivoire*
[b] *University of Carthage, SUP'COM, MEDIATRON Laboratory, Tunis, Tunisia*
[c] *Institut Polytechnique Felix Houphouet-Boigny (INPHB), Yamoussoukro, Côte d'Ivoire*

Abstract. Cloud gaming is an innovative model that congregates video games. The user may have different Quality-of-Experience (QoE), which is a term used to measure a user's level of satisfaction and enjoyment for a particular service. To guarantee general satisfaction for all users with limited cloud resources, it becomes a major issue in the cloud. This paper leverages a game theory in the cloud gaming model with resource optimization to discover optimal solutions to resolve resource allocation. The Rider-based harmony search algorithm (Rider-based HSA), which is the combination of Rider optimization algorithm (ROA) and Harmony search algorithm (HSA), is proposed for resource allocation to improve the cloud computing system's efficiency. The fitness function is newly devised considering certain QoE parameters, which involves fairness index, Quantified experience of players (QE), and Mean Opinion Score (MOS). The proposed Rider-based HSA showed better performance compared to Potential game-based optimization algorithm, Proactive resource allocation algorithm, QoE-aware resource allocation algorithm, Distributed algorithm, and Yusen Li *et al.*, with maximal fairness of 0.999, maximal MOS of 0.873, and maximal QE of 1.

Keywords: Cloud gaming, resource allocation, QoE, cloud computing, fairness

Abstracted/Indexed in

Academic Search

ACM *Computing* Reviews

ACM Digital Library

Applied Science & Technology Source

Business Source Complete

Cambridge Scientific Abstracts

Compendex

Computer Abstracts

Computer Science Index

CSA Illumina

DBLP Bibliography Server

EBSCO Databases

Inspec IET

io-port

Microsoft Academic Search

Science & Technology Collection

SciVerse Scopus

Ulrich's Periodicals Directory

Web of Science: Emerging Sources Citation Index

Web of Science: Journal Citation Reports/Science Edition

Web of Science: Science Citation Index Expanded (SciSearch®)

ANNEXE 2 : ARTICLE 2

Soumission : Février 2021 ; **Publication** : Décembre 2021

Journal of Information & Knowledge Management
Vol. 20, No. 3 (2021) 2150052 (27 pages)
© World Scientific Publishing Co.
DOI: 10.1142/S0219649221500520

World Scientific
www.worldscientific.com

Energy and Quality Aware Multi-Objective Resource Allocation Algorithm in Cloud

Koné Kigninman Désiré*,§, Eya Dhib†, Nabil Tabbane†
and Olivier Asseu‡
*Ecole Supérieure Africaine des TIC (ESATIC)
LASTIC Abidjan, Côte d'Ivoire
†University of Carthage, SUP'COM
MEDIATRON Laboratory, Tunis, Tunisia
‡Institut Polytechnique Felix Houphouet-Boigny (INPHB)
Yamoussoukro, Côte d'Ivoire
§konekigninmand@gmail.com; desire.kone@esatic.edu.ci

Published

Abstract. Cloud gaming has become the new service provisioning prototype that hosts the video games in the cloud and broadcasts the interactive game streaming to the players through the Internet. Here, the cloud must use massive resources for video representation and its streaming when several simultaneous players reach a particular point. Alternatively, various players may have separate necessities on Quality-of Experience, like low delay, high-video quality, etc. The challenging task is providing better service by the fixed cloud resource. Hence, there is a necessity for an energy-aware multi-resource allocation in the cloud. This paper devises a Fractional Rider-Harmony search algorithm (Fractional Rider-HSA) for resource allocation in the cloud. The Fractional Rider-HSA combines fractional calculus, Rider Optimization algorithm (ROA), and HSA. Moreover, the fitness function, like mean opinion score (MOS), gaming experience loss, fairness, energy consumption, and network parameters, is considered to determine the optimal resource allocation. The proposed model produces the maximal MOS of 0.8961, maximal gaming experience loss (QE) of 0.998, maximal fairness of 0.9991, the minimum energy consumption of 0.3109, and minimal delay 0.2266, respectively.

Keywords: Cloud computing; fractional calculus; spark architecture; harmony search algorithm; rider optimization algorithm.

