République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaïd 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

White Shark Optimisation Algorithm pour l'ordonnancement des tâches dans le Cloud computing

Réaliser par :

- MEDJDOUB Mohammed Yacine

Présenté le 28 juin 2025
Devant le jury composé de :

- Mr. Salim Ziani-Cherif President

- Mr. Ahmed Khalid Yassine Settouti Examinateur

- Mr. BENMAMMAR Badr Encadrant

- Mr. MALTI Arslan Nedhir Co-Encadrant

Année universitaire : 2024-2025

Remerciements

Je tiens à remercier **Dieu Tout-Puissant**, qui m'a armé de courage, de volonté et surtout de **Patience**, tout au long de ce travail.

Remerciements

J’exprime ma sincère gratitude à toutes les personnes qui ont contribué à la réalisation de ce Mémoire par leur aide et leur soutien.

Mes premiers remerciements vont à mes encadrants, **Malti Arslan Nedhir** et **Benmammar Badr**, Pour leur accompagnement, leurs précieux conseils et leurs encouragements tout au long de ce travail.

Je remercie également Monsieur **Salim Ziani-Cherif** et Monsieur **Ahmed Khalid Yassine Settouti**, membres du jury, pour avoir pris le temps d’examiner et d’évaluer mon mémoire.

Enfin, j’adresse une pensée particulière à ma famille, à qui j’exprime ma profonde reconnaissance Pour son soutien inconditionnel et ses encouragements constants.

Dédicace

Je dédie ce modeste travail à :

- ***Mes chers parents***
- ***Mes frères et sœurs***
- ***Mes amis et camarades***

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

Acronym	Signification
API	Application Program Interface
CPU	Central Processing Unit
DC	Data Center
SLA	Service-Level Agreement
PAYG	Pay As You Go
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
WSO	White-Shark-Optimizer
GWO	Grey Wolf Optimizer
SSA	Salp Swarm Algorithme
VM	Virtual Machine
QoS	Quality of Service
RR	Round Robin

Résumé

L'ordonnancement des tâches joue un rôle essentiel dans l'optimisation des performances d'un environnement de Cloud computing. En effet, les fournisseurs et les utilisateurs des services Cloud poursuivent des objectifs souvent divergents, rendant la gestion des ressources complexe. Un ordonnanceur efficace doit ainsi proposer un équilibre satisfaisant entre ces différentes contraintes. Ce travail s'inscrit dans cette problématique en cherchant à optimiser l'affectation des tâches aux machines virtuelles. Pour cela, nous avons exploité la métaheuristique White Shark Optimizer (WSO) en intégrant trois métriques de qualité de service (QoS) : le makespan, le coût et l'énergie. L'évaluation des performances a été réalisée à l'aide de CloudSim, et les résultats obtenus démontrent l'efficacité de l'approche proposée.

Mots clés : Cloud computing, optimisation multi-objectif, QoS, WSO, CloudSim.

Abstract

Task scheduling plays a crucial role in optimizing the performance of a Cloud computing environment. Cloud service providers and users often have conflicting objectives, making resource management a complex challenge. An efficient scheduler must find a balance that satisfies these different constraints. This work focuses on optimizing task allocation to virtual machines. To achieve this, we utilized the White Shark Optimizer (WSO) while incorporating three quality of service (QoS) metrics: makespan, cost, and energy. Performance evaluations were conducted using CloudSim, and the results obtained demonstrate the effectiveness of the proposed approach.

Keywords: Cloud computing, multi-objective optimization, QoS, WSO, CloudSim.

الملخص

تلعب جدولة المهام دورًا حاسمًا في تحسين أداء بيئة الحوسبة السحابية. غالبًا ما تكون لدى مزوّدي الخدمات السحابية والمستخدمين أهداف متعارضة، مما يجعل إدارة الموارد تحديًا معقدًا. يجب على جدول زمني فعّال أن يجد توازنًا يحقق هذه الشروط المختلفة. يركّز هذا العمل على تحسين تخصيص المهام إلى الآلات الافتراضية. ولتحقيق ذلك، استخدمنا خوارزمية القرش الأبيض (WSO) مع دمج ثلاث مقاييس لجودة الخدمة (QoS): مدة الإنجاز، التكلفة، والطاقة. أجريت تقييمات الأداء باستخدام CloudSim (بيئة محاكاة)، وأظهرت النتائج التي تم الحصول عليها فعالية النهج المقترح.

الكلمات المفتاحية: الحوسبة السحابية، التحسين متعدد الأهداف، جودة الخدمة,CloudSim، WSO، QoS.

Introduction Générale

Le Cloud computing, ou informatique en nuage, est un modèle informatique où les ressources de calcul et de stockage sont gérées par des serveurs distants accessibles via une connexion internet sécurisée. Ce paradigme permet aux utilisateurs d'exécuter des applications et d'accéder aux données sans avoir à gérer directement l'infrastructure matérielle. Les ordinateurs, smartphones, tablettes et autres objets connectés deviennent ainsi de simples points d'accès aux services hébergés dans le Cloud. L'ordonnancement des tâches dans un environnement Cloud est un défi majeur, car il vise à allouer efficacement les ressources disponibles en fonction des besoins des tâches et des contraintes du système. Ce problème devient encore plus complexe dans un environnement hétérogène comme le Cloud, où plusieurs objectifs doivent être pris en compte simultanément, tels que la réduction du temps d'exécution, l'optimisation de l'utilisation des ressources et la minimisation des coûts énergétiques. Pour répondre à ces défis, les métaheuristiques sont largement utilisées pour proposer des solutions optimales à des problèmes d'optimisation complexes. Parmi ces approches, le White Shark Optimizer (WSO) est un algorithme bio-inspiré qui simule le comportement des requins blancs lors de la chasse et de la recherche de proies. Grâce à ses stratégies d'exploration et d'exploitation, il permet d'améliorer l'ordonnancement des tâches dans le Cloud en recherchant des solutions optimales de manière efficace.

Ce mémoire se structure en plusieurs chapitres. Le premier est consacré à la présentation des concepts fondamentaux du Cloud computing. Le deuxième chapitre porte sur les métaheuristiques d'optimisation, avec un focus particulier sur l'algorithme WSO, dont le fonctionnement et les objectifs dans le cadre de l'ordonnancement des tâches seront étudiés. Le troisième chapitre détaillera l'implémentation de cet algorithme ainsi que les tests menés pour évaluer ses performances. Enfin, une annexe regroupera l'ensemble des outils et technologies utilisés au cours de cette étude.

Chapitre 1: Cloud Computing

1.1 Introduction

Depuis son émergence, la technologie de l'Internet a connu une évolution rapide et continue. Parmi les tendances marquantes dans le domaine des technologies de l'information et de la communication (TIC), le Cloud computing s'impose comme une innovation majeure. Ce modèle technologique permet aux entreprises de diminuer les coûts liés à l'exploitation des logiciels en les rendant accessibles directement en ligne. Grâce au Cloud, il est désormais possible d'accéder aux applications via Internet, tout en offrant la flexibilité de créer, configurer et personnaliser ces services à distance.

1.2 Historique du Cloud computing

Dans les années 1960 et 1970, les entreprises utilisaient de grands ordinateurs centraux pour traiter leurs données. Ces systèmes, bien que performants, représentaient un investissement coûteux et nécessitaient des terminaux pour permettre aux employés d'y accéder à distance. Au cours des années 1980, l'émergence des serveurs basés sur des ordinateurs classiques a marqué un tournant. Ces infrastructures, plus accessibles financièrement, ont permis aux entreprises de mieux gérer leurs ressources informatiques et de donner aux utilisateurs un contrôle accru sur leurs opérations.

Avec la généralisation d'Internet dans les années 1990, l'accès aux serveurs centraux est redevenu une nécessité. La demande croissante des utilisateurs a exigé des serveurs web plus puissants capables de supporter un grand nombre de requêtes simultanées. Dès lors, le besoin en stockage et en services en ligne n'a cessé d'augmenter.

L'amélioration des débits Internet et de la bande passante a rendu plus efficace l'hébergement des applications sur des infrastructures distantes. Cette évolution a ouvert la voie à un modèle informatique dans lequel plusieurs utilisateurs partagent la même infrastructure, ce qui optimise les performances tout en réduisant les coûts.

À la fin des années 1990, les Data Centers utilisaient en moyenne moins de 10 % de leur capacité, réservant le reste pour des périodes de forte demande, comme les fêtes. Pour remédier à ce problème, Amazon a introduit un modèle de location de serveurs à la demande, permettant aux entreprises de bénéficier de ressources informatiques flexibles. La société a également lancé des services comme S3 (Simple Storage Service) pour le stockage des données et EC2 (Elastic Compute Cloud) pour la puissance de calcul. Cette approche a non seulement optimisé l'utilisation des infrastructures informatiques, mais a également posé les bases du Cloud computing moderne.

