République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid Tlemcen
Faculté des Sciences
Département
d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option : Réseaux et Systèmes Distribués (RSD)

Thème

Harris Hawks Optimization Algorithm (HHO) pour l'ordonnancement des tâches dans le Cloud computing

Réalisé par :

- MOHAMMEDI Fatima Zohra

Présenté le 25 Juin 2024 devant le jury composé de :

- Mr BENMOUNA Youcef — Président
- Mr BAMBRIK Ilyas — Examinateur
- Mr BENMAMMAR Badr — Encadrant
- Mr MALTI Arslan Nedhir — Co-Encadrant

Année universitaire: 2023-2024

Remerciements

Tout d'abord, je veux dire merci à Dieu pour m'avoir donné le courage de terminer ce travail.

Ensuite, je suis profondément reconnaissante envers mon encadrant, ***Mr. Badr BENMAMMAR***, ainsi que mon co-encadrant, ***Mr. Arslan Nedhir Malti***, pour leurs précieux conseils, leur disponibilité et leur aide tout au long de la période de ce travail.

je veux exprimer ma gratitude envers les membres du jury, ***Mr Youcef BENMOUNA*** et ***Mr Ilyas BAMBRIK***, qui m'ont honoré et ont pris le temps d'évaluer mon travail.

Pour finir, je veux aussi exprimer ma gratitude envers tous mes professeurs tout au long de mon parcours, ainsi qu'à toutes les personnes qui m'ont aidé à réaliser mon projet de fin d'études.

Dédicace

Je dédie ce modeste travail

A mes très ***chers parents,***

Merci pour votre amour, votre soutien inconditionnel et vos sacrifices.
Vous êtes et vous resterez mes piliers, mes confidents, et ma plus grande source de réconfort.
Vos prières et votre présence ont illuminé mon chemin.
Tout ce que je suis et tout ce que je veux devenir, je le dois à vous
Que le Tout-Puissant Allah vous préserve, vous accorde la santé et vous protège de tout mal.

A mes très ***chères sœurs,***

Merci pour votre soutien inconditionnel et votre encouragement ont été une source de motivation essentielle pour moi.

Table des matières

Liste des figures

Liste des tables

Liste des abréviations

Acronyme	Signification
ACO	Ant Colony Optimization
API	Application Program Interface
FPA	Flower Pollination Algorithm
HHO	Harris Hawks Optimization
IDE	Integrated Development Environment
IHM	Interactions Homme-Machine
MIPS	Million Instructions Per Second
NP	Nondeterministic Polynomial Time
PE	Processing Element
QoS	Quality Of Service
RAM	Random Access Memory
RR	Round Robin
SLA	Service Level Agreement
SP	Service Provider
VM	Virtual Machine

Résumé

Le Cloud computing est une innovation technologique qui assure un accès simple et fiable à des données et des services stockés sur des serveurs distants, garantissant une qualité de service élevée et une disponibilité constante.Ce travail vise à l'optimisation mono-objectif dans le Cloud computing en utilisant la métaheuristique Harris Hawks Optimization (HHO) pour réduire le temps d'exécution. Le problème abordé concerne l'affectation des tâches aux différentes machines virtuelles. Notre approche a montré de meilleures performances comparées à la méthode classique (Round Robin). Nous avons utilisé le simulateur CloudSim pour les simulations, et développé l'interface utilisateur en Java pour faciliter les interactions avec le simulateur.

Mots clés: Cloud computing, CloudSim, HHO, Round Robin.

Abstract

The Cloud computing is a technological innovation that ensures simple and reliable access to data and services stored on remote servers, ensuring high quality of service and constant availability. This work aims at mono-objective optimization in Cloud computing using the Harris Hawks Optimization (HHO) metaheuristic to reduce execution time. The problem addressed concerns the allocation of tasks to different virtual machines. Our approach has shown better performance compared to the classical method (Round Robin). We used the CloudSim simulator for simulations, and developed the user interface in Java to facilitate interactions with the simulator.

Keywords: Cloud computing, CloudSim, HHO, Round Robin.

ملخص

الحوسبة السحابية هي ابتكار تكنولوجي يوفر وصولاً بسيطًا وموثوقا إلى البيانات والخدمات المخزنة على خوادم بعيدة مما يضمن جودة عالية للخدمة وتوافرها باستمرار، ويهدف هذا العمل إلى تحسين أحادي الهدف في الحوسبة السحابية باستخدام métaheuristique HHO لتقليل وقت التنفيذ. تتعلق المشكلة التي تمت معالجتها بتعيين المهام إلى أجهزة افتراضية مختلفة. أظهر نهجنا أداء أفضل مقارنةً بالطريقة التقليدية **(**Round Robin استخدمنا محاكي Cloudsim للمحاكاة , وطورنا واجهة المستخدم بلغة **java** لتسهيل التفاعل مع المحاكي.

الكلمات الرئيسية: الحوسبة السحابية , Round Robin , HHO , Cloudsim

Introduction générale

Le Cloud computing a révolutionné la manière dont les données et les applications sont gérées, accessibles et stockées en remplaçant les infrastructures informatiques locales traditionnelles par des solutions dématérialisées. Le Cloud computing permet aux entreprises de se concentrer pleinement sur leurs activités principales. Cette approche novatrice offre beaucoup de souplesse et de facilité pour évoluer avec ces besoins. Il est devenu un élément clé de la révolution numérique dans de nombreux secteurs d'activité.

L'ordonnancement des tâches est devenu un véritable challenge d'optimisation dans le contexte du Cloud computing. Cela représente un défi essentiel pour assurer une utilisation efficace des ressources car plusieurs utilisateurs peuvent accéder aux mêmes ressources dans cet environnement, entraînant ainsi des conflits de priorités. Pour résoudre cette problématique, il est nécessaire de trouver un équilibre entre les objectifs des utilisateurs et ceux des fournisseurs de services Cloud.

L'utilisation de méta-heuristiques et d'algorithmes sophistiqués est nécessaire pour maximiser les performances du système tout en répondant aux besoins spécifiques des utilisateurs. Cela implique la nécessité d'une utilisation adéquate de ces techniques afin de parvenir à cet équilibre.

Ce travail présente une étude sur l'ordonnancement des tâches dans le Cloud computing avec la métaheuristique Harris Hawks Optimization (HHO), nous avons pris en compte le critère makespan dans le but de mesurer les performances de cet algorithme par rapport à l'heuristique Round Robin, spécifiquement en contexte mono-objectif, en utilisant le Framework CloudSim.

CHAPITRE1 :Cloud computing

1.1 Introduction

Le Cloud computing est une nouvelle technologie de partage de ressources à grande échelle qui simplifie et rend plus efficace le travail des utilisateurs. Il résout les problèmes rencontrés par les entreprises, tels que le stockage des données, l'utilisation d'applications diverses et la disponibilité des informations.

Le but de ce chapitre est d'aborder quelques points essentiels sur le Cloud computing, pour la bonne compréhension de ce concept.

1.2 Historique du Cloud computing

Il est communément admis que le concept de Cloud computing a été initié par le géant Amazon en le cybermarchand avait investi dans un parc informatique afin de pallier les surcharges des serveurs dédiés au commerce en ligne constatées durant les fêtes de fin d'année. A ce moment-là, Internet comptait moins de 600 millions d'utilisateurs mais la fréquentation de la toile et les achats en ligne étaient en pleine augmentation.

En dépit de cette augmentation, les ressources informatiques d'Amazon restaient peu utilisées une fois que les fêtes de fin d'année étaient passées.

Ce dernier a alors eu l'idée de louer ses capacités informatiques le reste de l'année à des clients pour qu'ils stockent les données et qu'ils utilisent les serveurs. Ces services étaient accessibles via Internet et avec une adaptation en temps réel de la capacité de traitement, le tout facturé à la consommation.

Cependant, ce n'est qu'en 2006 que Amazon comprit qu'un nouveau mode de consommation de l'informatique et d'internet faisait son apparition.

Bien avant la naissance du terme de Cloud computing, utilisé par les informaticiens pour qualifier l'immense nébuleuse du net, des services de Cloud étaient déjà utilisés comme le webmail 2, le stockage de données en ligne (photos, vidéos, ...) ou encore le partage d'informations sur les réseaux sociaux [1].

1.3 Notion de base

1.3.1 Définition

Le Cloud computing est une technologie qui utilise Internet et des serveurs distants centraux pour entretenir des données et des applications. Il permet aux consommateurs et aux entreprises d'utiliser des applications sans installation et d'accéder à leurs fichiers personnels sur n'importe quel ordinateur doté d'un accès Internet. Cette technologie permet un calcul beaucoup plus efficace en centralisant le stockage, la mémoire, le traitement et la bande passante [2]**.**

Le Cloud computing se distingue par cinq caractéristiques clés :

- **Libre-service sur demande :** Les systèmes se mettent en place automatiquement, et c'est l'utilisateur qui les configure et les gère à distance via une console de

commande.

- **Accès au réseau étendu :** Ces centres sont souvent connectés directement à Internet pour avoir une très bonne connexion. Les grands fournisseurs les placent partout dans le monde pour que les systèmes soient accessibles.

- **La mutualisation des ressources :** Cette approche implique le partage des ressources par un prestataire de services Cloud entre plusieurs utilisateurs, afin de leur fournir des services adaptés à leurs besoins individuels. Cette démarche multi-utilisateur peut être appliquée à divers services tels que le stockage de données, le traitement et la bande passante.

- **L'élasticité rapide :** L'élasticité rapide dans le cloud computing fait référence à la capacité d'un système cloud à ajuster automatiquement et rapidement les ressources informatiques, telles que le stockage, la puissance de calcul et la mémoire, en fonction de la demande.

- **Un service de mesuré :** Le service de mesuré est une des raisons pour lesquelles le Cloud est super pour les entreprises. Il aide les fournisseurs de Cloud et leurs clients à suivre et à rapporter l'utilisation des services. Cela aide à contrôler les coûts et à utiliser les ressources au mieux.

1.3.2 Avantages et inconvénients

1.3.2.1 Avantage

Le Cloud offre une multitude d'avantages, énumérés ci-dessous [3] :

- **Flexibilité idéale et accès optimisé aux ressources :** Selon l'évolution de ses besoins, l'entreprise peut augmenter ou réduire ses ressources dans un délit réduit sans se soucier de l'obtention des équipements grâce à l'allocation dynamique, ainsi de pouvoir accéder aux multiples logiciels et applications proposées par le fournisseur de services dont elle ne pourrait s'offrir en interne.

- **Optimisation des coûts :** Les services de Cloud permettent à une entreprise de démarrer une activité professionnelle sans avoir à investir dans l'infrastructure IT, qui nécessite également une gestion interne plus coûteuse entre achat de matériels, affectation du personnel, locaux, électricité, maintenance et renouvellement des équipements. Autre point intéressant est la facturation à l'usage, ce que signifie que l'utilisateur paie uniquement ce qu'il consomme.

- **Une diminution des pertes de données accidentelles :** Grace à la réplication de données qui se fait précieusement et en temps réel et qui permet donc aux utilisateurs de récupérer les informations cruciales de leur établissement.

- **Facilité d'utilisation, disponibilité, évolutivité :** Ces qualités sont réalisées grâce aux hautes technologies de pointes utilisées par les fournisseurs de service telles que les Data centers, les supercalculateurs, les connexions internet haut débit et une puissance de calcul que les entreprises ne peuvent pas s'offrir en interne.

1.3.2.2 Inconvénient

Malgré ses nombreux avantages, le Cloud computing peut aussi présenter des inconvénients dans certains cas [4].

- **Temps d'arrêt :** Le Cloud computing étant basés sur l'Internet, il existe toujours un risque de temps d'arrêt et de pannes de service. De telles interruptions des processus commerciaux critiques, surtout si elles se produisent régulièrement, peuvent vous faire perdre du temps et des clients.

- **Sécurité :** Dans le Cloud computing, chaque composant est en ligne. Cela signifie également que chaque composant peut être exposé à un large éventail d'attaques. Le Cloud computing signifie également que les organisations confient partiellement la sécurité et la confidentialité des données aux entreprises qui fournissent leurs serveurs de Cloud computing.

- **Blocage des fournisseurs :** Dans certains cas, la migration des services de Cloud computing d'un fournisseur à un autre peut s'avérer difficile. Les différences entre les plateformes des fournisseurs peuvent compliquer la migration d'une plateforme de Cloud computing vers une autre, ce qui entraîne souvent des coûts supplémentaires et des complexités de configuration.