Abstracted & Indexed in

- ABDC Journal Quality List *by Australian Business Deans Council*
- Academic OneFile

- Academic Journal Guide *by Chartered Association of Business Schools*
- Baidu
- CNKI Scholar
- CnpLINKer
- Compendex
- CrossRef
- DBLP Computer Science Bibliography
- Ebsco Discovery Service
- Ebsco Electronic Journal Service (EJS)
- Emerging Sources Citation Index (ESCI)
- ExLibris Primo Central
- Google Scholar
- INSPEC
- io-port.net
- J-Gate
- Naver
- NSTL - National Science and Technology Libraries
- OCLC WorldCat®
- ProQuest Computer & Information Systems Abstracts
- ProQuest SciTech Premium Collection
- ProQuest Library & Information Science Collection
- RePEC
- Scopus
- The Summon® Service

NB : L'article 2 a été accepté en juin 2021 et mis en ligne en octobre 202. La publication est prévue pour décembre 2021 pour que la copie de l'auteur me soit remise. L'article est accessible en ligne via *https://www.worldscientific.com/doi/abs/10.1142/S0219649221500520*

ANNEXE 3 : ARTICLE 3

Soumission : Janvier 2021 ; **Publication** : Mars 2021

Scientific Research Publishing

Engineering, 2021, 13, 135-157
https://www.scirp.org/journal/eng
ISSN Online: 1947-394X
ISSN Print: 1947-3931

Fractional Rider Deep Long Short Term Memory Network for Workload Prediction-Based Distributed Resource Allocation Using Spark in Cloud Gaming

Koné Kigninman Désiré[1,2], Kouassi Adlès Francis[1], Konan Hyacinthe Kouassi[1], Eya Dhib[3], Nabil Tabbane[3], Olivier Asseu[1,2*]

[1]Ecole Supérieure Africaine des TIC, LASTIC, Abidjan, Côte d'Ivoire
[2]Institut National Polytechnique Felix H. Boigny (INPHB), Yamoussoukro, Côte d'Ivoire
[3]MEDIATRON Laboratory, Higher School of Communication of Tunis, Tunis, Tunisia
Email: *oasseu@yahoo.fr

How to cite this paper: Désiré, K.K., Francis, K.A., Kouassi, K.H., Dhib, E., Tabbane, N. and Asseu, O. (2021) Fractional Rider Deep Long Short Term Memory Network for Workload Prediction-Based Distributed Resource Allocation Using Spark in Cloud Gaming. *Engineering*, 13, 135-157.
https://doi.org/10.4236/eng.2021.133011

Received: January 26, 2021
Accepted: March 15, 2021
Published: March 18, 2021

Copyright © 2021 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

The modern development in cloud technologies has turned the idea of cloud gaming into sensible behaviour. The cloud gaming provides an interactive gaming application, which remotely processed in a cloud system, and it streamed the scenes as video series to play through network. Therefore, cloud gaming is a capable approach, which quickly increases the cloud computing platform. Obtaining enhanced user experience in cloud gaming structure is not insignificant task because user anticipates less response delay and high quality videos. To achieve this, cloud providers need to be able to accurately predict irregular player workloads in order to schedule the necessary resources. In this paper, an effective technique, named as Fractional Rider Deep Long Short Term Memory (LSTM) network is developed for workload prediction in cloud gaming. The workload of each resource is computed based on developed Fractional Rider Deep LSTM network. Moreover, resource allocation is performed by fractional Rider-based Harmony Search Algorithm (Rider-based HSA). This Fractional Rider-based HSA is developed by combining Fractional calculus (FC), Rider optimization algorithm (ROA) and Harmony search algorithm (HSA). Moreover, the developed Fractional Rider Deep LSTM is developed by integrating FC and Rider Deep LSTM. In addition, the multi-objective parameters, namely gaming experience loss QE, Mean Opinion Score (MOS), Fairness, energy, network parameters, and predictive load are considered for efficient resource allocation and workload prediction. Additionally, the developed workload prediction model achieved better performance using various parameters, like fairness, MOS, QE, energy and delay.

Hence, the developed Fractional Rider Deep LSTM model showed enhanced results with maximum fairness, MOS, QE of 0.999, 0.921, 0.999 and less energy and delay of 0.322 and 0.456.