L'ère de l'information a connu de nombreuses évolutions : des mainframes aux mini-ordinateurs, puis aux ordinateurs personnels, jusqu'à l'émergence du Cloud computing, qui représente une étape importante parmi d'autres dans cette évolution continue.

La figure 1.1 illustre les différentes technologies dans l'ère de l'information.

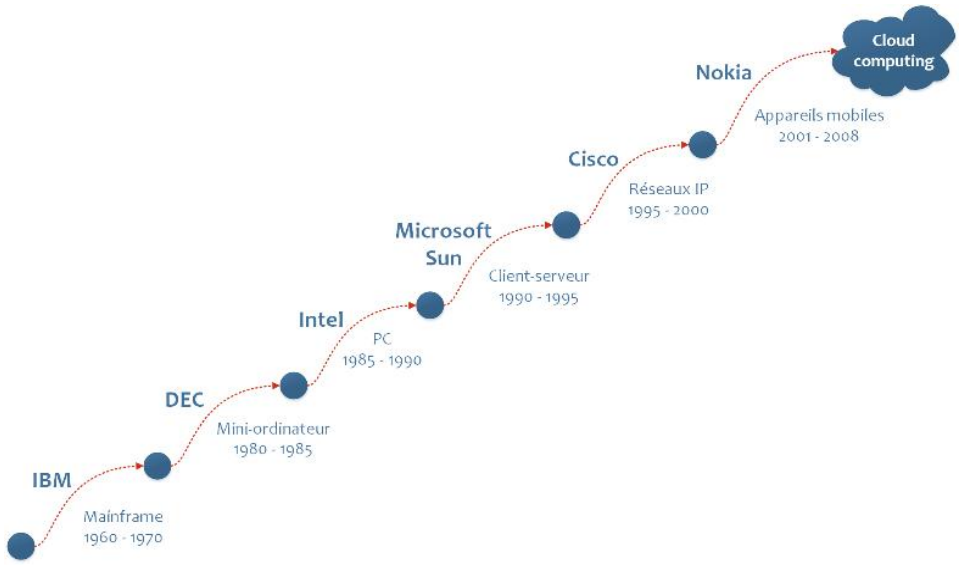

Figure 1.1: Les différentes technologies dans l'ère de l'informatique [1].

1.3 Notions de base

1.3.1 Définition

Le Cloud computing, ou informatique en nuage, désigne l'utilisation à distance de ressources informatiques telles que la puissance de calcul ou la capacité de stockage, accessibles via un réseau – le plus souvent Internet. Ces ressources sont mises à disposition à la demande, généralement selon des modalités de tarification basées sur l'usage (performances, bande passante, etc.) ou sous forme de forfaits. Ce modèle se distingue par sa grande flexibilité : selon les compétences de l'utilisateur, il est possible soit de gérer soi-même l'environnement serveur, soit de se limiter à l'utilisation d'applications hébergées à distance. D'après la définition du National Institute of Standards and Technology (NIST), le Cloud computing correspond à un accès à la demande, via un réseau de télécommunication, à un ensemble de ressources informatiques partagées et configurables. Il s'agit ainsi d'une délocalisation de l'infrastructure informatique [1, 2].

Figure 1.2: Architecture Cloud [2].

1.3.2 Avantages et inconvénients

1.3.2.1 Avantages

Le Cloud Computing présente de nombreux avantages. Certains d'entre eux sont énumérés ci-dessous :

➢ Service d'une grande disponibilité.

➢ Aucun investissement préalable.

➢ Facturation à la consommation.

➢ Accès simplifié utilisant des navigateurs web.

1.3.2.3 Inconvénients

Les grandes catégories de risques liées à l'adoption du Cloud se déclinent selon les points suivants :

➢ **Sécurité et privacité :** C'est la plus grande préoccupation concernant le Cloud computing. Étant donné que la gestion des données et la gestion de l'infrastructure dans le Cloud sont fournies par des tiers, il est toujours possible de transférer les informations sensibles à ces fournisseurs. Bien que les fournisseurs du Cloud computing garantissent des comptes protégés par un mot de passe plus sécuriser, tout signe de violation de sécurité entraînerait la perte des clients et des entreprises.

➢ **Connexion :** C'est l'autre goulot d'étranglement. Si l'utilisateur n'a pas de connexion internet, ou une connexion insuffisante, il ne pourra pas accéder à sa plateforme de travail. L'idée dans ce cas est de mettre le travail sur une application locale qui synchronise ensuite les données avec le serveur dès que l'utilisateur a à nouveau accès au réseau. Le problème de la sécurité des données en local se pose donc à nouveau.

➢ **Consommation d'électricité :** Le plus gros problème du Cloud computing c'est la consommation élevée de l'électricité et l'énergie dépensé par les Datacenters.

➢ **Localisation des données :** La dématérialisation des données sur différents sites physiques de stockage peut conduire à un éclatement des données et leurs répartitions sur différentes zones géographiques.

➢ **Budget :** Les frais de transfert peuvent s'avérer élevés, en particulier en fonction de l'usage que l'entreprise fait du Cloud. Des besoins importants en bande passante peuvent entraîner une hausse significative des coûts, voire faire exploser le budget.

1.3.3 Virtualisation

La virtualisation recouvre l'ensemble des techniques matérielles et ou logiciels qui permettent de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation, plusieurs instances différentes et cloisonnées d'un même système ou plusieurs applications, séparément les uns des autres, comme s'ils fonctionnaient sur des machines physiques distinctes. Les intérêts de la virtualisation sont : l'utilisation optimale des ressources, l'économie sur le matériel, l'allocation dynamique de la puissance de calcul, la facilité d'installation, de déploiement et la migration des machines virtuelles.

Un système hôte est installé sur un serveur physique et sert de base à la virtualisation.

Un hyperviseur permet d'exécuter plusieurs systèmes invités de manière indépendante sur une même machine. Ces systèmes fonctionnent dans des machines virtuelles (VM), qui ont un accès dédié aux ressources du serveur. Chaque VM opère isolément, garantissant une gestion optimisée des ressources.

La figure 1.3 illustre le modèle de virtualisation dans le Cloud computing :

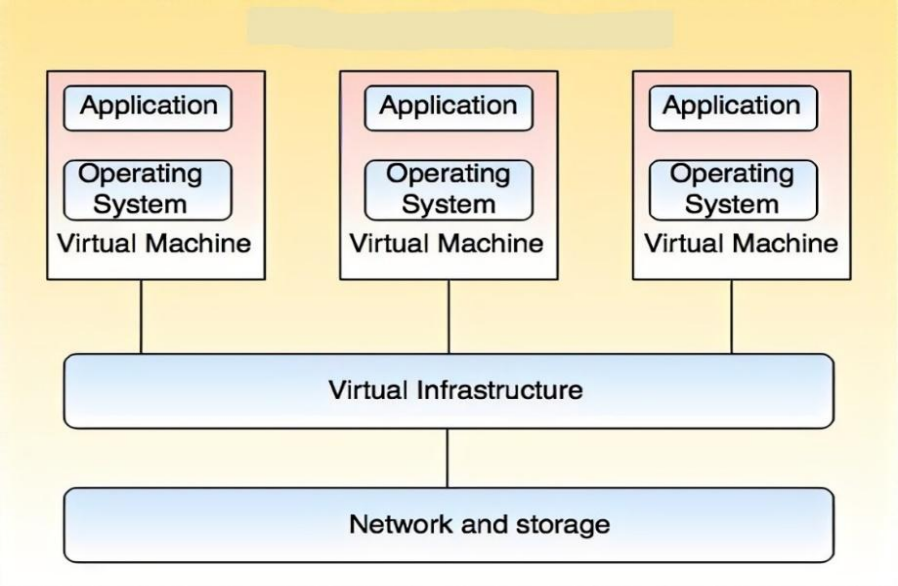

Figure 1.3: Model de nuage virtuel [3].

1.3.4 Data Center

Un Data Center, ou centre de données, est une infrastructure regroupant un ensemble de serveurs et d'équipements de stockage interconnectés. Il est conçu pour permettre aux entreprises de gérer, traiter et conserver d'importants volumes de données de manière sécurisée. Un Data Center intègre divers composants essentiels tels que des serveurs, des systèmes de stockage, des commutateurs réseau, des routeurs et des pares-feux, assurant ainsi la connectivité et la protection des données.