1.3.3 Virtualisation

La virtualisation est une technologie que vous pouvez utiliser pour créer des représentations virtuelles de serveurs, de stockage, de réseaux et d'autres machines physiques. Le logiciel virtuel imite les fonctions du matériel physique pour exécuter plusieurs machines virtuelles sur une seule machine physique.

Les entreprises ont recours à la virtualisation pour utiliser efficacement leurs ressources matérielles et obtenir un meilleur rendement de leurs investissements. Elle alimente également les services de Cloud computing qui aident les organisations à gérer leur infrastructure plus efficacement [5].

La virtualisation offre de nombreux avantages. Par exemple :

- Elle permet d'utiliser au mieux les ressources d'un groupe de machines en les répartissant selon les besoins.

- Elle permet de faire des économies sur le matériel, comme l'électricité, la maintenance et la surveillance.

- On peut installer, tester et développer des logiciels sans risquer d'endommager le système principal.

1.3.4 Data Center

Un data center, également connu sous le nom de centre de données, est une infrastructure qui abrite un grand nombre d'ordinateurs et d'équipements liés au stockage, à la gestion et au traitement des données.

Les data centers sont utilisés par les organisations pour héberger des applications, des services et des données critiques qui assurent leur bon fonctionnement [6].

1.3.5 Service provider

Le fournisseur de services (service provider ou SP) est une entreprise qui fournit des services au client : du matérielle virtuelle, des logicielle et d'autre service loués et gérés par le fournisseur, basés sur un accord de niveau de service (Service Level Agreement ou SLA) et une négociation entre le fournisseur de services et le consommateur [7]**.**

1.3.6 Service Level Agreement

Le SLA est un accord de niveau de service qui est un contrat entre un fournisseur de services Cloud et son client. Il spécifie les niveaux de service que le prestataire de services doit fournir, tels que la disponibilité, la qualité, la sécurité, les performances et la maintenance, ainsi que les responsabilités du client.

Le SLA garantit que les services sont fournis en temps opportun et avec les niveaux requis de qualité et de sécurité. En outre, il établit les attentes du client en ce qui concerne les niveaux de service ainsi que les responsabilités du prestataire de services en cas de violation des conditions du SLA [2].

1.3.7 Consolidation

La consolidation survient généralement pendant les fusions et les acquisitions d'entreprises. Lorsque l'entreprise majoritaire n'a pas besoin des centres de données utilisées par les firmes qu'elles rachètent, donc l'entreprise avec plusieurs Datacenter peut choisir de les consolider, en réduisant leur nombre pour minimiser les coûts des opérations [7].

1.3.8 Machine virtuelle

Une machine virtuelle est un ordinateur défini par logiciel qui s'exécute sur un ordinateur physique doté d'un système d'exploitation et de ressources informatiques distincts.

L'ordinateur physique est appelé machine hôte et les machines virtuelles sont des machines invitées. Plusieurs machines virtuelles peuvent être exécutées sur une seule machine physique [5].

1.3.9 Broker

Le Broker est l'intermédiaire entre les VMs et les Cloudlets (les tâches). Il est le responsable de la médiation des négociations entre l'utilisateur et les prestataires de service selon les conditions de qualité de service (QoS) des utilisateurs [7].

1.4 Types de Cloud computing

Les modèles de Cloud définissent les moyens par lesquels les utilisateurs peuvent accéder aux ressources du Cloud et les utiliser. On peut distinguer trois types principaux.

1.4.1 Cloud privé

Un Cloud privé est un système réservé à une seule organisation. Il permet de stocker et de traiter des données et des applications, mais seuls les utilisateurs autorisés au sein de cette organisation y ont accès, garantissant ainsi un niveau supérieur de contrôle et de sécurité.

1.4.2 Cloud public

Un Cloud public repose sur le modèle standard de Cloud computing dans lequel le fournisseur de services rend des ressources accessibles à tout utilisateur qui a une connexion Internet. Il offre à ses utilisateurs l'évolutivité, la flexibilité et aussi des options supplémentaires de contrôle. Les services de Cloud public peuvent être gratuits ou facturés à l'utilisation [7].

1.4.3 Cloud hybride

Il s'agit d'un environnement combinant deux Cloud ou plus (Privé et Publique). Lorsque l'utilisation des ressources du Cloud privé atteint le maximum, l'entreprise peut bénéficier d'autres ressources supplémentaires appartenant au Cloud public. Une autre propriété caractérise ce type de Cloud, c'est la répartition de la puissance de calcul ou traitement entre des Cloud s publiques et privés ceci permet d'avoir des résultats dans un temps très rapide [3].

1.5 Couches de Cloud computing

L'architecture du Cloud computing est un modèle qui décrit la structure et l'organisation des ressources informatiques disponibles via le Cloud. Elle comprend 3 clés, notamment IaaS, PaaS et SaaS.

La figure 1.1 ci-dessous illustre les différentes couches du Cloud, de la moins visible pour l'utilisateur final à la plus visible, organisées en forme de pyramide.

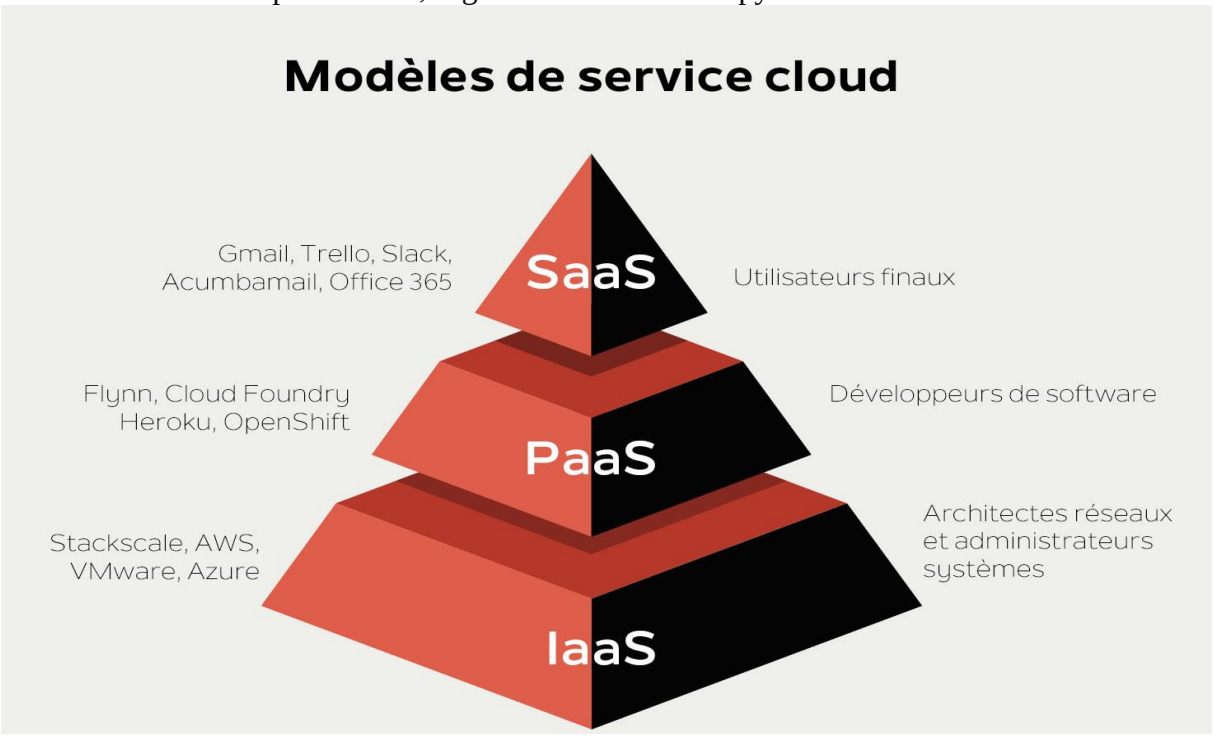


Figure 1-1: Les trois couches du Cloud computing [9].

1.5.1 Infrastructure as a Service

Infrastructure as a Service (IaaS) est un modèle de service de Cloud computing qui permet aux utilisateurs de louer des ressources de traitement telles que des serveurs, des réseaux et du stockage à la demande. Les fournisseurs de services Cloud mettent à disposition un système virtualisé via Internet.

Les utilisateurs peuvent déployer et gérer leurs propres applications et systèmes d'exploitation sur ces ressources louées. Les applications sont accessibles à partir de divers périphériques clients via une interface de client léger telle qu'un navigateur Web (par exemple, messagerie électronique basée sur le Web) [2].

1.5.2 Platform as a Service

Dans le modèle PaaS, le fournisseur de services Cloud fournit une plate-forme qui permet aux développeurs de créer, de personnaliser et de déployer des applications basées sur le Cloud. Cette plate-forme fournit aux développeurs un environnement de développement complet, qui comprend un système d'exploitation, des bibliothèques de développement, des serveurs d'applications, des bases de données et d'autres outils nécessaires pour créer des applications [2].

1.5.3 Software as a Service

Software as a Service (Saas) est un modèle logiciel basé sur le Cloud qui fournit des applications aux utilisateurs finaux via un navigateur Internet. Les fournisseurs de SaaS hébergent des services et des applications auxquels les clients peuvent accéder à la demande [8].

1.6 Conclusion

Dans ce chapitre, nous avons abordé les bases du Cloud computing, qui est le sujet principal de notre projet de fin d'études. Nous avons examiné différents aspects de ce domaine, notamment son architecture, ses diverses variantes, ainsi que ses avantages et ses inconvénients. De plus, nous avons clarifié plusieurs concepts clés tels que les machines virtuelles, les SLA (Service Level Agreements), les centres de données, la virtualisation, et bien d'autres encore.

Le deuxième chapitre de notre PFE sera consacré à la présentation des métaheuristiques.

CHAPITRE 2 : Les métaheuristiques

2.1 Introduction

Les méthodes traditionnelles ne peuvent pas toujours trouver la meilleure solution, c'est parce que certains problèmes sont très compliqués et il n'existe pas de méthodes rapides pour les résoudre.

Au lieu de cela, les chercheurs ont essayé différentes astuces pour s'approcher de la meilleure solution. Ensuite, ils ont commencé à utiliser des méthodes plus générales appelées "métaheuristiques". Ces méthodes sont devenues très importantes pour trouver de bonnes solutions aux problèmes difficiles.

Dans ce chapitre, nous allons découvrir les différentes façons de trouver les meilleures solutions à des problèmes. Ensuite, nous allons nous pencher sur une méthode particulière appelée Harris Hawks Optimization.

2.2 Les méthodes de résolution

Pour trouver une solution à un problème d'optimisation, on a principalement deux façons de le faire : soit avec des méthodes qui trouvent la solution exacte, soit avec des méthodes qui trouvent une solution approximative.

2.2.1 Les méthodes exactes

Ces méthodes garantissent de trouver la meilleure solution possible pour chaque cas d'un problème de combinaison. Elles examinent toutes les possibilités pour trouver la solution parfaite. Cependant, plus la taille de problème est grande, plus cela prend de temps pour trouver la solution idéale. Complexité temporelle et spéciale augmentent avec la taille du problème. En pratique, ces méthodes conviennent mieux aux petits problèmes ou à ceux qui ne sont pas très compliqués.

2.2.2 Les méthodes approchées

Parfois, les méthodes exactes ne conviennent pas pour résoudre des problèmes réels. Ceci dépend de la taille du problème. Dans ces cas-là, trouver la solution parfaite peut être très difficile, voire impossible. C'est pourquoi on utilise d'autres méthodes qui permettent de trouver des solutions satisfaisantes dans un temps raisonnable.

Ces solutions ne sont pas toujours les meilleurs possibles, bien qu'elles soient garanties par rapport à la meilleure solution. Les méthodes approchées sont utilisées pour cela, même si elles ne promettent pas de trouver la solution optimale. Elles aident néanmoins à atteindre l'objectif rapidement.

La figure 2.1 donne un aperçu global sur les méthodes de résolution.