Keywords

Cloud Computing, Rider Deep LSTM, Fractional Calculus, Workload Prediction, Resource Allocation

Abstracted & Indexed in

- Academic Journals Database
- AiritiLibrary
- Autoritetslister for serier og forlag
- Base Search
- CALIS
- ChaoXing Periodicals
- Chemical Abstracts Service (CAS)
- Citefactor
- CNKI SCHOLAR
- Cnplinker
- COPAC
- CQVIP
- CrossRef
- DTU Findit
- Elektronische Zeitschriftenbibliothek(EZB)
- GETIT@YALE(Yale University Library)
- Harvard Library E-Journals
- Infotrieve
- i-Scholar
- JournalSeek
- JournalTOCs
- MIAR

- National Science and Technology Library (NSTL)
- National Science Library, Chinese Academy of Sciences (NSLC)
- Open Access Journals Search Engine (OAJSE)
- Open Access Library
- Open J-Gate
- PUBDB DESY Publication Database
- Research Bible
- ResearchGate
- SciFinder
- SciLit
- SHERPA/RoMEO
- Stanford University Libraries
- Système Universitaire de Documentation (Sudoc)
- Technische Informationsbibliothek (TIB)
- The Library of Congress
- Ulrich's Periodicals Directory
- WanFang
- World Journal Clout Index (2020 STM)
- Worldcat
- Zeitschriftendatenbank (ZDB)

ANNEXE 4 : RESULTATS DE PERFORMANCE DE NOS ALGORITHMES

Tableau 2.1. Assessment based on fairness with task size=100

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.140122	0.148862	0.251247	0.296246	0.307288
20	0.145214	0.150584	0.262348	0.279426	0.377425
30	0.144251	0.156045	0.269215	0.293127	0.427923
40	0.155348	0.160467	0.274218	0.309235	0.451812
50	0.157374	0.17727	0.282315	0.328933	0.491942

Tableau 2.2. Assessment based on MOS with task size=100

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.781255	0.793889	0.804218	0.810242	0.824522
20	0.791246	0.800132	0.812436	0.819525	0.824252
30	0.861819	0.870506	0.879521	0.882452	0.895217
40	0.874252	0.883988	0.887422	0.894216	0.904216
50	0.912577	0.921116	0.929453	0.934218	0.939425

Tableau 2.3. Assessment based on QE with task size=100

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.445428	0.454716	0.531565	0.654213	0.724514
20	0.495214	0.51307	0.634216	0.723155	0.752315
30	0.521325	0.551285	0.675432	0.732458	0.812438
40	0.532146	0.556553	0.682433	0.739246	0.845213
50	0.551244	0.577636	0.694651	0.742532	0.899721

Table 2.4. Assessment based on fairness with task size=200

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.394132	0.46114	0.475231	0.482354	0.495347
20	0.421578	0.491633	0.501248	0.517247	0.524275
30	0.465213	0.519616	0.521355	0.532755	0.542379
40	0.496521	0.535196	0.541287	0.554287	0.567235
50	0.495322	0.547464	0.553755	0.567525	0.574219

Tableau 2.5. Assessment based on MOS with task size=200

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.721581	0.749704	0.795424	0.818565	0.822297
20	0.764216	0.790703	0.798451	0.819532	0.827452
30	0.814725	0.831936	0.836426	0.842517	0.849562
40	0.819425	0.839986	0.840856	0.846232	0.849236
50	0.834216	0.848304	0.851375	0.857451	0.867452

Tableau 2.6. Assessment based on QE with task size=200

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.914214	0.977463	0.981659	0.982954	0.985342
20	0.924512	0.980956	0.984512	0.985342	0.989451
30	0.937542	0.989088	0.990123	0.992345	0.994751
40	0.938451	0.990899	0.994513	0.996154	0.997524
50	0.968453	0.979439	0.995421	0.998453	0.999451

Tableau 2.7. Assessment based on fairness with task size=300

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.890534	0.955583	0.964513	0.972486	0.981968
20	0.924158	0.963633	0.968453	0.974128	0.984528
30	0.945219	0.979838	0.984751	0.986413	0.987524
40	0.954287	0.987247	0.989453	0.991745	0.993855
50	0.985274	0.999091	0.999275	0.999545	0.999755

Tableau 2.8. Assessment based on MOS with task size=300

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.794613	0.818955	0.823547	0.834257	0.845273
20	0.808565	0.82646	0.834513	0.845128	0.849523
30	0.814258	0.83567	0.839452	0.846518	0.848422
40	0.824587	0.852371	0.859452	0.861028	0.867451
50	0.845237	0.873178	0.879855	0.884216	0.889535