1.3.5 Service provider

Un fournisseur de services (Service Provider ou SP) est une entité qui met à disposition des clients divers services, tels que des ressources informatiques virtuelles, des logiciels ou d'autres solutions gérées. Ces services sont proposés selon les termes d'un accord de niveau de service (SLA) ou l'achat de ce que je cherche seulement (PAYG), défini à l'issue d'une négociation entre le fournisseur et le client, garantissant ainsi la qualité et la disponibilité des prestations fournies.

1.3.6 SLA & PAYG

1.3.6.1 Service-Level Agreement (SLA)

Le SLA est un accord au niveau de service, c'est un contrat passé entre un fournisseur de service et ses clients internes ou externes. Ce contrat documente les services que le fournisseur met à la disposition de ses clients et les différents paramètres de chaque service, comme la disponibilité, le temps de réponse…etc.

1.3.6.2 Paiement à l'utilisation

Les utilisateurs paient les ressources qu'ils utilisent en fonction de leur consommation réelle et précise, tous en suivant les modalités des services fournis.

1.3.7 Consolidation

La consolidation intervient principalement lors de fusions ou acquisitions d'entreprises. Si une entreprise acquéreuse estime que certains centres de données des sociétés rachetées sont inutiles, elle peut décider de regrouper ou réduire leur nombre. Cette démarche vise à optimiser les ressources et à réduire les coûts opérationnels.

1.3.8 Machine virtuelle

Une machine virtuelle (VM) est un environnement logiciel qui simule un matériel physique, permettant l'installation et l'exécution d'un système d'exploitation ou d'une application de manière isolée. Du point de vue de l'utilisateur, son fonctionnement est similaire à celui d'un ordinateur physique.

1.3.9 Broker

Un courtier (en anglais broker), est une personne ou une entreprise tierce qui joue le rôle d'intermédiaire entre l'acheteur d'un service de Cloud computing et les vendeurs de ce service. En général, broker peut endosser trois rôles :

• **Le négociateur :** il peut être mandaté pour négocier des contrats. Dans ce cas, il peut répartir les services entre divers fournisseurs dans un souci d'économie maximale, malgré les éventuelles complexités engendrées par les négociations impliquant plusieurs fournisseurs. Une fois la recherche terminée, le courtier présente au client une liste de fournisseurs recommandés ainsi qu'une comparaison des caractéristiques du service, des ventilations des coûts, et d'autres critères.

• **L'agrégateur de services :** il combine et intègre plusieurs services en un ou plusieurs nouveaux services. Il assure l'intégration des données et les connexions sécurisées entre son client et les multiples fournisseurs de Cloud. Il peut choisir les services de plusieurs fournisseurs, selon les critères déterminés auparavant avec son client.

• **Le facilitateur :** il peut améliorer un service donné en optimisant certaines capacités spécifiques ou en proposant des services à valeur ajoutée ou personnalisés. Cette amélioration peut concerner la gestion de l'accès aux services Cloud, le contrôle des identités, les rapports de performance, etc…

1.4 Types de Cloud computing

Comme le montre la figure 1.4, le NIST (National Institute of Standards and Technology) a défini quatre types de modèles de déploiement du Cloud [1.4] :

➢ Public;

➢ Privé;

➢ Communautaire;

➢ Hybrid.

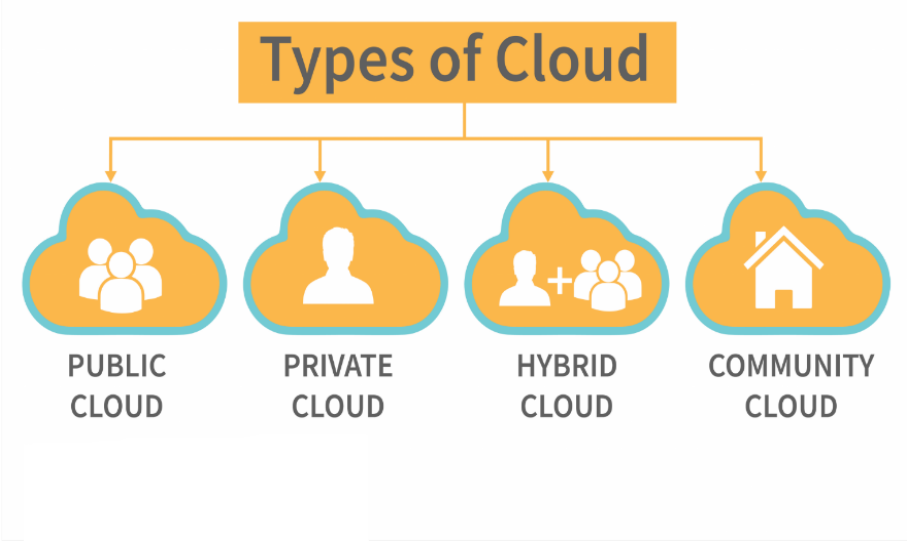

Figure 1.4: Modèle de déploiement du Cloud [4].

1.4.1 Cloud public

Dans un Cloud public, l'environnement est entièrement détenu par la société qui met à disposition ses services Cloud. Les utilisateurs et clients d'un Cloud public n'ont de droit ni sur l'infrastructure, ni sur le matériel, ni sur les logiciels, ni sur quoi que ce soit d'autre. Le fournisseur de Cloud met à disposition des utilisateurs des ressources. Il conçoit, gère, maintient et fait évoluer ces ressources en fonction des besoins des clients au fil du temps. Dans le domaine du Cloud, et du Cloud public en particulier, la sécurité est la clé de voute. L'infrastructure étant partagée avec de nombreux clients, la sécurité est de la responsabilité du fournisseur qui en assure la gestion. Ce point est d'autant plus crucial que le client n'a qu'un degré de contrôle et de surveillance très faible des aspects physiques et logiques de sécurité sur les ressources qui sont mises à sa disposition. Le fournisseur doit donc tout mettre en œuvre pour garder la confiance de ses utilisateurs (voir la figure 1.5).

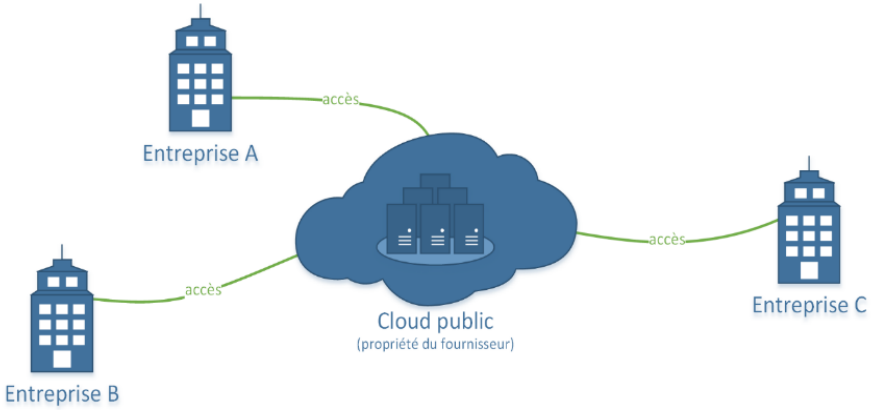

Figure 1.5: Modèle de déploiement d'un Cloud public [1].

1.4.2 Cloud privé

Le terme Cloud privé est utilisé pour décrire l'infrastructure qui émule le Cloud computing sur un réseau privé. Le Cloud privé a pour ambition d'offrir certains avantages du Cloud computing tout en en limitant ses inconvénients. Un Cloud privé est détenu par l'entreprise utilisatrice, cela nécessite d'acheter, de construire et de maintenir l'ensemble de ses constituants, ce qui implique de supporter un investissement initial très important.

Les Clouds privés diffèrent des Clouds publics en ce que les réseaux, serveurs, et infrastructures de stockage qui lui sont associés sont dédiés à une seule entreprise et ne sont pas partagés avec d'autres. Puisque le Cloud est entièrement contrôlé par l'entreprise elle-même, les risques de sécurité associés à un Cloud privé sont minimisés. Ce haut degré de contrôle et de transparence permet au propriétaire d'un Cloud privé de se conformer plus facilement à des normes, politiques de sécurités ou conformités réglementaires qui peuvent être requises dans certains domaines (voir la figure 1.6).