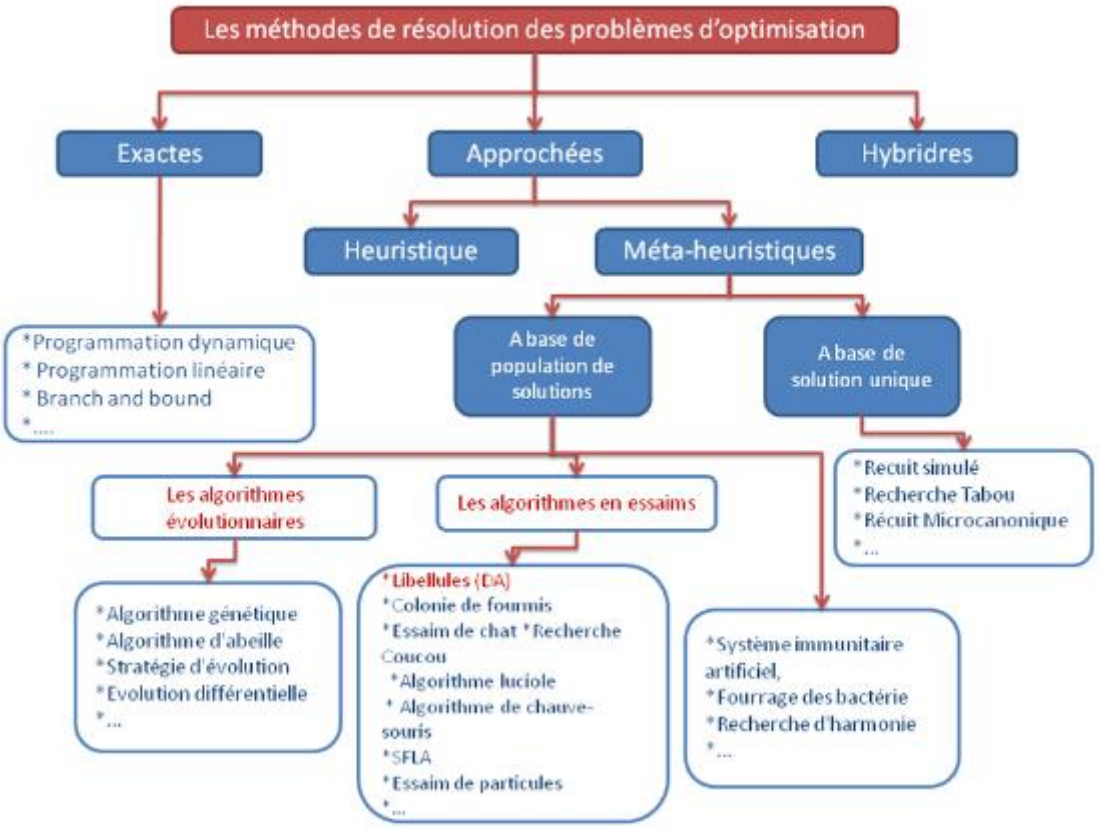


Figure 2-1: Classes des méthodes de résolutions [10].

2.2.2.1 Heuristique

Une heuristique est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation NP-difficile.

Une heuristique ou méthode approximative, est donc le contraire d'un algorithme exact qui trouve une solution optimale pour un problème donné. Les algorithmes de résolution exacts étant de complexité exponentielle, il est généralement plus judicieux de faire appel à des méthodes heuristiques pour des problèmes difficiles.

2.2.2.2 Métaheuristique

Une méta-heuristique est un algorithme plus avancé qu'une heuristique, capable de résoudre différents types de problèmes NP-complets. Il cherche à trouver une solution de haute qualité pour de nombreux problèmes d'optimisation dans un laps de temps raisonnable.

Plusieurs classifications des métaheuristiques ont été proposées, la plupart distinguent globalement deux catégories en fonction du nombre de solutions qu'elles manipulent : les méthodes à base de solution unique, qui travaillent sur un seul point de l'espace de recherche à un instant donné, appelées aussi méthodes à base de voisinage, et les méthodes à base de population, qui travaillent sur un ensemble de points de l'espace de recherche en parallèle [7].

- **Métaheuristiques à base de solution unique :** les métaheuristiques à base de solution unique, aussi appelées méthodes de trajectoire. Contrairement aux métaheuristiques à base de population, les métaheuristiques à solution unique commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, la méthode du recuit simulé, la recherche taboue, la méthode GRASP, la recherche à voisinage variable, la recherche locale itérée, et leurs variantes [12].
- **Métaheuristiques à base de population de solutions :** les métaheuristiques à population de solution appelée aussi les méthodes évolutionnaires, consiste à travailler sur une population de solutions et non pas sur une solution unique. Le principe de ces méthodes c'est de combiner ensemble des solutions entre elles pour en former de nouvelles en essayant d'hériter des bonnes caractéristiques des solutions parentes. Ce processus est répété jusqu'à arriver à un critère d'arrêt. On peut citer comme exemple les algorithmes génétiques, les algorithmes de fourmis, les algorithmes de d'optimisation par essaim [13].

2.3 Quelques algorithmes basés sur l'intelligence en essaim

Les algorithmes d'intelligence en essaim sont des techniques de résolution de problèmes inspirées par le comportement collectif d'organismes sociaux.

Dans ce qui suit, nous allons vous montrer quelques méthodes basées sur le comportement des groupes comme : l'algorithme de colonies de fourmis (ant colony optimization - ACO), l'algorithme de pollinisation des fleurs (Flower Pollination Algorithm ou FPA) et l'algorithme d'optimisation des faucons de Harris (Harris Hawks Optimization Algorithm ou HHO).

Nous allons nous concentrer davantage sur celui-ci HHO, car il sera utilisé dans notre projet de fin d'études.

2.3.1 Algorithme de colonies de fourmis

Les algorithmes de colonies de fourmis (ant colony optimization - ACO) sont des algorithmes proposés par Marco Dorigo dans les années 1990 pour la recherche de chemins optimaux dans un graphe [14].

Cette métaheuristique imite le comportement de fourmis cherchant de la nourriture. A chaque fois qu'une fourmi se déplace, elle laisse sur la trace de son passage une odeur (la phéromone). Comme la fourmi est rarement une exploratrice isolée, avec plusieurs de ses congénères, elle explore une région en quête de nourriture. Face à un obstacle, le groupe des fourmis explore les deux côtés de l'obstacle et se retrouvent, puis elles reviennent au nid avec de la nourriture. Les autres fourmis qui veulent obtenir de la nourriture elles aussi vont emprunter le même chemin. Si celui-ci se sépare face à l'obstacle, les fourmis vont alors emprunter préférablement le chemin sur lequel la phéromone sera la plus forte. Mais la phéromone étant une odeur elle s'évapore. Si peu de fourmis empruntent une trace, il est possible que ce chemin ne soit plus valable au bout d'un moment, il en est de même si des fourmis exploratrices empruntent un chemin plus long (pour le contournement de l'obstacle par exemple). Par contre, si le chemin est fortement emprunté, chaque nouvelle fourmi qui passe

redépose un peu de phéromone et renforce ainsi la trace, donnant alors à ce chemin une plus grande probabilité d'être emprunté [15].

2.3.2 Algorithme de pollinisation des fleurs

À l'origine, Xin-She Yang en 2012 a proposé un algorithme de pollinisation des fleurs bio-inspiré naturel tirant sa métaphore du processus de pollinisation des plantes à fleurs. Récemment, l'algorithme de pollinisation des fleurs (Flower Pollination Algorithm ou FPA) acquiert une grande prise de conscience en raison de son applicabilité efficace aux problèmes d'optimisation dans divers domaines de la vie réelle.

Du point de vue de l'évolution biologique, l'objectif principal de la pollinisation des fleurs est la reproduction optimale des plantes en survivant des fleurs les plus aptes dans les plantes à fleurs.

Généralement, la pollinisation des fleurs est associée au transfert d'une substance chimique appelée pollen, et ce transfert est souvent lié à certaines créatures appelées pollinisateurs ou parfois vecteurs polliniques, qui peuvent être de divers types comme les insectes, les oiseaux, les chauves-souris et d'autres animaux. Lorsqu'un insecte visite une fleur, les grains de pollen collent à son corps. Si l'insecte visite une autre fleur, le pollen est transféré vers la stigmatisation de la fleur visitée, ce qui facilite la pollinisation [7].

2.3.3 Algorithme de Harris Hawks

2.3.3.1 Inspiration de HHO

En 1997, l'étude de Louis Lefebvre a révélé que la Buse de Harris est l'oiseau le plus intelligent que l'on trouve dans le sud de l'Arizona, aux États-Unis.

Le comportement de recherche de nourriture de la Buse de Harris varie considérablement de celui des autres oiseaux, car la Buse de Harris continue de se nourrir avec d'autres membres de la même famille de la même espèce.

Les Harris Hawks utilisent une technique appelée « saut surprise », connue sous le nom d'approche des « sept meurtres » pour tendre une embuscade à la proie. Au cours de cette attaque, quelques autres faucons tendaient une embuscade dans un certain nombre de directions et convergeaient vers le lapin cible, et l'attaque était terminée en quelques secondes. Les faucons de Harris utilisent différents styles de chasse en fonction du comportement d'évasion de la proie et du changement dynamique des instances. Par exemple, les faucons utilisent des tactiques de changement lorsque le faucon chef plonge rapidement pour attaquer la proie, et que la proie essaie d'échapper au faucon chef, puis qu'un autre faucon de l'équipe du groupe poursuivra immédiatement la poursuite. Ces tactiques de changement confondent la proie ciblée et cherchent à épuiser la proie détectée et à augmenter son danger. Enfin, les proies fatiguées ne peuvent pas échapper à l'équipe du faucon, car l'un des puissants faucons massacre la proie fatiguée et la partage avec les membres du groupe [16].

Figure 2-2: Les faucons de Harris [17].

2.3.3.2 Description de l'algorithme HHO

2.3.3.2.1 Phase d'exploration

Les faucons de Harris ont des yeux puissants qui peuvent surveiller et identifier les proies, mais parfois les proies ne sont pas visibles. Pendant cette condition, les faucons ont attendu de longues heures et surveillé pour identifier la proie. Dans HHO, les faucons sont considérés comme des solutions candidates, et à chaque itération, la proie est considérée comme la solution optimale.

Les faucons se perchent à des endroits précis et surveillent constamment l'environnement environnant pour identifier leurs proies à l'aide de deux stratégies, qui sont représentées dans l'équation **(1)**. Si **q < 0.5**, les faucons se perchent en fonction de la position des membres de la famille. Si **q ≥ 0,5**, les faucons se perchent dans un espace aléatoire à l'intérieur de la zone de population.

$$\mathbf{X}(t+1) = \begin{cases} X_{rand}(t) - r_1 |X_{rand}(t) - 2r_2 X(t)| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases} \quad \textbf{(1)}$$

Où ***X(t + 1)*** est le vecteur de position des faucons à la prochaine itération ***t+1,*** $X_{rabbit}(t)$ est la position du lapin, ***X(t)*** est le vecteur de position actuelle des faucons, ***r1, r2, r3, r4*** et ***q*** sont des nombres aléatoires dans l'intervalle [0, 1] qui sont mis à jour à chaque itération, ***LB*** et ***UB*** montrent les bornes supérieure et inférieure des variables des variables de décision du problème d'optimisation , $X_{rabbit}(t)$ est un faucon sélectionné au hasard dans la population actuelle, et $X_m(t)$ est la position moyenne du la population courant de faucons. La position moyenne des faucons est atteinte à l'aide de l'équation **(2).**

$$X_m(t) = \frac{1}{N} \Sigma_{i=0}^{N} X_i(t) \quad \textbf{(2)}$$

Où $X_i(t)$ indique l'emplacement de chaque faucon dans l'itération ***t*** et ***N*** désigne le nombre total de faucons.

2.3.3.2.2 Transition de l'exploration à l'exploitation

La phase d'exploration à la phase d'exploitation, en fonction du niveau d'énergie de la proie à s'échapper, qui est modélisée comme

$$E = 2E_0\left(1 - {}^{t}/_{T}\right) \qquad (3)$$

E indique l'énergie de la proie, T est le nombre maximal d'itérations et E_0 est l'état initial de son énergie. E_0 changera la valeur de [−1, 1]. Lorsque E_0 descend de 0 à −1, la proie est épuisée, de même, lorsque E_0 La valeur augmente de 0 à 1, la proie est renforcée[16].

2.3.3.2.3 Phase d'exploitation

Cette phase décrit le moment où les faucons ciblent et attaquent leur proie, tandis que cette dernière tente de s'échapper de l'attaque. En fonction des comportements d'évasion des proies et des stratégies de poursuite des faucons de Harris, quatre stratégies possibles sont proposées dans le HHO pour modéliser le stade d'attaque.