Tableau 2.9. Assessment based on QE with task size=300

Iteration	RHSA with population size=10	RHSA with population size=20	RHSA with population size=30	RHSA with population size=40	RHSA with population size=50
10	0.895247	0.909888	0.914528	0.925488	0.932155
20	0.904522	0.938271	0.942518	0.952437	0.964571
30	0.914257	0.956619	0.957452	0.964218	0.975481
40	0.945218	0.986619	0.992253	0.995422	0.997458
50	0.954219	0.999724	0.999845	0.999911	0.999929

Tableau 3.1. Assessment based on delay with task size=200

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.212515	0.211126	0.210565	0.209458	0.198547
20	0.201829	0.198228	0.195125	0.193457	0.190649
30	0.212274	0.208655	0.194845	0.184576	0.172459
40	0.238221	0.232172	0.224571	0.215647	0.204579
50	0.239815	0.22663	0.224578	0.214578	0.210965

Tableau 3.2. Assessment based on energy with task size=100

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.09248	0.0706	0.0622	0.059546	0.042458
20	0.17616	0.1444	0.1402	0.12764	0.10146
30	0.20835	0.1933	0.16045	0.142156	0.103136
40	0.2984	0.2964	0.26845	0.215469	0.193513
50	0.351331	0.3109	0.29434	0.29234	0.28234

Table 3.3. Assessment based on fairness with task size=200

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.651779	0.750978	0.760368	0.784521	0.799869
20	0.745261	0.757893	0.760134	0.787094	0.801462
30	0.74698	0.772019	0.786343	0.794527	0.808152
40	0.750796	0.7923	0.806006	0.814529	0.826791
50	0.757774	0.820991	0.821667	0.830671	0.842437

Tableau 3.4. Assessment based on MOS with task size=200

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.761333	0.791017	0.81172	0.827893	0.837909
20	0.758655	0.811059	0.828187	0.836879	0.847892
30	0.776744	0.846876	0.858962	0.873413	0.893527
40	0.781227	0.885999	0.898836	0.900561	0.915783
50	0.783757	0.89607	0.898181	0.915671	0.919024

Tableau 3.5. Assessment based on QE with task size=200

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.978452	0.988629	0.988735	0.988835	0.992598
20	0.978556	0.988558	0.988961	0.988973	0.994621
30	0.978575	0.989441	0.989521	0.989721	0.995129
40	0.983675	0.995459	0.99649	0.997546	0.998458
50	0.983451	0.999706	0.999806	0.999848	0.999925

Tableau 3.6. Assessment based on delay with task size=300

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.469034	0.46847	0.456548	0.441259	0.432578
20	0.479565	0.478488	0.466547	0.452179	0.443257
30	0.458375	0.456413	0.445218	0.438547	0.427234
40	0.460048	0.459685	0.442784	0.431429	0.425437
50	0.463856	0.460127	0.452159	0.443875	0.438126

Tableau 3.7. Assessment based on energy with task size=300

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.0845	0.0755	0.064216	0.051246	0.038548
20	0.154276	0.1244	0.114278	0.113275	0.103524
30	0.23576	0.2074	0.20512	0.195647	0.186547
40	0.292371	0.2689	0.242578	0.234578	0.214578
50	0.403248	0.3545	0.342851	0.324578	0.317549

Tableau 3.8. Assessment based on fairness with task size=300

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.970325	0.971919	0.982458	0.984567	0.989457
20	0.961565	0.96891	0.972459	0.981245	0.984128
30	0.972466	0.984386	0.989453	0.992458	0.994218
40	0.978125	0.988712	0.989458	0.999546	0.999746
50	0.984578	0.999091	0.999246	0.999643	0.999846

Tableau 3.9. Assessment based on MOS with task size=300

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.831711	0.841945	0.852464	0.864522	0.873245
20	0.838565	0.849154	0.856428	0.864313	0.882459
30	0.856488	0.876238	0.896451	0.901273	0.912458
40	0.865647	0.893588	0.899431	0.915349	0.918524
50	0.874581	0.883114	0.901275	0.912457	0.924571