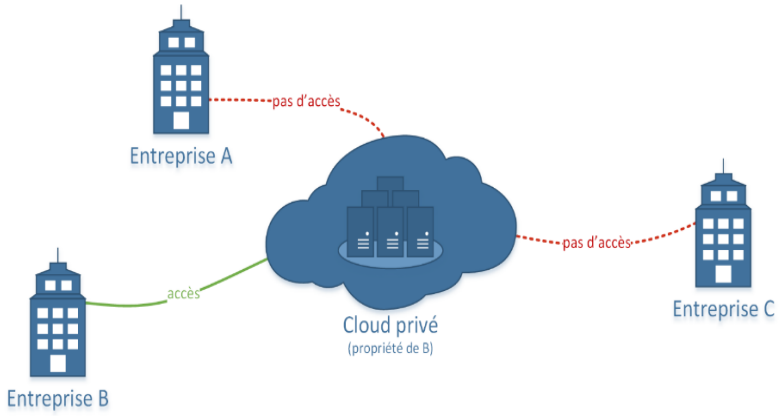

Figure 1.6: Modèle de déploiement d'un Cloud privé [1].

1.4.3 Cloud communautaire

Le Cloud communautaire est un modèle de déploiement mutualisé, partagé entre plusieurs entreprises ou organisations. Il est administré, gouverné et sécurisé soit conjointement par les participants, soit par un fournisseur de services désigné.

Un Cloud communautaire est une forme hybride de Cloud privé construit et exploité spécifiquement pour un groupe restreint et ciblé. Ces communautés ont des exigences semblables et réunissent leurs moyens humains et financiers pour atteindre leurs objectifs communs.

L'infrastructure commune est spécifiquement conçue pour répondre aux exigences d'une communauté ; à titre d'exemple, des organismes gouvernementaux, des hôpitaux ou des entreprises de télécommunication qui auraient des contraintes de réseau, de sécurité, de stockage, de calcul ou d'automatisation similaires pourraient trouver des intérêts communs à déployer collectivement un Cloud communautaire (voir la figure 1.7).

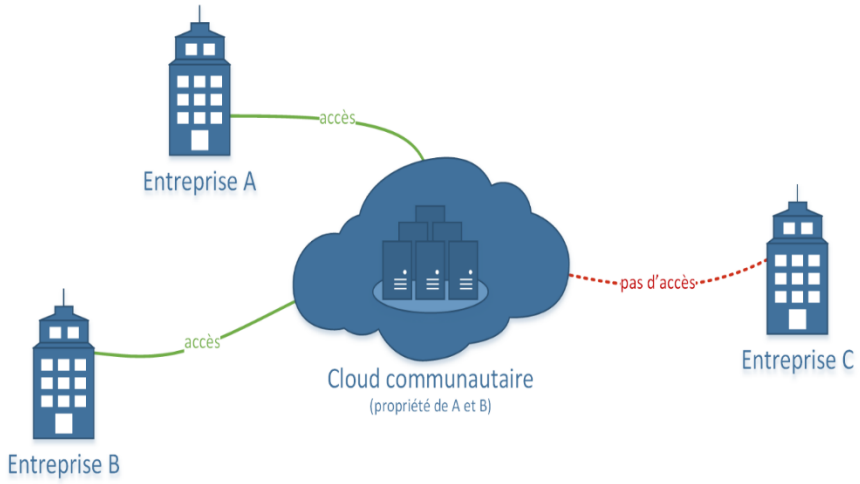

Figure 1.7: Modèle de déploiement d'un Cloud communautaire [1].

1.4.4 Cloud hybride

Comme son nom l'indique, un Cloud hybride est la combinaison de plusieurs modèles de déploiement de Clouds. Avec un Cloud hybride, une entreprise peut tirer parti de la simplicité et du faible coût d'un Cloud public — pour héberger des services classiques ne requérants pas de précautions particulières tout en créant son propre Cloud privé — pour des applications étroitement intégrées aux systèmes existants ou pour le stockage de données sensibles. Elle a également la possibilité de privilégier l'utilisation de son Cloud privé tout en gardant la possibilité de déborder sur Une offre de cloud public en cas de besoin temporaire.

Dans un Cloud hybride, les Clouds public, privé ou communautaire restent des entités uniques, mais sont reliés entre eux par une technologie normalisée ou propriétaire qui permet la portabilité des données et des applications. En raison de la complexité que la combinaison de plusieurs types de Clouds engendre, la conception, la gestion et le maintien d'un Cloud hybride peuvent être un véritable défi (voir la figure 1.8)

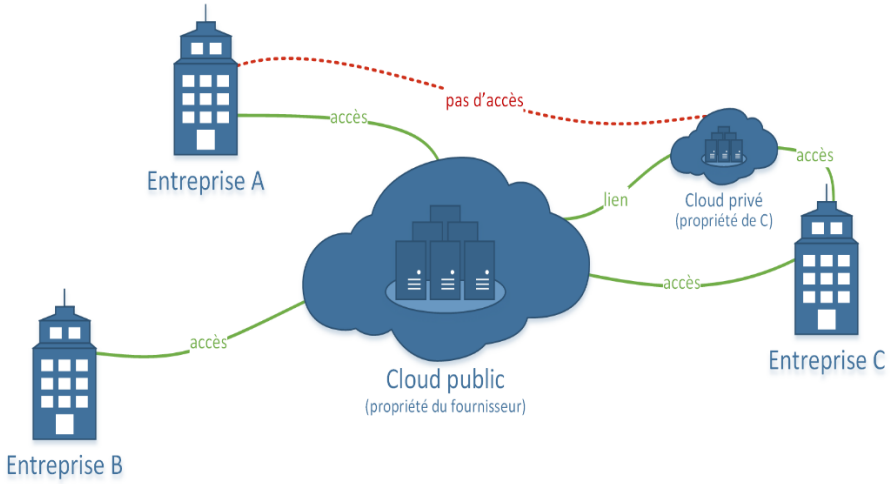

Figure 1.8: Modèle de déploiement d'un Cloud hybride [1].

1.5 Architecture de Cloud computing

1.5.1 IaaS (Infrastructure as a Services)

L'Infrastructure en tant que service (IaaS) constitue la base du Cloud computing. Elle fournit un accès à des ressources informatiques essentielles telles que le réseau, la puissance de calcul (via des machines virtuelles ou physiques) et le stockage de données. Les fournisseurs de services IaaS offrent un haut niveau de flexibilité ainsi qu'un contrôle poussé sur la gestion des ressources, ce qui en fait une solution proche des infrastructures informatiques traditionnelles, bien connues des équipes IT et des développeurs (voir figure 1.9).

1.5.1.1 Avantages de (IaaS)

- Grande flexibilité.
- Personnalisation

1.5.1.2 Inconvénients de (IaaS)

- Besoin d'administrateurs système comme pour les solutions de serveurs classiques sur site.
- Besoin de sécurité.

Exemple : hébergement de serveurs virtuels.

1.5.2 PaaS (Platform as a Service)

Grâce aux fournisseurs de plateforme en tant que service (PaaS), les entreprises n'ont plus besoin de gérer l'infrastructure sous-jacente (en règle générale, le matériel et les systèmes d'exploitation) et cette intégration vous permet de vous concentrer sur le déploiement et la gestion de vos applications. Vous êtes ainsi plus efficace, car vous n'avez pas à vous soucier de l'approvisionnement des ressources, de la planification des capacités, de la maintenance logicielle, de l'application de correctifs ou de toute autre charge indifférenciée liée à l'exécution de votre application (voir la figure 1.9).

1.5.2.1 Avantages de (PaaS)

- Le déploiement est automatisé.

• Pas de logiciel supplémentaire à acheter ou à installer.

1.5.2.2 Inconvénients de (PaaS)

- Support souvent limité à une ou quelques technologies spécifiques (par exemple, Python ou Java).
- Absence de contrôle direct sur les machines virtuelles sous-jacentes.

1.5.3 SaaS (Software as a Service)

Les fournisseurs de logiciels en tant que service (SaaS) vous proposent des applications logicielles exécutées et gérées par le fournisseur. Dans la plupart des cas, les personnes qui font référence au service SaaS pensent aux applications des utilisateurs finaux tiers. Avec une offre SaaS, vous n'avez pas à vous soucier de la gestion du service ou de celle de l'infrastructure sous-jacente. Vous devez juste réfléchir à l'utilisation de ce logiciel spécifique. Une messagerie Web dans laquelle vous pouvez envoyer et recevoir des courriels sans avoir à gérer des ajouts de fonctionnalités ni à effectuer la maintenance des serveurs et des systèmes d'exploitation sur lesquels elle s'exécute est un exemple courant d'application SaaS (voir la figure 1.9).

1.5.3.1 Avantages de (Saas)

• Plus d'installation.

• Plus de licence.

• Migration de données.

• Paiement à l'usage.

1.5.3.2 Inconvénients de (Saas)

• Limitation du logiciel proposé.

• Pas de contrôle sur le stockage et la sécurisation des données associées au logiciel.