- **Siège en douceur :** le siège en douceur se produit quand **r ≥ 0.5 et |E| ≥ 0.5**, et quand la proie a assez d'énergie pour s'échapper, mais les faucons l'encerclent et lui prennent son énergie. Ce comportement est modélisé par les équations suivantes :

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)| \qquad (4)$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \qquad (5)$$

Où $\Delta X(t)$ est la différence entre le vecteur de position du lapin et l'emplacement actuel à l'itération t, r_5 est un nombre aléatoire à l'intérieur de [0, 1] et $J = 2(1-r_5)$ représente la force de saut aléatoire du lapin tout au long de la procédure d'évacuation. La valeur J change aléatoirement à chaque itération pour simuler la nature des mouvements du lapin.

- **Assiéger durement** :se produit quand $r \geq 0.5$ et $|E| < 0.5$, la proie est fatiguée et n'a pas assez d'énergie pour s'échapper. Les faucons entourent la proie et réalisent le saut surpris. Les positions actuelles des faucons sont mises à jour à l'aide de l'équation suivante :

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \qquad (6)$$

- **Siège en douceur avec des plongées rapides progressives** : Cette sous-section traite du siège en douceur avec des plongeons rapides progressifs où la proie a suffisamment d'énergie $|E| \geq 0,5$ pour échapper à l'attaque, mais le faucon construit un siège en douceur $r < 0,5$. Dans cette étape, le faucon doit penser intelligemment et choisir la meilleure position pour cibler la proie [16].

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X_{(t)}| \qquad (7)$$

Un plongeon est sélectionné sur la base d'un vol de levy (LF) à l'aide de l'équation suivante :

$$Z = Y + S \times (D) \qquad (8)$$

Où D est la dimension du problème et S est un vecteur aléatoire de taille $1 \times D$ et LF est levol de prélèvement fonction, qui est calculée à l'aide de l'équation suivante :

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \ , \sigma = \left(\frac{\Gamma(1+\beta) \times sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \qquad (9)$$

Où u, sont des valeurs aléatoires à l'intérieur de**[0,1]**, **β** est une constante par défaut fixée à ***1,5***. Par conséquent, la stratégie finale pour mettre à jour les positions des faucons dans la phase de siège doux peut être exécutée par l'équation suivante :

$$X(t+1) = \begin{cases} Y \text{ si } F(Y) < F(X(t)) \\ Z \text{ si } F(Z) < F(X(t)) \end{cases} \qquad (10)$$

Y et Z sont obtenus en utilisant les équations **(7)** et **(8)**.

- **Assiégée avec des plongées rapides progressives :** Cette sous-section traite du siège dur avec des plongées rapides progressives lorsque $|E| < 0,5$ et $r < 0,5$ où la proie n'a pas assez d'énergie pour échapper à l'attaque et Hawk construit un siège dur pour attraper et tuer la proie. Dans cette phase, l'état de la proie est similaire à celui du siège doux, mais les faucons ont l'intention de minimiser la distance entre leurs emplacements pour échapper à la proie. L'équation explique la dure situation d'assiégé [16].

$$X(t+1) = \begin{cases} Y \text{ si } F(Y) < F(X(t)) \\ Z \text{ si } F(Z) < F(X(t)) \end{cases} \qquad (11)$$

Où Y et Z sont obtenus en utilisant de nouvelles règles dans les équations suivantes :

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X_{m(t)}| \qquad (12)$$

$$Z = Y + S \times (D) \qquad (13)$$

Où $X_{(t)}$ est obtenu en utilisant équation **(2).**

2.3.3.3 Pseudo codes de l'algorithme d'optimisation Harris Hawks

Entrées : La taille de la population N et le nombre maximal d'itérations T.

Sorties : L'emplacement du lapin et sa valeur de fitnesse

Début :

Initialiser la population aléatoire X (i = 1, 2, …, N).

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

Calculer les valeurs de fitness des faucons.

Définir $\boldsymbol{Xrabbit}$ comme emplacement du lapin (meilleur emplacement).

Pour (chaque faucon($\boldsymbol{Xi}$)) **faire**

Mettre à jour l'énergie initiale ***E0*** et
la force de saut **J** ►*E0*=2rand ()-1, J=2(1-rand ())

Mettez à jour le *E* en utilisant Eq **(3).**

Si ($|E| \geq 1$) **alors** ►Phase d'exploration

Mettre à jour le vecteur de localisation à l'aide d'Eq **(1).**

Fin Si

Si ($|E| < 1$) **alors** ►Phase d'exploitation

Si ($r \geq 0{,}5$ $\boldsymbol{et}$ $|E| \geq 0{,}5$) **alors** ► Siège en douceur

Mettre à jour le vecteur de localisation à l'aide d'Eq **(4).**

Sinon si ($r \geq 0{,}5$ $\boldsymbol{et}$ $|E| < 0{,}5$) **alors** ►Assiéger durement

Mettre à jour le vecteur de localisation à l'aide d'Eq **(6).**

Sinon si ($r < 0{,}5$ $\boldsymbol{et}$ $|E| \geq 0{,}5$) **alors** ► Siège en douceur avec des plongées rapides progressives

Mettre à jour le vecteur de localisation à l'aide d'Eq **(10).**

Sinon si ($r < 0{,}5$ $\boldsymbol{et}$ $|E| < 0{,}5$) **alors** ► Assiégée avec des plongées rapides progressives

Mettre à jour le vecteur de localisation à l'aide d'Eq **(11).**

Fin Si

Fin Si

Fin Pour

Fin tant que

Retour $\boldsymbol{Xrabbit}$

FIN

2.4 Conclusion

Dans ce chapitre, nous avons expliqué comment résoudre des problèmes en utilisant des méthodes exactes et approximatives, en mettant l'accent sur les métaheuristiques. Nous avons particulièrement détaillé la métaheuristique HHO (Harris Hawks Optimization) en présentant ses caractéristiques et son pseudo-code.

Dans le chapitre suivant, nous intéressons, à l'application de l'algorithme HHO pour l'ordonnancement de tâches mono-objectif dans le domaine du Cloud computing.

CHAPITRE 3 :
Implémentation de l’application et évaluation des résultats obtenus

3.1 Introduction

L'ordonnancement est généralement un problème très compliqué, ce qui rend impossible de trouver une solution optimale avec une recherche exhaustive. Par conséquent, ce défi est important et nécessite une façon plus efficace d'évaluer différents objectifs rapidement. C'est pourquoi on utilise les métaheuristiques.

Dans ce chapitre, nous expliquons ce que nous avons fait pour ce projet de fin d'études. Nous avons employé la méthode HHO pour l'optimisation mono-objectifs dans le Cloud computing.

Nous débutons par présenter les outils de simulation utilisés dans le cadre de ce projet, notamment le langage Java, l'IDE NetBeans et le simulateur CloudSim. Nous examinons également l'architecture de ce simulateur, ses composants principaux ainsi que les classes que nous avons intégrées pour notre travail.

Par la suite, nous montrons l'interface utilisateur que nous avons créé pour ce projet et les résultats des simulations.

3.2 Environnement de développement et de simulation

3.2.1 Java

Java est un langage de programmation de haut niveau orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld [18].

3.2.2 Netbeans

NetBeans est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML [19].

3.2.3 JFreeChart

JFreeChart est une API Java open source sous licence LGPL (Lesser General Public Licence) qui permet aux développeurs de visualiser facilement des graphiques de qualité professionnelle dans leurs applications. Elle propose de nombreuses options de configuration pour personnaliser le rendu des graphes.

- Composants Swing.
- Images (PNG ou JPEG).
- Images vectorielles PDF, EPS et SVG [7].

3.2.4 CloudSim

CloudSim est un Framework open-source, qui est utilisé pour simuler l'infrastructure et les services de cloud computing. Il est développé par l'organisation CLOUDS Lab et est entièrement écrit en Java. Il est utilisé pour modéliser et simuler un environnement de cloud computing comme moyen d'évaluer une hypothèse avant le développement d'un logiciel afin de reproduire des tests et des résultats [20].

CloudSim dispose de deux stratégies d'ordonnancement :

- **Space shared :** l'ordonnanceur affecte une tâche à une machine virtuelle à un moment donné. Une fois cette tâche achevée, une autre tâche est ensuite exécutée sur cette même machine virtuelle.

- **Time shared :** l'ordonnanceur attribue simultanément toutes les tâches à une machine virtuelle. Le temps est réparti entre ces tâches, permettant leur exécution concurrente sur la machine virtuelle.

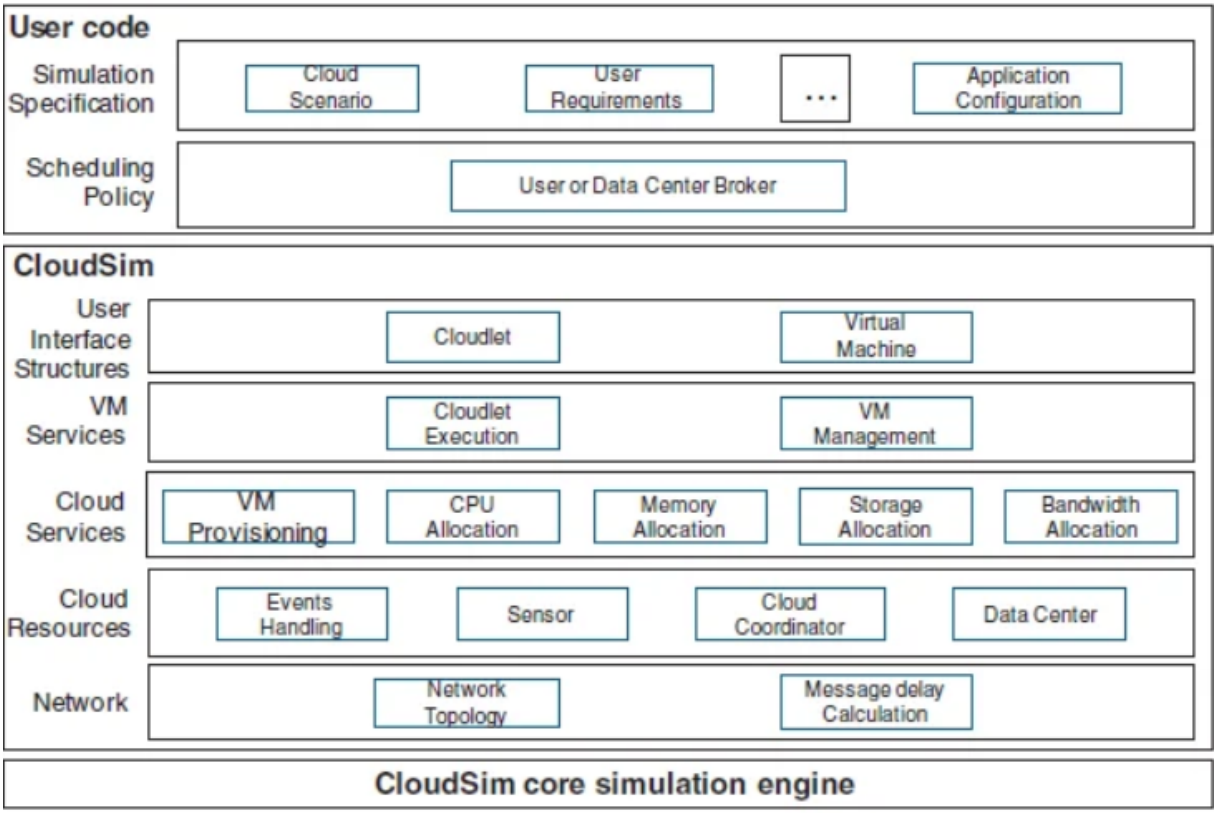


Figure 3-1: Architecture de CloudSim [21].

L'architecture de CloudSim est structurée en plusieurs couches, illustrées dans la figure ci-dessous.

La couche **User Code** où l'utilisateur configure divers éléments : les hôtes, les machines virtuelles, les applications (les cloudlets et leurs besoins), le nombre d'utilisateurs et les politiques de planification des brokers.

La couche **CloudSim** supporte la modélisation et la simulation de l'environnement de

datacenter basé sur Cloud, tel que des interfaces de gestion dédiées aux VMs, la mémoire, le stockage et la bande passante. La couche CloudSim gère l'instanciation et l'exécution des entités de base (VM, hôtes, Datacenters, applications) au cours de la période de simulation [22].

La couche CloudSim est constituée de plusieurs classes, voici quelques-unes des classes les plus utilisées pendant la simulation :

- **Datacenter :** Datacenter permet de représenter l'infrastructure fondamentale des services dans le Cloud computing, comprenant à la fois le matériel et le logiciel.
  À ce niveau, se trouvent des machines physiques appelées "Hosts". Chaque Datacenter utilise des algorithmes spécifiques pour distribuer la bande passante, la mémoire et le stockage entre les différentes machines virtuelles et "hosts" du Cloud.