Tableau 3.10. Assessment based on QE with task size=300

Iteration	FRHSA with population size=10	FRHSA with population size=20	FRHSA with population size=30	FRHSA with population size=40	FRHSA with population size=50
10	0.905128	0.913374	0.924538	0.945782	0.954257
20	0.954276	0.968862	0.974581	0.986547	0.988565
30	0.954279	0.960441	0.978541	0.988453	0.988643
40	0.963458	0.988712	0.992146	0.994522	0.995547
50	0.975865	0.999772	0.999846	0.999956	0.999981

Tableau 4.1. Assessment based on delay with task size=200

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.215648	0.20679	0.201426	0.195348	0.154277
20	0.198565	0.180726	0.165428	0.145218	0.124578
30	0.200458	0.198711	0.154688	0.134578	0.114257
40	0.218569	0.209793	0.191346	0.15429	0.135248
50	0.214319	0.193184	0.165248	0.134585	0.114258

Tableau 4.2 Assessment based on energy with task size=200

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.09457	0.0683	0.065874	0.061458	0.059722
20	0.198548	0.1387	0.128458	0.114258	0.111426
30	0.214258	0.1912	0.165875	0.147595	0.124578
40	0.287549	0.2412	0.215479	0.196525	0.165325
50	0.297855	0.2705	0.257355	0.225325	0.205175

Tableau 4.3. Assessment based on fairness with task size=200

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.658754	0.762469	0.812435	0.834573	0.854313
20	0.672432	0.764533	0.819645	0.845734	0.864513
30	0.713466	0.78309	0.834513	0.854313	0.869451
40	0.753461	0.816429	0.854214	0.864537	0.872451
50	0.794615	0.834118	0.864572	0.875431	0.884527

Tableau 4.4. Assessment based on MOS with task size=200

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.802438	0.812559	0.824352	0.834525	0.845613
20	0.812458	0.828666	0.834513	0.845128	0.854614
30	0.834612	0.861843	0.875422	0.885422	0.894615
40	0.865415	0.907062	0.908954	0.910451	0.914249
50	0.894561	0.918627	0.924576	0.929532	0.935422

Tableau 4.5. Assessment based on QE with task size=200

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.924579	0.991916	0.992454	0.994572	0.995784
20	0.935242	0.992349	0.993451	0.995462	0.996524
30	0.952479	0.993508	0.994572	0.996451	0.997525
40	0.965248	0.99742	0.997544	0.997652	0.997786
50	0.975125	0.999729	0.999741	0.999842	0.999861

Tableau 4.6. Assessment based on delay with task size=300

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.478452	0.443126	0.424513	0.412476	0.402452
20	0.479855	0.461507	0.451532	0.446522	0.421785
30	0.449875	0.446898	0.432579	0.424358	0.410274
40	0.465429	0.457859	0.431025	0.420525	0.401246
50	0.469758	0.456391	0.421756	0.412758	0.402316

Tableau 4.7. Assessment based on energy with task size=300

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.108565	0.0715	0.068755	0.057249	0.056742
20	0.119855	0.1143	0.110754	0.110812	0.109855
30	0.201428	0.1952	0.186532	0.175426	0.157265
40	0.284528	0.202	0.198545	0.165244	0.147513
50	0.398543	0.3222	0.287542	0.254318	0.217549

Tableau 4.8. Assessment based on fairness with task size=300

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.865243	0.980729	0.983425	0.987549	0.988543
20	0.874252	0.978819	0.979653	0.981428	0.984258
30	0.895244	0.987534	0.988754	0.988945	0.990242
40	0.912458	0.997271	0.998653	0.998998	0.999245
50	0.924358	0.999538	0.999652	0.999754	0.999825

Table 4.9. Assessment based on MOS with task size=300

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.798565	0.85688	0.86525	0.872454	0.887542
20	0.812458	0.868903	0.88755	0.891255	0.901246
30	0.836452	0.895701	0.907542	0.912548	0.925379
40	0.864578	0.915806	0.924522	0.945218	0.954287
50	0.894257	0.921877	0.932452	0.949759	0.968542

Tableau 4.10. Assessment based on QE with task size=300

Iteration	FRDL with population size = 10	FRDL with population size = 20	FRDL with population size = 30	FRDL with population size = 40	FRDL with population size = 50
10	0.842158	0.946652	0.952134	0.968543	0.971245
20	0.865413	0.977465	0.977981	0.982146	0.987548
30	0.909754	0.979982	0.982155	0.985428	0.988754
40	0.917685	0.99883	0.998854	0.998856	0.998875
50	0.924252	0.999863	0.999875	0.999877	0.999888