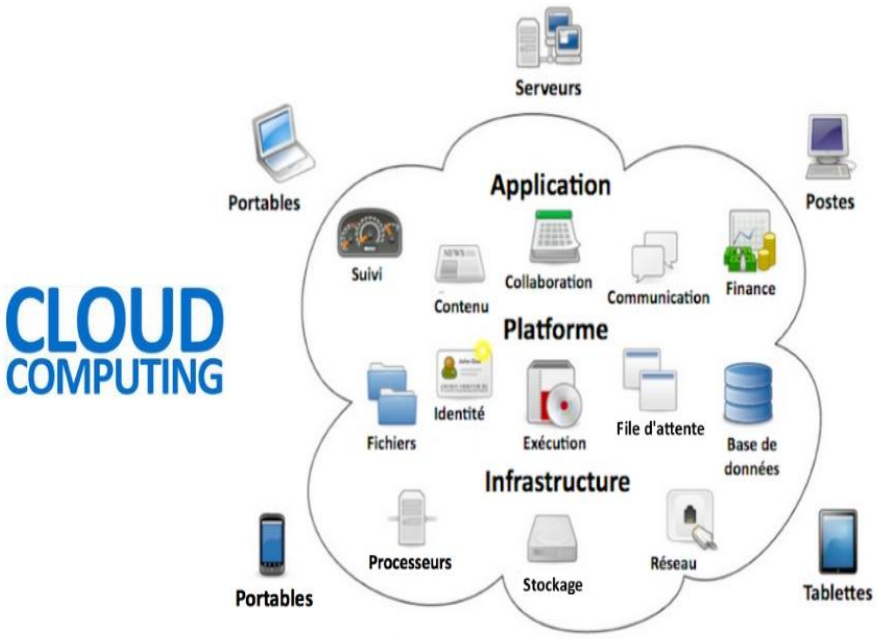

Figure 1.9: Les différentes couches des services du Cloud computing [5].

1.6 Conclusion

Dans ce chapitre, nous avons clarifié le concept de Cloud computing en mettant en lumière ses usages, ses avantages ainsi que ses inconvénients. À la base de cette technologie se trouvent de puissants serveurs, de plus en plus nombreux, hébergés dans des centres de données. Ces serveurs exécutent des logiciels, stockent et diffusent des données, rendant ainsi le Cloud indispensable pour les grandes entreprises. Cette analyse nous a amenés à nous interroger sur la consommation énergétique et les coûts associés à l'utilisation du Cloud computing. Le chapitre suivant porte sur les métaheuristiques d'optimisation, avec un focus particulier sur l'algorithme WSO.

Chapitre 2 : Les métaheuristiques

2.1 Introduction

Les métaheuristiques sont des algorithmes puissants conçus pour trouver des solutions satisfaisantes à des problèmes d'optimisation complexes, en particulier lorsque les méthodes traditionnelles se révèlent inefficaces. Elles sont spécialement adaptées aux problèmes difficiles, voire impossibles à résoudre de manière exacte. Elles s'inspirent généralement de phénomènes naturels, comme l'évolution biologique ou les comportements collectifs observés chez les animaux, qu'elles modélisent pour guider la recherche de solutions optimales. Parmi les métaheuristiques utilisées pour résoudre les problèmes d'ordonnancement de tâches dans le Cloud computing, le White Shark Optimiser (WSO) a été choisi pour être étudié et testé dans ce travail.

Ce présent chapitre sera consacré à la présentation de cet algorithme, en mettant en avant ses différentes améliorations ainsi que les développements qui en découlent.

2.2 Les méthodes de résolution

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

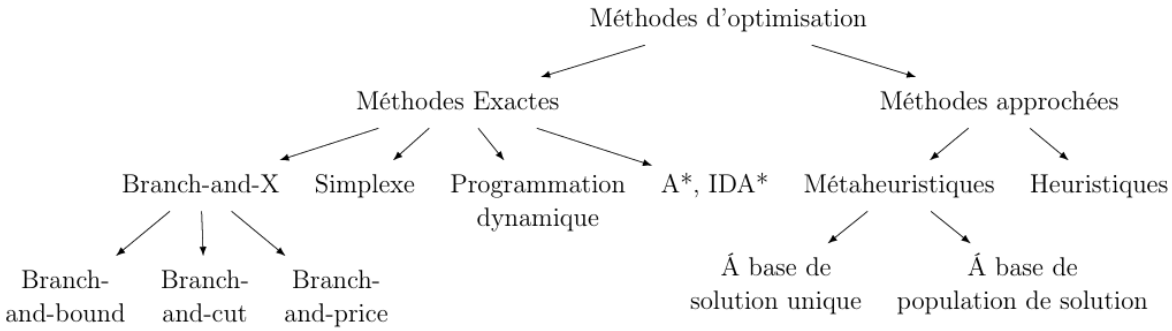

Figure 2.1: Classification des méthodes de résolution de problèmes d'optimisation [16] .

2.2.1 Les méthodes exacts

Les méthodes exactes garantissent l'obtention d'une solution optimale en explorant de manière exhaustive l'ensemble des possibilités. Bien qu'efficaces pour les problèmes de petite taille, leur performance diminue rapidement à mesure que la complexité du problème augmente, en raison de leur nature exponentielle. Ainsi, malgré leur capacité à fournir des solutions optimales, ces méthodes sont principalement utilisées comme référence pour évaluer la qualité des résultats obtenus par des approches heuristiques ou métaheuristiques [17].

2.2.2 Les méthodes approchées

Les méthodes approchées sont souvent utilisées pour résoudre des problèmes plus complexes ou de taille plus grande que ceux traités par les méthodes exactes. Les méthodes approchées ne garantissent pas de solutions optimales comme le cas des méthodes exactes, mais plutôt souvent proches de l'optimum [18].

2.2.2.1 Principale caractéristique des méthodes approchées

Facilité : Les méthodes approchées sont généralement simples à comprendre et à appliquer.

Efficacité : Elles cherchent à fournir des solutions dans un laps de temps raisonnable, même pour des problèmes complexes.

2.2.2.2 Heuristique

En optimisation combinatoire, une heuristique est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale. L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte. Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre sans offrir aucune garantit quant à la qualité de la solution calculée. Les heuristiques peuvent être classées en deux catégories :

- Méthodes constructives : Qui génèrent des solutions à partir d'une solution initiale en essayant d'en ajouter petit à petit des éléments jusqu'à ce qu'une solution complète soit obtenue.
- Méthodes de fouilles locales : Qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière répétitive essaie d'améliorer cette solution en explorant son voisinage [19].

2.2.2.3 Méta-heuristique

Les métaheuristiques sont généralement des algorithmes itératifs et stochastiques qui s'approchent progressivement d'un optimum global en explorant l'espace des solutions à l'aide de l'échantillonnage d'une fonction objectif. Elles fonctionnent comme des algorithmes de recherche adaptatifs, cherchant à identifier les caractéristiques du problème afin d'en approcher la meilleure solution possible, à la manière des algorithmes d'approximation [6].

- Métaheuristiques à base de solution unique

Les métaheuristiques à solution unique peuvent être appelées aussi méthodes de recherche locale ou bien méthodes de trajectoire. Leur mécanisme consiste à faire évoluer itérativement une solution dans l'espace de recherche afin de se diriger vers l'optimum global. Nous citons par exemple : la méthode de descente, le recuit simulé et la recherche tabou [20].

- Métaheuristiques à base de population de solutions

Les métaheuristiques à population de solutions, contrairement aux méthodes à solution unique, font évoluer simultanément un ensemble de solutions dans l'espace de recherche. Il y a d'ailleurs souvent une interaction entre les solutions qu'elle soit directe ou indirecte afin de faire évoluer la population. Deux grandes classes de métaheuristiques à population de solutions sont distinguables : les algorithmes évolutionnaires inspirés de la théorie de l'évolution de Darwin, et les algorithmes d'intelligence en essaim inspirés de biologie ou de l'éthologie. Dans ce qui suit, nous nous intéressons à cette dernière catégorie en présentant trois méthodes : l'optimisation par l'algorithme de Grey Wolf Optimiser, l'optimisation par essaims Salp Swarm Optimiser, l'optimisation par l'algorithme de White Shark Optimiser [11].

2.3 Quelque algorithmes basés sur l'intelligence en essaim

2.3.1 Grey Wolf optimizer

L'algorithme Grey Wolf Optimizer (GWO) est inspiré du comportement social et de chasse du loup gris (Canis lupus) [13]. Ce dernier et en tant que prédateur de premier ordre, occupe une position dominante au sommet de la chaîne alimentaire. Les loups gris vivent généralement en meute,

composée en moyenne de 5 à 12 individus. Il est particulièrement intéressant de souligner qu'ils adoptent une hiérarchie sociale très stricte, comme l'illustre la figure 2.2.