- **DatacenterBroker :** le Broker joue le rôle de médiateur dans les négociations entre les utilisateurs et les fournisseurs de services, en fonction des exigences de qualité de service des utilisateurs. Les discussions visent à obtenir une allocation de ressources qui satisfait les besoins en QoS des utilisateurs, tout en facilitant le déploiement des tâches de service à travers les Clouds.

- **Host :** cette classe inclut des détails clés tels que la quantité de mémoire et de stockage, le type de processeurs (pour représenter une machine multi-core), ainsi que les politiques d'allocation pour le partage de la puissance de traitement entre les machines virtuelles, ainsi que les politiques d'attribution de mémoire et de bande passante aux machines virtuelles.

- **Vm :** cette classe sert à représenter une copie d'une machine virtuelle, qui est pris en charge et exécuté par une machine physique. Chaque partie de cette machine virtuelle a accès à une section qui stocke différentes informations sur cette machine virtuelle, comme sa mémoire, son processeur et son espace de stockage.

- **Cloudlet :** cette classe représente toute tâche exécutée sur une machine virtuelle, comme un traitement, un accès mémoire…etc. Elle conserve des paramètres décrivant les caractéristiques d'une tâche, comme sa durée, sa taille en million d'instructions (mi). De plus, elle offre des méthodes pour définir le temps d'exécution, l'état, le coût et l'historique d'une tâche.

3.3 Approche et démarche proposée

Dans cette partie, nous présentons formellement l'approche que nous avons adoptée dans le cadre de ce projet de fin d'études. Notre objectif dans ce contexte est d'écrire précisément notre méthode d'ordonnancement, qui vise à assigner des tâches aux machines virtuelles.

Par la suite, nous abordons un ordonnancement mono-objectif en nous concentrant sur une seule métrique : le temps d'exécution (Makespan). Nous évaluons ainsi les performances de notre algorithme HHO en le comparant aux heuristique Round Robin.

3.3.1 Evaluation Mono-objectif

Nous présentons ci-dessous l'algorithme round robin pour justifier les résultats dans notre projet de fin d'études visant à optimiser le critère du makespan.

3.3.1.1 Round robin

Dans l'ordonnancement round-robin, chaque tâche est exécutée à tour de rôle dans une file d'attente circulaire, pour une durée définie. Cet algorithme garantit que toutes les tâches reçoivent du temps de traitement, sans en oublier aucune. Cet algorithme est implémenté dans le simulateur CloudSim.

Voici les caractéristiques importantes de l'ordonnancement round-robin [23] :

- Le round robin est un algorithme préemptif.
- L'unité centrale de traitement est allouée au processus suivant après un intervalle de temps fixe, qui est appelé quantum de temps/tranche de temps.
- Le processus qui est préempté est ajouté à la fin de la file d'attente.
- La tranche de temps allouée à une tâche spécifique doit être minimale. Cependant, cette durée peut varier d'un système d'exploitation à l'autre.

3.3.1.2 makespan

Le temps d'exécution d'une Cloudlet correspond à la durée nécessaire pour l'exécuter sur une machine virtuelle. Étant donné qu'une tâche s'exécute sur un seul processeur (PE), nous appliquons la formule suivante :

$$makespane = Cloudletlength / VMmips.$$

D'où :

Cloudletlength : c'est la longueur de tache, la tache en mips (million instructions par seconde).

VMmips : vitesse de la machine virtuelle en MIPS.

3.3.2 Adaptation de la métaheuristique HHO

La métaheuristique HHO, qui a été adaptée dans le cadre de ce PFE pour l'affectation des tâches dans les différentes VMs, est représentée par le pseudocode suivant :

Entrées : La taille de la population N et le nombre maximal d'itérations T

Sorties : Le meilleur ordonnancement des tâches et sa valeur de fitness

Début :

Initialiser la population initiale de N ordonnancements (i=1,2,…..,N)

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

Calculer les valeurs de fitness des ordonnancements des tâches

Définir le meilleur ordonnancement des tâches

Pour (chaque ordonnancement) **faire**

Mettre à jour l'énergie initiale ***E0*** et
la force de saut **J** ►*E0*=2rand ()-1, J=2(1-rand ()) .

Mettez à jour le *E* en utilisant Eq **(3).**

Si ($|E| \geq 1$) **alors** ►Phase d'exploration

Mettre à jour le vecteur de localisation à l'aide d'Eq **(1).**

Fin Si

Si ($|E| < 1$) **alors** ►Phase d'exploitation

Si ($r \geq 0{,}5$ $\boldsymbol{et}$ $|E| \geq 0{,}5$ **alors**) ► Siège en douceur

Mettre à jour le vecteur de localisation à l'aide d'Eq **(4).**

Sinon si ($r \geq 0{,}5$ $\boldsymbol{et}$ $|E| < 0{,}5$) **alors** ►Assiéger durement

Mettre à jour le vecteur de localisation à l'aide d'Eq **(6).**

Sinon si ($r < 0{,}5$ $\boldsymbol{et}$ $|E| \geq 0{,}5$) **alors** ► Siège en douceur avec des plongées rapides progressives

Mettre à jour le vecteur de localisation à l'aide d'Eq **(10).**

Sinon si ($r < 0{,}5$ $\boldsymbol{et}$ $|E| < 0{,}5$) **alors** ► Assiégée avec des plongées rapides progressives

Mettre à jour le vecteur de localisation à l'aide d'Eq **(11).**

Fin Si

Fin Si

Fin Pour

Fin tant que

Retour le meilleur ordonnancement des tâches

FIN

Une phase de pré-simulation en utilisant le HHO adapté pour trouver les meilleures façons de répartir les tâches sur les machines virtuelles avant de les envoyer au Broker.

3.4 IHM développée

CloudSim ne dispose que d'une interface en mode console, sans interface graphique. Pour rendre l'accès et la manipulation de la simulation plus simples, nous avons développé notre propre interface utilisateur.

Avant d'analyser le comportement de notre algorithme, nous devons d'abord passer par une étape de configuration pour définir les paramètres de simulation.

Les interfaces suivantes décrivent les différentes étapes de configuration de la simulation.

Interface Principale :

Dès que notre application démarre, une fenêtre d'accueil s'affiche comme illustré dans la figure 3.2, présentant les détails de notre projet de fin d'études.

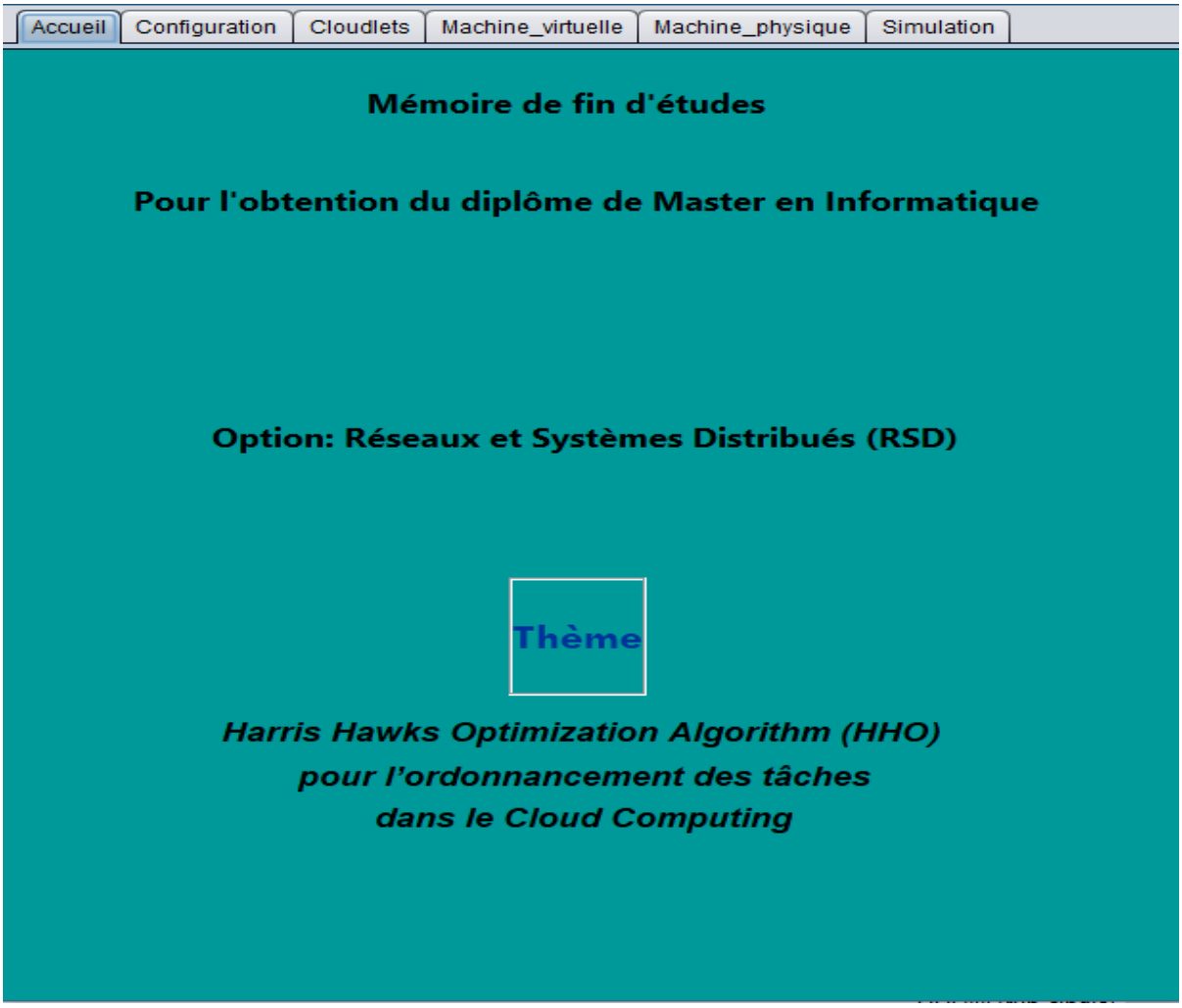


Figure 3-2: Interface principale.

Configuration :

Notre application propose deux choix de configuration. Le premier choix, par défaut comme illustré dans la figure 3.3

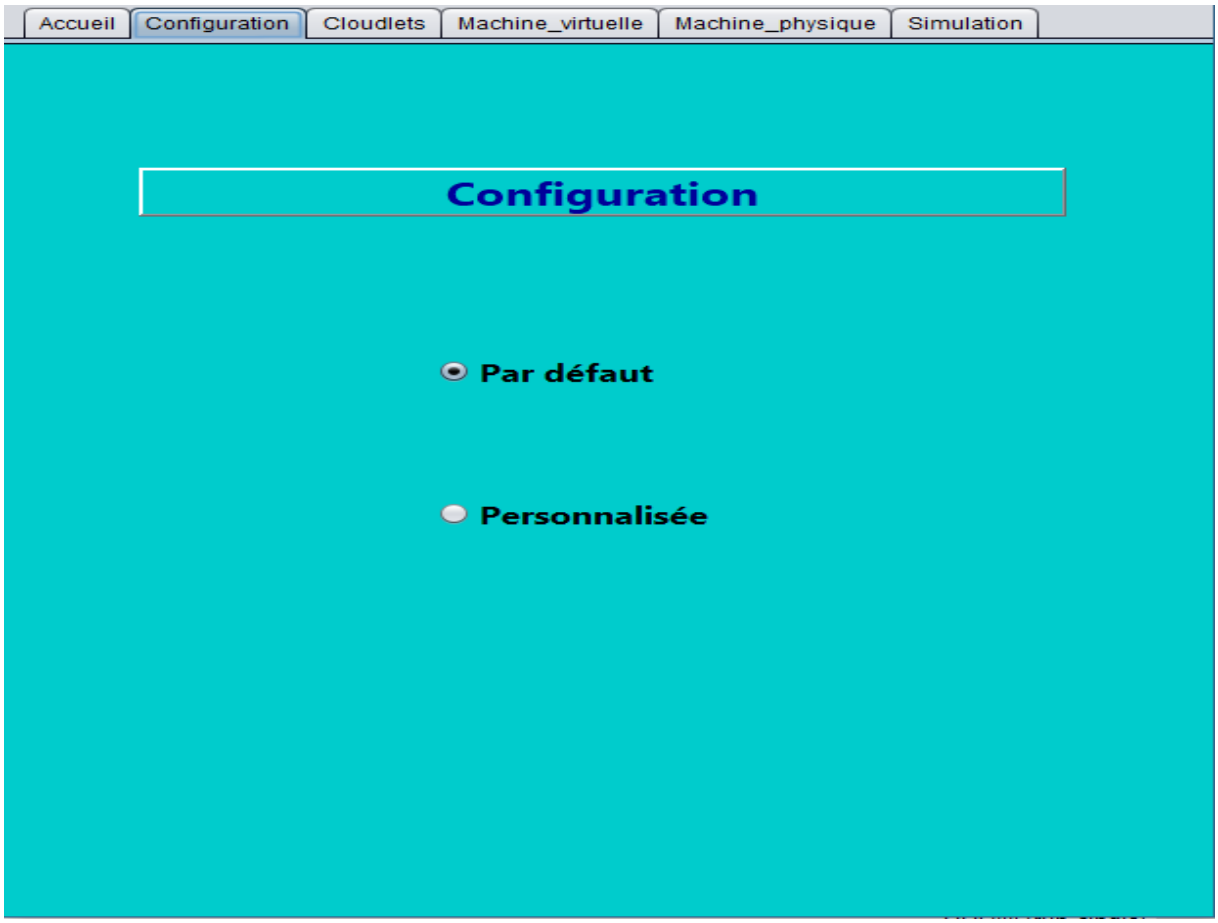


Figure 3-3: Configuration par défaut.