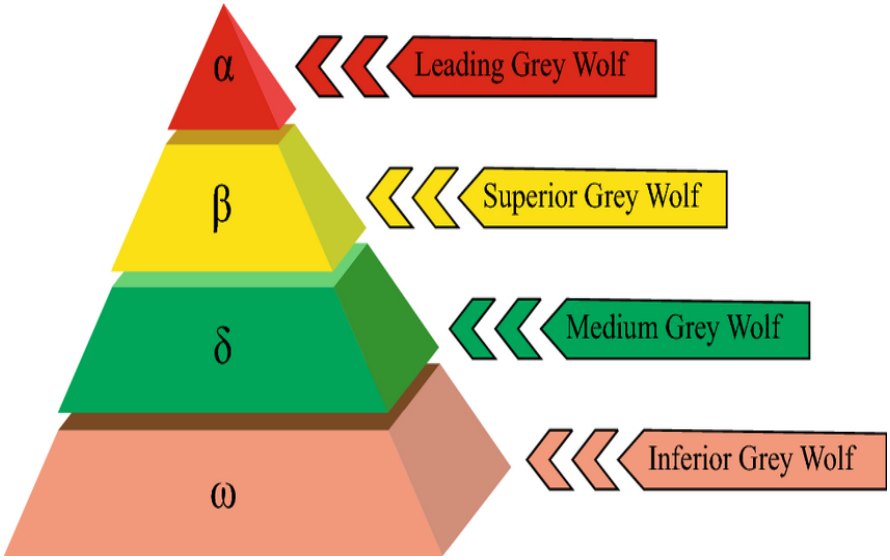

Figure 2.2: Hiérarchie du loup gris (la dominance décroît de haut en bas) [22].

Les chefs sont un mâle et une femelle, appelés alphas. L'alpha est principalement responsable des décisions concernant la chasse, le lieu de couchage, l'heure du réveil, etc. Les décisions de l'alpha sont dictées à la meute. Cependant, une sorte de comportement démocratique a également été observé, dans lequel un alpha suit les autres loups de la meute. Lors des rassemblements, toute la meute salue l'alpha en tenant sa queue baissée. Le loup alpha est également appelé loup dominant car ses ordres doivent être suivis par la meute. Les loups alpha ne sont autorisés à s'accoupler qu'au sein de la meute. Il est intéressant de noter que l'alpha n'est pas nécessairement le membre le plus fort de la meute, mais le meilleur en termes de gestion de la meute. Cela montre que l'organisation et la discipline d'une meute sont bien plus importantes que sa force.

Le deuxième niveau de la hiérarchie des loups gris est celui des bêtas. Les bêtas sont des loups subordonnés qui aident l'alpha dans la prise de décision ou dans d'autres activités de la meute. Le loup bêta peut être un mâle ou une femelle, et il/elle est probablement le meilleur candidat pour être l'alpha au cas où l'un des loups alpha mourrait ou deviendrait très vieux. Le loup bêta doit respecter l'alpha, mais commande également les autres loups de niveau inférieur. Le bêta renforce les ordres de l'alpha dans toute la meute et lui donne des retours d'information.

Le loup gris le moins bien classé est l'oméga. L'oméga joue le rôle de bouc émissaire. Les loups oméga doivent toujours se soumettre à tous les autres loups dominants. Ils sont les derniers loups autorisés à manger. Il peut sembler que l'oméga ne soit pas un individu important dans la meute, mais il a été observé que toute la meute est confrontée à des conflits internes et à des problèmes en cas de perte de l'oméga. Cela est dû à l'évacuation de la violence et de la frustration de tous les loups par l'oméga(s).

Cela contribue à satisfaire l'ensemble de la meute et à maintenir la structure de dominance. Dans certains cas, l'oméga est également le baby-sitter de la meute. Si un loup n'est pas un alpha, un bêta ou un oméga, il est appelé subordonné (ou delta dans certaines références). Les loups delta doivent se soumettre aux alphas et aux bêtas, mais ils dominent l'oméga.

Les éclaireurs, les sentinelles, les anciens, les chasseurs et les gardiens appartiennent à cette catégorie. Les éclaireurs sont chargés de surveiller les limites du territoire et d'avertir la meute en cas de danger. Les sentinelles protègent et garantissent la sécurité de la meute. Les anciens sont les loups expérimentés qui ont été alpha ou bêta. Les chasseurs aident les alphas et les bêtas à chasser leurs proies et à fournir de la nourriture à la meute. Enfin, les gardiens sont chargés de prendre soin des loups faibles, malades et blessés de la meute (voir la figure 2.3).

Figure 2.3: Comportement de chasse des loups gris [23].

A : Une meute de loups avance en groupe. Ils suivent les traces dans la neige et traquent une proie potentielle.

B : Les loups ont trouvé un bison isolé et commencent à l'encercler. Ils essaient de l'intimider et d'identifier ses points faibles.

C : Le bison est acculé et les loups commencent à le harceler. Ils testent ses réactions et cherchent à l'épuiser.

D : Les loups attaquent plus directement, mordant le bison pour le blesser. Ils essaient de le fatiguer et de l'empêcher de s'enfuir.

E : Le bison est soit mort, soit a réussi à échapper aux loups. Les loups se dispersent, probablement après avoir consommé leur proie ou après un échec.

2.3.2 Salp Swarm Algorithm

Le SSA est un optimiseur inspiré du comportement en essaim des salpes [24]. Il met à jour progressivement les positions des agents en tenant compte des interactions dans une foule dynamique. Son unique paramètre adaptatif permet de trouver un équilibre entre diversification et intensification de la recherche. Cette dynamique aide à éviter la convergence prématurée vers des optima locaux. Le SSA se distingue par sa simplicité, sa flexibilité et son implémentation facile en mode parallèle ou série.

Il utilise la salpe élite pour guider l'ensemble de l'essaim vers de meilleures zones de l'espace de recherche (voir la figure 2.4).

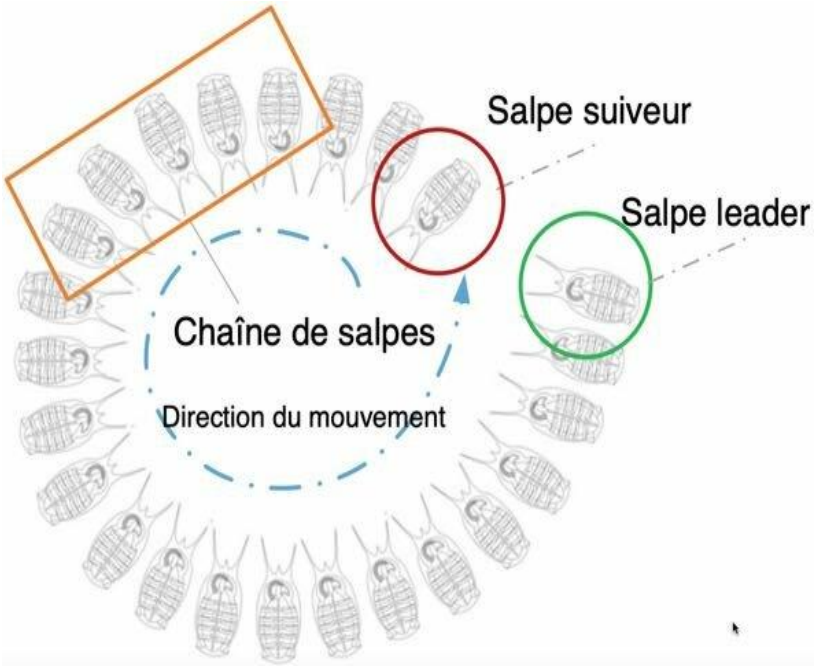

Figure 2.4: Illustration de la chaîne de salpe et du concept de leader et de suiveur [24].

A. **Mise à jour des vecteurs de localisation :** Chaque salpe met à jour sa position, différenciant le rôle du chef de chaîne et celui des suiveurs.

B. **Action du leader :** La salpe leader se déplace en direction d'une source de nourriture (F).

C. **Action des suiveurs :** Les suiveurs se déplacent vers les autres salpes, en suivant soit directement le leader, soit indirectement les autres membres de la chaîne.

2.3.3 White-Shark Optimizer

2.3.3.1 Inspiration de WSO

WSO est un algorithme métaheuristique inspiré du comportement du requin blanc, récemment proposé dans [25].