Le deuxième choix propose une configuration personnalisée, permettant d'accéder aux paramètres et de les modifier pour les machines physiques, virtuelles, ainsi que les cloudlets.

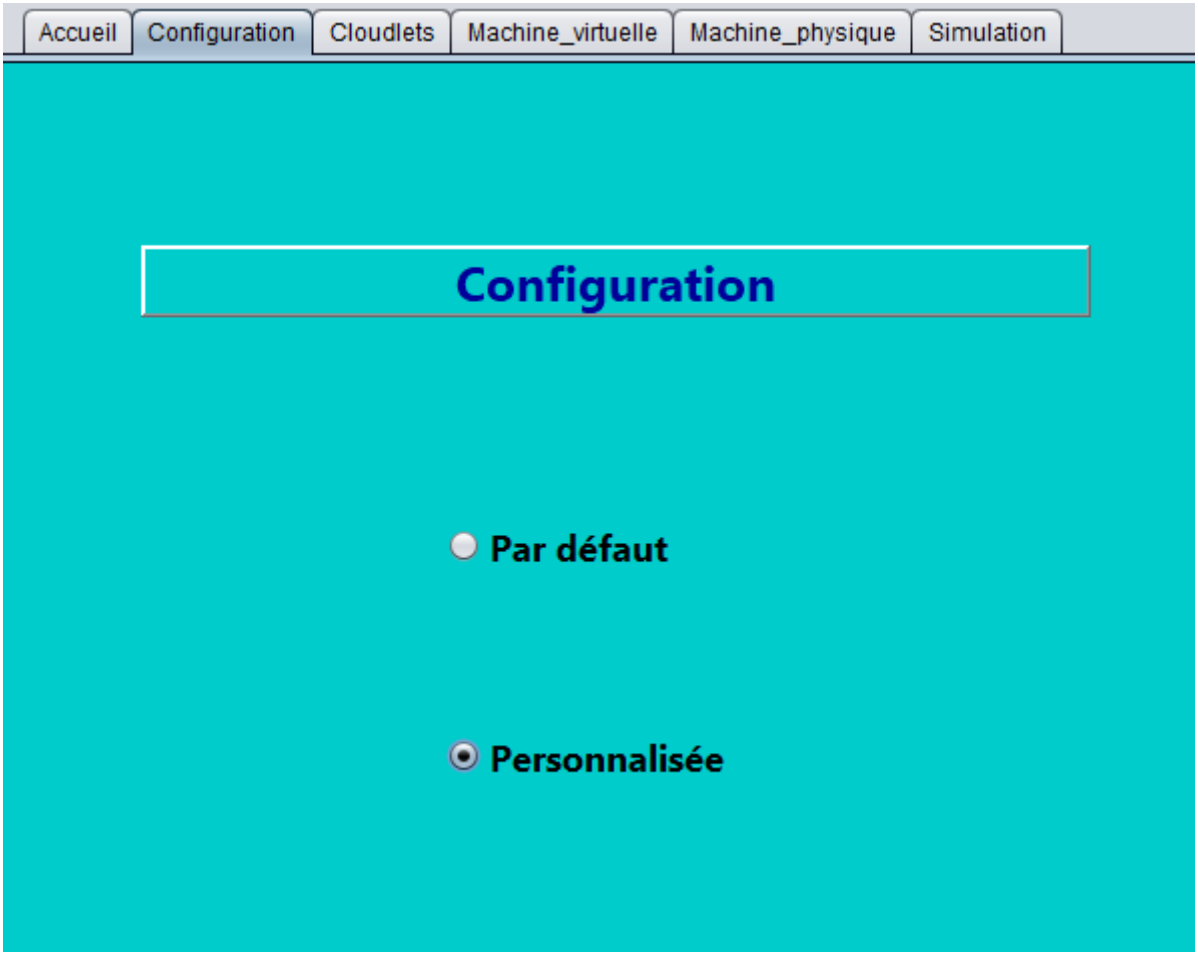


Figure 3-4: Configuration personnalisée.

Configuration des Cloudlets :

Cette interface utilisateur, nous permet de définir les propriétés des Cloudlets telles que le nombre de Cloudlets, length min et length max comme illustré dans la figure 3.5.

Figure 3-5: Création des Cloudlets.

Configuration des machines virtuelles :

À cette étape, nous configurons les paramètres requis pour la création des machines virtuelles, de telle sorte que les VMs soient obligatoirement hébergées sur un hôte et chaque hôte peut contenir une ou plusieurs machines virtuelles.

Chaque machine virtuelle possède une vitesse de traitement (MIPS), une capacité de RAM, une bande passante et une capacité de stockage, comme illustré dans la figure 3.6.

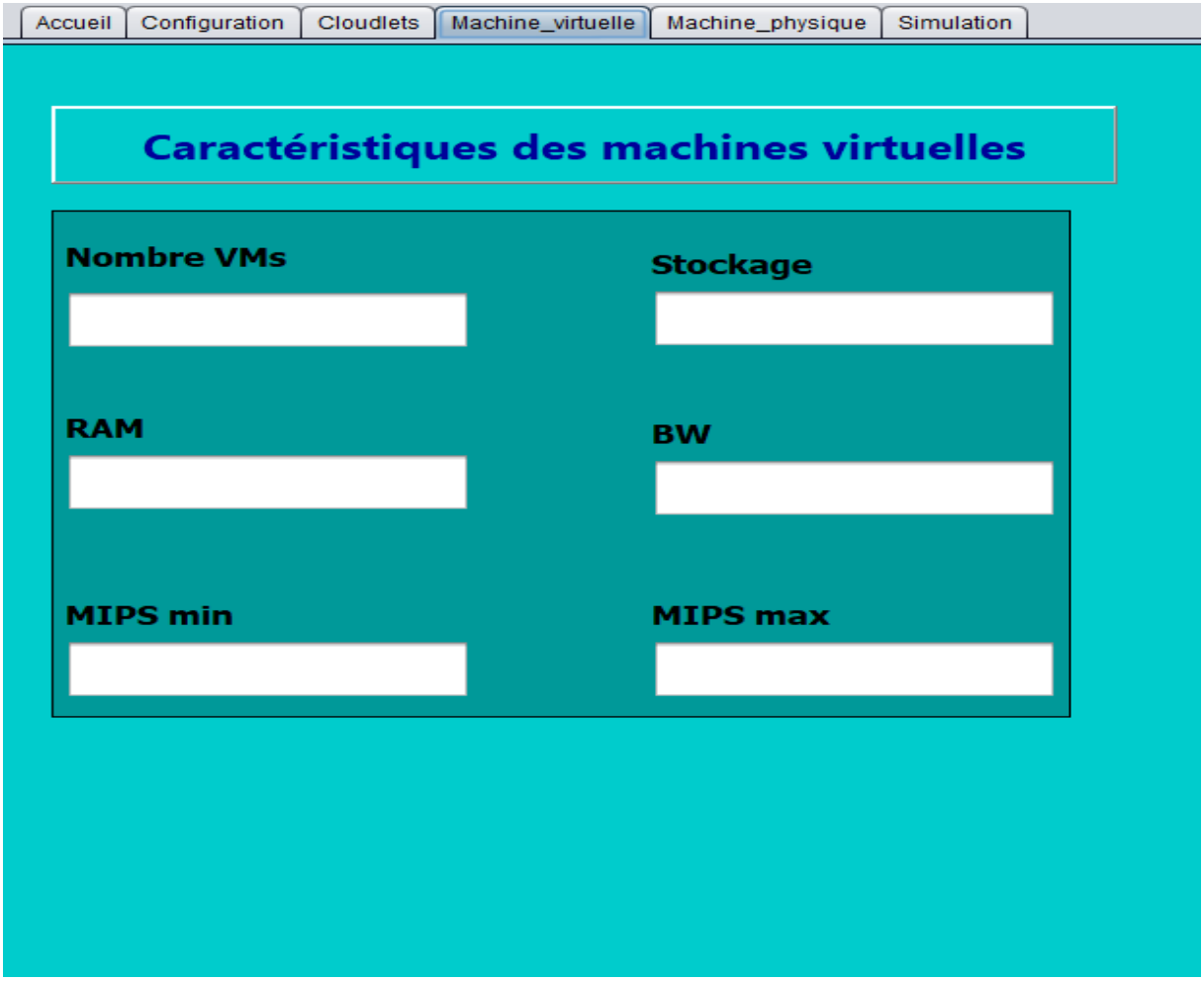


Figure 3-6: Création des machines virtuelles.

Configuration des machines physiques :

Pour configurer le Cloud, nous débutons en saisissant les informations essentielles concernant les machines physiques. Cela inclut le nombre de DataCenters, le nombre d'hôtes dans chaque DataCenter, le nombre de processeurs (PE) dans chaque hôte, la vitesse de chaque processeur (MIPS), la quantité de RAM, la bande passante et également la capacité de stockage comme indiquer dans la figure 3.7.

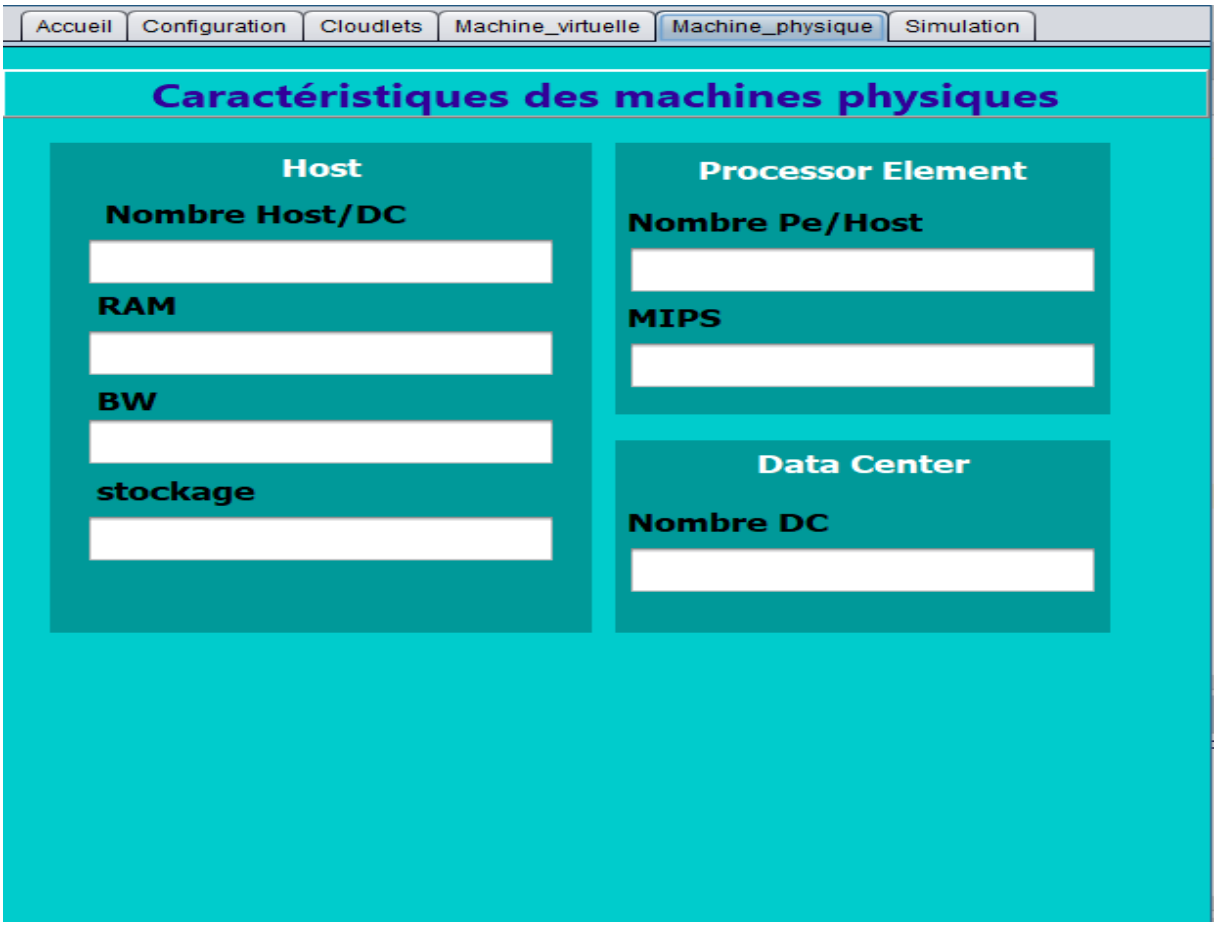


Figure 3-7: Création des machines physiques.