Les requins blancs sont des prédateurs parfaitement adaptés et des chasseurs impressionnants, possédant des muscles puissants, une excellente vision et un odorat développé. Leur proie comprend de nombreux organismes marins et non marins, tels que les crustacés, les invertébrés, les mammifères, les amphibiens et les oiseaux marins. Ils chassent généralement leurs proies par embuscade, un mode opératoire dans lequel un requin cherche à prendre sa cible par surprise en lui infligeant une morsure massive et létale. Les aspects bien plus fascinants du comportement collectif des grands requins résident dans leurs capacités uniques à capturer des proies grâce à leur nage, ainsi que dans leurs sens inhabituels permettant de détecter les odeurs des proies et d'entendre (voir la figure 2.5).

Figure 2.5: Great-white-shark-fish [7].

2.3.3.2 Description de l'algorithme WSO

Le grand requin blanc (Carcharodon carcharias), avec son allure svelte et ses dents de 6 centimètres de long, spécialement conçues pour déchiqueter, est un prédateur intimidant. Si intimidant que, au premier abord, sa disparition pourrait presque passer pour une bonne nouvelle. Cependant, en réalité, la perte d'un animal aussi important, connu comme le plus grand requin prédateur du monde, aurait des répercussions qui se feraient sentir à travers la totalité de l'écosystème. Une nouvelle étude, publiée dans Frontier in Marine Science, s'attèle à démontrer les conséquences de la disparition des requins.

Elle analyse pour cela le cas de Seal Island, une petite île au large de la baie False, en Afrique du Sud, qui constituait autrefois l'une des rares destinations sur Terre où l'on pouvait observer de grands requins blancs surgir de l'eau pour capturer leurs proies. [8]

2.3.3.3 Suivies des proies

Comme tout autre organisme dans la nature, les requins parcourent les océans à la recherche de proies et ajustent leur position en conséquence. Ils utilisent pratiquement tous les outils à leur disposition pour suivre, traquer et localiser leurs victimes dans ce contexte. Ils possèdent une variété de sens qui sont intégrés et complémentaires, comme illustré dans la figure 2.6.

Tout d'abord, les requins disposent d'une capacité auditive étonnamment développée, qu'ils utilisent pour explorer une vaste zone lorsqu'ils chassent des proies. Ensuite, ils ont un odorat très aigu.

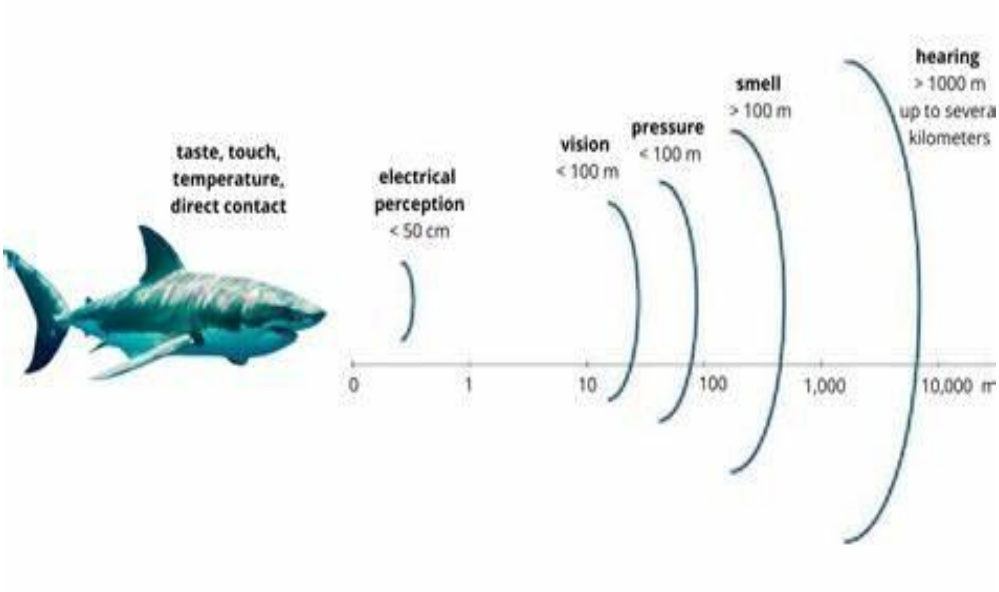

Figure 2.6: Les sens du requin blanc : l'odorat, la vue et l'ouïe [9].

2.3.3.4 Phase d'exploration (recherche de proie)

Lorsqu'ils recherchent une proie, les grands requins utilisent un sens de l'ouïe inhabituel pour explorer le champ de recherche. Ils peuvent entendre sur toute la longueur de leur corps, ce qui est illustré dans la figure 2.7.

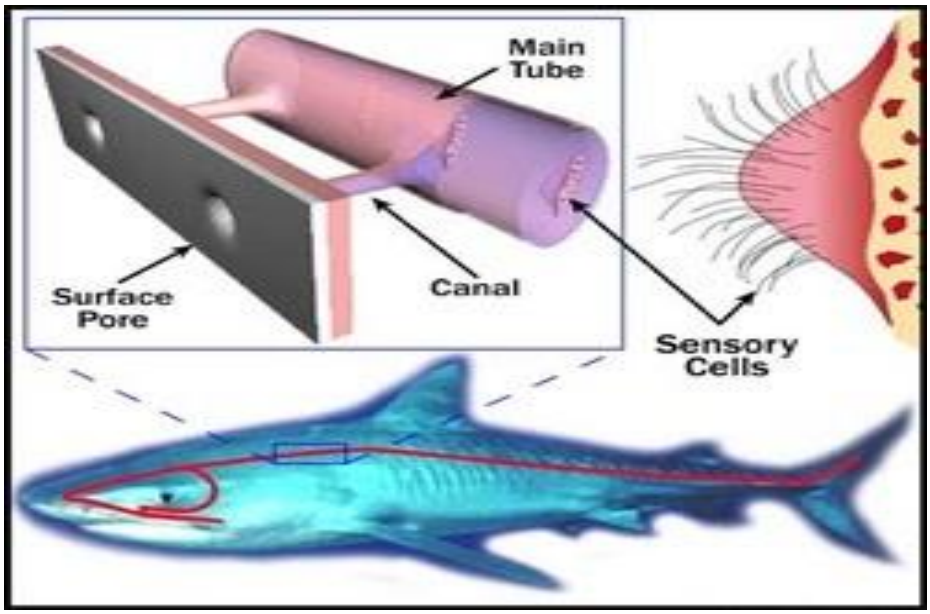

Figure 2.7: Un grand requin blanc avec un capteur de ligne auditive montré sur son torse [10].

Deux lignes de chaque côté de leur corps. Les changements de pression de l'eau peuvent être détectés par ces deux lignes, suggérant les mouvements de la proie. Les requins blancs seront incités à s'approcher d'une proie turbulente par les fluctuations de la pression de l'eau libérée par la proie. De plus, il possède des organes qui peuvent détecter les minuscules champs électromagnétiques créés par le mouvement de la proie. Le requin peut alors détecter avec précision la position de la proie et sa taille en fonction de la fréquence des ondes qui dérivent vers lui pendant la mobilité de la

proie et ses turbulences. Chaque fois que le requin s'approche dangereusement de sa proie au point de pouvoir détecter les champs électromagnétiques, il avance vers elle de manière ondulante.

Voici l'expression mathématique qui peut être utilisée pour décrire la vitesse ondulante des requins :

$$\boldsymbol{v} = \boldsymbol{x} \times \boldsymbol{f} \quad \textbf{(1)}$$

Là où v est la vitesse du mouvement ondulatoire, x est la longueur d'onde, qui définit la distance qu'un requin doit parcourir lors d'un mouvement ondulatoire pour accomplir une révolution complète, et f est la fréquence de l'onde du mouvement ondulatoire, qui est calculée en fonction du nombre de révolutions (c'est-à-dire cycles) accomplies par seconde par le requin, où Hertz (Hz) représente le cycle par seconde.

2.3.3.5 Phase d'exploitation (Affiner autour de la proie)

Les requins utilisent leur extraordinaire sens de "l'odorat" pour exploiter toutes les zones disponibles dans l'espace de recherche à la recherche de proies. Leur sens de l'odorat entre en jeu dès qu'ils s'approchent d'une proie. Lorsque les grands requins arrivent près de leur proie, leur sens de l'odorat peut augmenter exponentiellement jusqu'à ce qu'ils localisent précisément l'emplacement de la proie.