Simulation :

A partir de l'interface présentée dans la figure 3.8, l'exécution de la simulation est lancée en cliquant sur le bouton **Démarrer**, tel que le type d'ordonnancement à utiliser pendant la simulation c'est l'ordonnancement mono-objectif par rapport au makespan.

Figure 3-8 : Interface de simulation.

Une fois les étapes de configuration et de pré-simulation achevées, la répartition des tâches sur les différentes machines virtuelles est transmise au courtier pour démarrer la simulation et afficher les résultats.

3.5 Résultats obtenus et étude comparative

Dans cette section, nous examinerons la configuration des simulations ainsi que les résultats obtenus après l'exécution de l'algorithme HHO pour l'ordonnancement des tâches.

Les différentes simulations ont été réalisées sous une machine sous Windows 10 de 64 bits, avec un processeur Intel(R) Core (TM) i3-7020U CPU @ 2.30GHz, une vitesse de 2.30 Ghz et une capacité mémoire de 4GB.

Les simulations ont été réalisées dans un environnement hétérogène, comme détaillé dans les tableaux 3.1 et 3.2, lesquels présentent les valeurs attribuées à chaque paramètre.

Data center	Nombre	2
Host	Nombre	5
	PES	4 Dual core et 26 Quad core
	MIPS	4000
	RAM	8 GB
	Bande passante	3000 MB
	Stockage	1 TB
Machine virtuelle	Nombre	25
	MIPS	100 jusqu'à 1000
	RAM	1 GB
	Bande passante	1000 MB
	Stockage	10 GB
	Type de politique	Timed Shared
Cloudlet	Nombre	200 – 400 – 600– 800 – 1000

Tableau 3-1: Les paramètres de simulation de CloudSim [24]

Type de cloudlets	MIPS des taches	Distribution
Tiny	200	35%
Small	1000	40%
Medium	5000	5%
Large	15000	15%
Extra Large	45000	5%

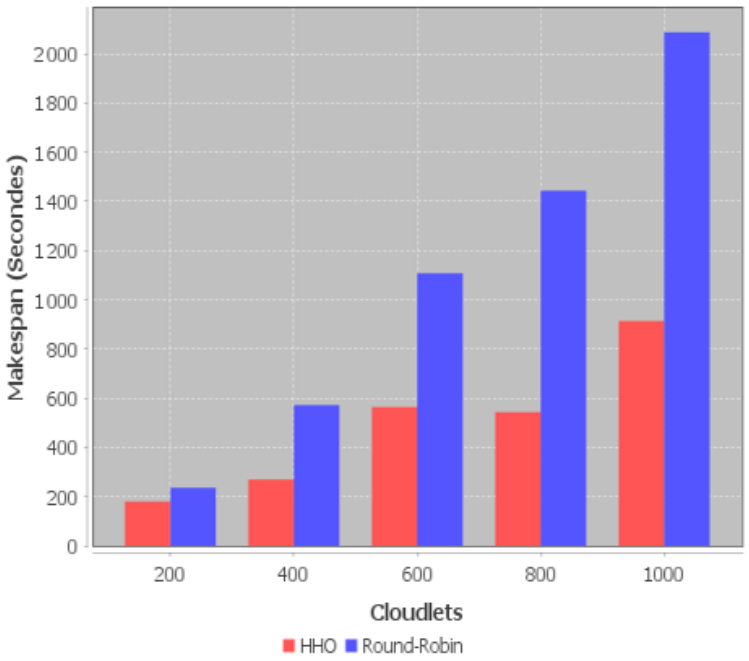
Tableau 3-2: Les types des cloudlets [24].

Pour avoir des résultats plus fiables, nous avons fait la moyenne des résultats sur 10 simulations pour l'ordonnancement mono-objectif, parce que les algorithmes utilisés sont non déterministes.

3.5.1 Evaluation de l'ordonnancement mono-objectif

Pour valider l'ordonnancement mono-objectif, nous avons comparé la métaheuristique HHO et Round Robin en termes de makespan dans les deux configurations par défaut et personnalisée.

Configuration par défaut :



200	179.98	235.98
400	269.582	571.908
600	563.984	1107.88
800	543.564	1443.84
1000	913.928	2087.8

Figure 3-9: Comparaison entre HHO et Round Robin en termes de makespan.

Configuration personnalisée :

Les 4 figures suivantes montrent les résultats de la comparaison entre HHO et l'heuristique Round Robin en termes de makespan (mesuré en secondes) pour des nombres de cloudlets égaux à 500, 700 et 1200 respectivement.

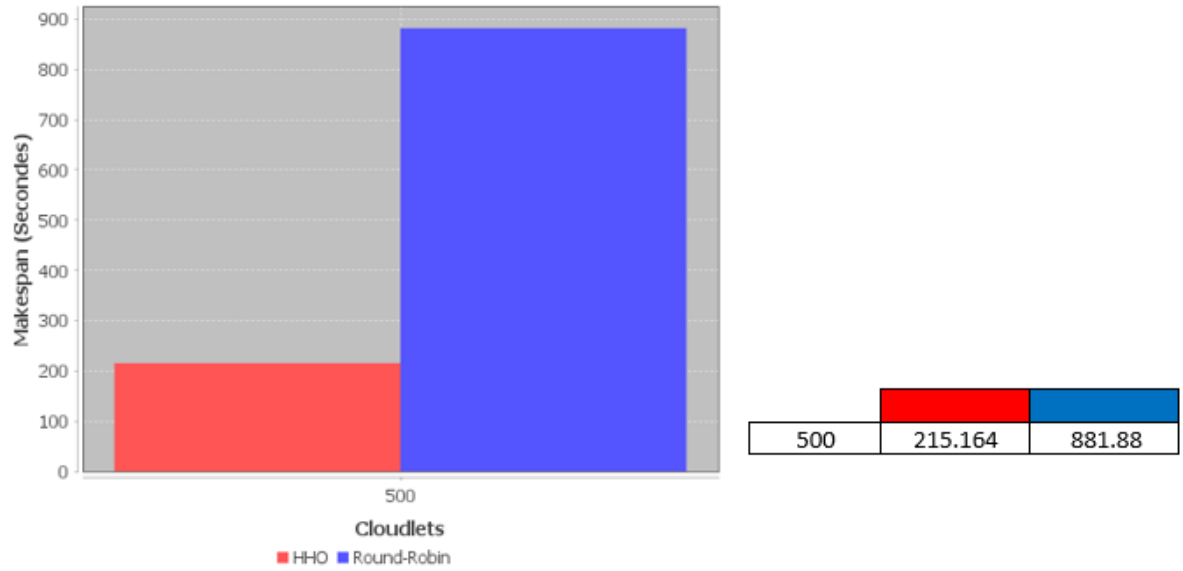


Figure 3-10: Comparaison entre HHO et RR en termes de makespan pour 500 cloudlets.

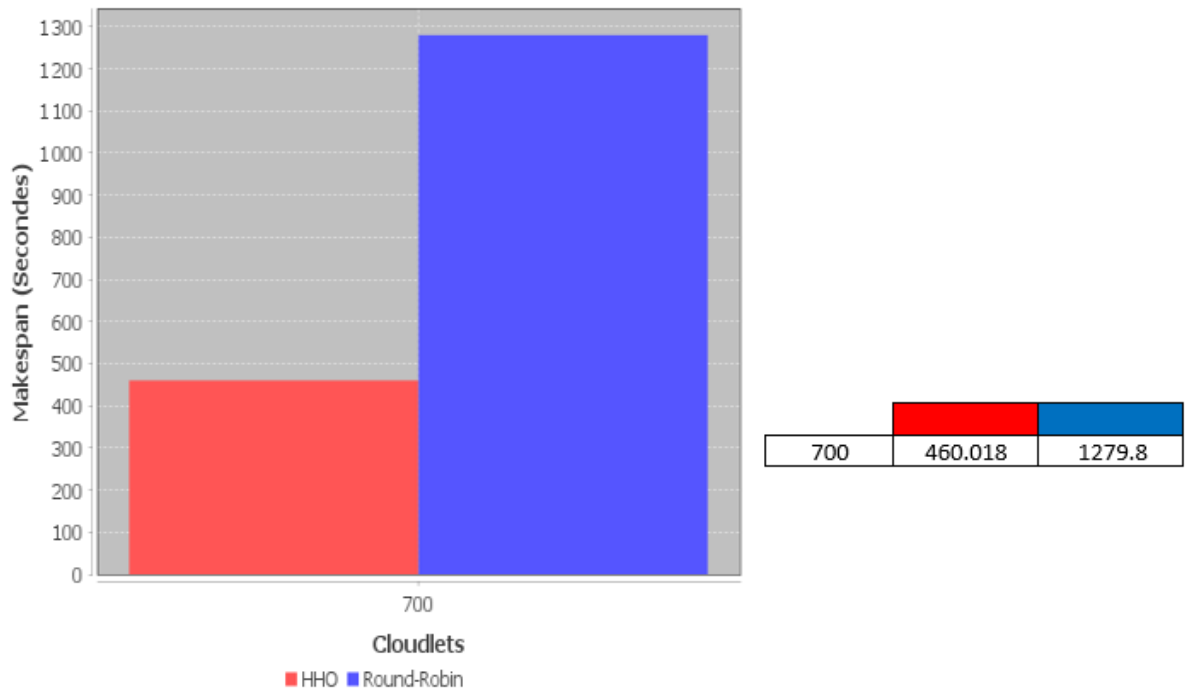
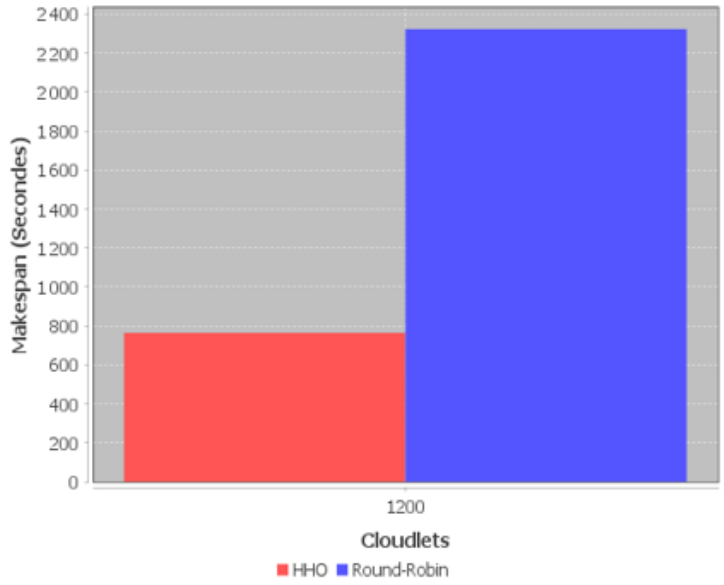


Figure 3-11: Comparaison entre HHO et RR en termes de makespan pour 700 cloudlets.



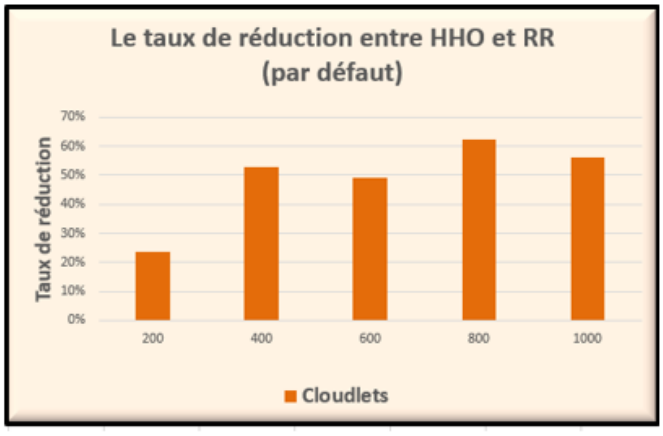
1200	764.026	2323.608

Figure 3-12: Comparaison entre HHO et RR en termes de makespan pour 1200 cloudlets.

Résultat : Nous remarquons, selon les 4 figures précédentes que le HHO a donné de meilleurs résultats par rapport au RR en termes de makespan. Le nombre de cloudlets a un impact sur le temps d'exécution : plus le nombre de cloudlets augmente, plus le temps d'exécution s'allonge pour les deux approches. En revanche, les performances du HHO par rapport au RR sont meilleures dans les deux configurations.

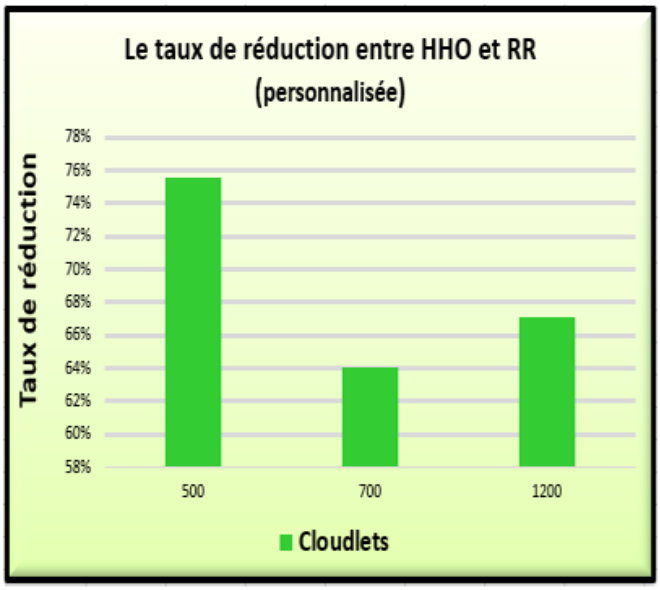
Comparaison des deux approches en termes de taux de réduction du makespan :

Dans cette partie, nous comparons les deux approches courantes RR et HHO. L'objectif est de déterminer lequel des deux permet de réduire le temps d'exécution des tâches de manière plus efficace. Les 2 figures suivantes montrent les résultats de cette comparaison.



Cloudlets	HHO	RR	Taux
200	179,98	235,98	24%
400	269,582	571,908	53%
600	563,984	1107,88	49%
800	543,564	1443,84	62%
1000	913,928	2087,8	56%

Figure 3-13: Le taux de réduction entre HHO et RR dans la configuration par défaut



Cloudlets	HHO	RR	Taux
500	215,164	881,88	76%
700	460,018	1279,80	64%
1200	764,026	2323,608	67%

Figure 3-14: Le taux de réduction entre HHO et RR dans la configuration personnalisée

Résultat : Les résultats obtenus montrent un gain significatif en termes de temps d'exécution lorsque l'on utilise l'algorithme HHO comparé à l'approche Round Robin. Ce gain est quantifié par le taux de réduction du makespan pour différents nombres de cloudlets. A titre d'exemple, pour 800 cloudlets, l'algorithme Round Robin prend 1443.84 secondes pour exécuter les tâches, tandis que l'algorithme HHO ne prend que 543.564 secondes. Cela représente un taux de réduction d'environ ~62% (62.35%). En termes de gain en temps, l'utilisation de HHO permet d'économiser 62 % du temps d'exécution par rapport à Round Robin.

Ces gains montrent clairement l'efficacité de l'algorithme HHO par rapport à Round Robin. En réduisant le makespan de manière significative, HHO permet d'améliorer la performance et l'efficacité des systèmes traitant un grand nombre de cloudlets.

3.6 Conclusion

Ce chapitre a été dédié à la présentation de notre contribution dans le cadre de ce PFE. Nous avons commencé par introduire les outils de simulation utilisés : le langage JAVA, l'IDE NetBeans, et le simulateur CloudSim.

Les résultats obtenus démontrent l'efficacité de notre approche comparée à l'approche classique en termes de temps de réponse et de taux de réduction.

Conclusion générale

Dans ce mémoire, nous avons utilisé la métaheuristique HHO pour identifier la solution optimale dans un délai raisonnable. La problématique étudiée concerne la répartition des tâches sur différentes machines virtuelles.

Pour mettre en œuvre l'approche proposée, nous avons opté pour le simulateur CloudSim en raison de ses capacités avancées.

Les performances de notre algorithme sont évaluées uniquement en termes de temps de réponse. Pour valider l'approche développée dans ce projet de fin d'études, nous l'avons comparée à un algorithme classique, le Round Robin qui est déjà intégré dans CloudSim.

Les simulations effectuées avec CloudSim mettent en évidence l'efficacité de l'approche proposée par rapport à l'algorithme Round Robin.

Pour la suite de ce travail, il serait pertinent de l'étendre à d'autres contraintes telles que la fiabilité, cout et la tolérance aux pannes. Il serait également intéressant d'appliquer d'autres métaheuristiques afin de les comparer avec le HHO.

Bibliographie

[1] Le cloud computing. Disponible sur : https://docplayer.fr/434716-Le-cloud-computing-etudes-sectorielles-direccte-septembre-2012-www-idf-direccte-gouv-fr-une-nouvelle-filiere-fortement-structurante-ile-de-france.html . Consulté le 20/04/2024.

[2] MADANI Aymen Ibnou Eddin. Butterfly optimization algorithm (BOA) pour l'ordonnancement des tâches dans le Cloud computing. Mémoire de Master en Informatique. Université Abou Bakr Belkaid Tlemcen. 2023.

[3] MEZOUER Leila, MEKKIOUI Nadjet. Tolérances aux Pannes dans les infrastructures Cloud Computing. Mémoire de Master en Informatique. Université Abou Bakr Belkaid Tlemcen 2016.

[4] Les avantages et les inconvénients du Cloud computing. Disponible sur : Les avantages et les inconvénients du cloud computing | Skillsoft's Global Knowledge. Consulté le 26/04/2024.

[5] Qu'est-ce que la virtualisation ? Disponible sur : Qu'est-ce que la virtualisation ? – La virtualisation du cloud computing expliquée – AWS (amazon.com) . Consulté le 26/04/2024.

[6] Comprendre les data centers : définition et fonctionnement . Disponible sur : Comprendre les data centers : définition et fonctionnement - Le Gros Blog (lgblog.fr) . Consulté le 26/04/2024.

[7] MALTI Arslan Nedhir, KHEDDAOUI Med Badreddine. Ordonnancement des tâches multi-objectif dans le Cloud computing . Mémoire de Master en Informatique. Université Abou Bakr Belkaid Tlemcen 2020.

[8] Qu'est-ce qu'un SaaS (logiciel en tant que service) ? Disponible sur : Qu'est-ce que le SaaS ? – Le logiciel en tant que service expliqué – AWS (amazon.com). Consulté le 28/04/2024.

[9] Types de services Cloud computing . Disponible sur : Main cloud service models: IaaS, PaaS and SaaS (stackscale.com) . Consulté le 28/04/2024.

[10] Classes des méthodes de résolutions. Disponible sur : https://www.researchgate.net/figure/Classification-des-methodes-de-resolution-deprobi-emes-doptimisation-125_fig6_339167527. Consulté le 24/04/2024.

[11] Heuristique - Définition. Disponible sur : https://www.techno-science.net/glossaire-definition/Heuristique.html. Consulté le 29/04/2024.

[12] Ilhem BOUSSAÏD, Perfectionnement de métaheuristiques pour l'optimisation continue, Thèse de doctorat en informatique, Université paris-est Créteil avec l'université des sciences et de la technologie houari Boumediene 2013.

[13] MAHBOUBI THEMER. Une approche intelligente pour un problème d'optimisation multicritère. Mémoire de Master en Informatique. Université de 8 Mai 1945 – Guelma - 2019.

[14] Algorithme de colonies de fourmis. Disponible sur : https://www.datafranca.org/wiki/Algorithme_de_colonies_de_fourmis. Consulté le 29/04/2024.

[15] Mehenni Tahar, Utilisation des métaheuristiques pour résoudre un problème d'ordonnancement sur machine a contrainte de ressource non renouvelable. Mémoire de Magister en informatique, Université Mohamed Boudiaf de M'sila 2006.

[16] Harris Hawk Optimization: A Survey onVariant and Applications. Disponible sur : https://www.hindawi.com/journals/cin/2022/2218594/. Consulté le 09/05/2024.

[17] Les faucons de Harris. Disponible sur : La nature de Kathy : Harris Hawks alias Wolves of the Desert (naturefotog.blogspot.com). Consulté le 23/05/2024.

[18] Java (langage). Disponible sur : Java (langage) - Wikiwand. Consulté le 25/05/2024.

[19] NetBeans-Définition. Disponible sur : NetBeans : définition et explications (techno-science.net). Consulté le 25/05/2024.

[20] Qu'est-ce que CloudSim. Disponible sur : Qu'est-ce que CloudSim ? – StackLima. Consulté le 27/05/2024.

[21] CloudSim Simulation. Disponible sur : CloudSim Simulation Toolkit: An Introduction - Cloudsim Tutorials . Consulté le 27/05/2024.

[22] Le Framework de simulation de l'environnement du Cloud. Disponible sur : https://graal.ens-lyon.fr/~ecaron/m2rts/2015/blogbazm/. Consulté le 29/05/2024.

[23] Algorithme d'ordonnancement round robin avec exemple . Disponible sur : Algorithme d'ordonnancement Round Robin avec exemple | My Star Idea Consulté le 29/05/2024.

[24] Alsadie, Deafallah. "TSMGWO: Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers." IEEE Access 9 (2021): 37707-37725.

Résumé :

Le Cloud computing est une innovation technologique qui assure un accès simple et fiable à des données et des services stockés sur des serveurs distants, garantissant une qualité de service élevée et une disponibilité constante. Ce travail vise à l'optimisation mono-objectif dans le Cloud computing en utilisant la métaheuristique Harris Hawks Optimization (HHO) pour réduire le temps d'exécution. Le problème abordé concerne l'affectation des tâches aux différentes machines virtuelles. Notre approche a montré de meilleures performances comparées à la méthode classique (Round Robin). Nous avons utilisé le simulateur CloudSim pour les simulations, et développé l'interface utilisateur en Java pour faciliter les interactions avec le simulateur.

Mots clés: Cloud computing, CloudSim, HHO, Round Robin.

Abstract :

The Cloud computing is a technological innovation that ensures simple and reliable access to data and services stored on remote servers, ensuring high quality of service and constant availability. This work aims at mono-objective optimization in Cloud computing using the Harris Hawks Optimization (HHO) metaheuristic to reduce execution time. The problem addressed concerns the allocation of tasks to different virtual machines. Our approach has shown better performance compared to the classical method (Round Robin). We used the CloudSim simulator for simulations, and developed the user interface in Java to facilitate interactions with the simulator.

Keywords: Cloud computing, CloudSim, HHO, Round Robin.

ملخص

الحوسبة السحابية هي ابتكار تكنولوجي يوفر وصولاً بسيطًا وموثوقا إلى البيانات والخدمات المخزنة على خوادم بعيدة مما يضمن جودة عالية للخدمة وتوافرها باستمرار، ويهدف هذا العمل إلى تحسين أحادي الهدف في الحوسبة السحابية باستخدام métaheuristique HHO لتقليل وقت التنفيذ. تتعلق المشكلة التي تمت معالجتها بتعيين المهام إلى أجهزة افتراضية مختلفة. أظهر نهجنا أداء أفضل مقارنةً بالطريقة التقليدية **(**Round Robin**)** استخدمنا محاكي Cloudsim للمحاكاة , وطورنا واجهة المستخدم بلغة **java** لتسهيل التفاعل مع المحاكي.

الكلمات الرئيسية: الحوسبة السحابية , Round Robin **,** HHO **,** Cloudsim