La suivante expression cinématique avec une accélération continue peut être utilisée pour mettre à jour la position des requins lorsqu'ils s'approchent de la proie :

$$\boldsymbol{x} = \boldsymbol{x}_{\boldsymbol{i}} + \boldsymbol{v}_{\boldsymbol{i}} \times \boldsymbol{\Delta pt} + \frac{1}{2}\boldsymbol{a}(\boldsymbol{\Delta pt})^2 \quad \textbf{(2)}$$

Où la nouvelle position du requin est indiquée par la lettre x, la position initiale est représentée par xi, l'intervalle de temps entre la position de départ et la position actuelle est représenté par Δpt, et le facteur d'accélération constant est noté par a. Dans de nombreuses situations, les proies, comme les phoques, laissent leurs odeurs derrière elles lorsqu'elles quittent leur emplacement, de sorte que les requins ne trouvent aucune proie lorsqu'ils sont proches de cette odeur. Dans cette situation, les requins doivent utiliser activement leurs sens de l'odorat, de l'ouïe et de la vue pour chercher dans les régions adjacentes et explorer d'autres endroits dans l'espace de recherche de manière aléatoire.

2.3.3.6 Mécanisme d'optimisation

Pour localiser les positions des proies, les requins adoptent trois comportements différents :

-Le mouvement vers la proie dépend des hésitations causées par les mouvements ondulatoires de la proie ; le requin navigue vers la proie en utilisant ses sens liés à l'ouïe et à l'odorat de manière ondulatoire.

-La recherche aléatoire de proies dans les profondeurs de l'océan, où les requins se déplacent vers la proie tout en restant proches de la proie optimale.

-Le comportement d'un requin lorsqu'il cherche une proie dans une zone donnée. Dans ce cas, le requin imite le comportement d'un banc de poissons en se déplaçant vers le meilleur requin qui est extrêmement proche de la meilleure proie. Sur la base de ces comportements, tous les emplacements des requins seront mis à jour avec les meilleures solutions idéales au cas où la proie ne serait pas identifiée à temps. Ces comportements sont modélisés mathématiquement comme suit :

- **Initialisation de WSO**

Lors du démarrage du processus d'optimisation pour résoudre un problème, WSO produit un ensemble de solutions initiales aléatoires car il s'agit d'une méthode basée sur la population. La matrice 2D suivante présente une population de N requins (c'est-à-dire la taille de la population) dans l'espace de recherche à deux dimensions (c'est-à-dire la dimension du problème), où la position de chaque requin représente une solution potentielle à un problème :

$$F = \begin{bmatrix} \omega_1^1 & \omega_2^1 & \omega_3^1 & \cdots & \omega_d^1 \\ \omega_1^2 & \omega_2^2 & \omega_3^2 & & \omega_d^2 \\ & \vdots & & \ddots & \vdots \\ \omega_1^N & \omega_2^N & \omega_3^N & \cdots & \omega_d^N \end{bmatrix} \quad \textbf{(3)}$$

Où tous les requins dans la zone de recherche sont représentés par la lettre ω, le nombre de variables de décision pour un problème particulier est noté par d, et le i-ème Localisation du requin blanc dans le d-ème la dimension est indiquée par ω_d^i . Une initialisation aléatoire uniforme est utilisée pour établir la population initiale, comme suit :

$$\boldsymbol{\omega_j^i = l_j + r \times (u_j - l_j)} \quad \textbf{(4)}$$

Où ω_j^i est le vecteur de départ du i-ème requin blanc dans la j-ème dimension, les bornes supérieure et inférieure dans la j-ème dimension sont représentées respectivement par u_j et l_j, et r est un nombre aléatoire généré entre 0 et 1.

- **Vitesse de déplacement vers la proie**

Lorsqu'un requin détecte la position d'une proie en fonction de la pause dans les vagues qu'il entend pendant que la proie se déplace, il se déplace dans un mouvement ondulatoire qui peut être caractérisé comme indiqué dans l'équation suivante :

$$v_{k+1}^i = \mu\left[v_k^i + p_1(\omega_{gbestk} - \omega_k^i) \times c_1 + p_2(\omega_{best}^{v^i k} - \omega_k^i) \times c_2\right] \quad \textbf{(5)}$$

Là où le vecteur de vitesse mis à jour du i-ème requin à l'étape (k + 1) est noté v_{k+1}^i , v_k^i spécifie le vecteur de vitesse actuel du i-ème requin, ω_{gbestk} désigne le meilleur vecteur de position globale atteint par n'importe quel requin jusqu'à présent, ω_k^i est le vecteur de position actuel du i-ème requin à l'étape k, $\omega_{best}^{v^i k}$ est le i-ème meilleur vecteur de position connu du essaim, v^i est le vecteur d'index du i-ème requin atteignant la meilleure position comme défini dans l'équation (3), deux nombres aléatoires uniformément générés entre 0 et 1 sont c1 et c2, les forces des requins qui influencent l'impact de ω_{gbestk} et $\omega_{best}^{v^i}$ sur ω_k^i sont représentées par p1 et p2, qui sont calculées en utilisant les équations (7) et (8) et μ est le facteur de contraction pour évaluer le comportement de convergence des requins, et il est calculé à l'aide de l'équation (9).

$$\boldsymbol{v = [n \times rand(1, n)] + 1} \quad \textbf{(6)}$$

Où $\boldsymbol{rand(1, n)}$ est un vecteur de nombres générés aléatoirement avec une distribution uniforme entre 0 et 1.

$$\boldsymbol{p_1 = p_{max} + (p_{max} - p_{min}) \times e^{-(\frac{4k}{K})^2}} \quad \textbf{(7)}$$

$$\boldsymbol{p_2 = p_{min} + (p_{max} - p_{min}) \times e^{-(\frac{4k}{K})^2}} \quad \textbf{(8)}$$

Où k et K représentent respectivement le nombre actuel et le nombre maximal de répétitions, tandis que p_{min} et p_{max} sont les vitesses initiale et subordonnée permettant une bonne mobilité pour les requins.

Après une analyse approfondie, il a été constaté que les valeurs de p_{min} et p_{max} sont respectivement de 0,5 et 1,5.

$$\mu = \frac{2}{\left|2-\tau-\sqrt{\tau^2-4\tau}\right|} \quad \textbf{(9)}$$

Où τ représente le coefficient d'accélération, qui est égal à 4,125, et ce chiffre a été déterminé après de nombreuses recherches.

- **Déplacement vers la proie optimale**

L'odeur de la proie reste souvent dans le lieu, même si le requin peut encore la sentir. Dans ce scénario, le requin se déplace vers des sites aléatoires à la recherche de la proie, de manière similaire à la façon dont un banc de poissons cherche de la nourriture. Le mécanisme de mise à jour de position décrit par l'équation (10) a été utilisé pour caractériser le comportement des requins lorsqu'ils s'approchent de la proie dans cette situation.

$$\boldsymbol{\omega_{k+1}^i} = \begin{cases} \boldsymbol{\omega_k^i} \cdot \neg \oplus \boldsymbol{\omega_0} + u.a + l.b & rand < mv \\ \boldsymbol{\omega_k^i} + \boldsymbol{v_k^i}/\boldsymbol{f} & rand < mv \end{cases} \quad \textbf{(10)}$$

Où le vecteur de position mis à jour du i-ème requin à l'étape d'itération (k + 1) est noté par $\boldsymbol{\omega}_{k+1}^{i}$, un opérateur de négation est représenté par q, les équations (11) et (12) définissent a et b comme des vecteurs binaires unidimensionnels, respectivement, les limites supérieure et inférieure de l'espace de recherche sont notées par u et l, respectivement. ω_0 , est un vecteur logique défini tel que vu dans l'équation (13), f représente la fréquence du mouvement ondulatoire d'un requin, donnée dans l'équation (14), un nombre aléatoire dans l'intervalle [0,1] est noté par $rand$, et mv symbolise la force de mouvement qui augmente avec le nombre de tours lorsque le grand requin blanc atteint sa proie, telle que donnée dans l'équation (15).

$$\boldsymbol{a = sgn\left(\omega_k^i - u\right) > 0} \quad \textbf{(11)}$$

$$\boldsymbol{b = sgn\left(\omega_k^i - l\right) < 0} \quad \textbf{(12)}$$

$$\boldsymbol{\omega_0 = \oplus\,(a, b)} \quad \textbf{(13)}$$

$$\boldsymbol{f = f_{min} + \frac{f_{max} - f_{min}}{f_{max} + f_{min}}} \quad \textbf{(14)}$$

$$\boldsymbol{mv = \frac{1}{(a_0 + e^{(K/2-k)/a_1})}} \quad \textbf{(15)}$$

La fonction **sgn** est utilisée pour vérifier si la différence entre chaque élément du vecteur et les limites est positive ou négative.

La fonction renvoie 1 si l'entrée est positive, −1 si elle est négative, et 0 lorsque l'entrée est nulle.

- **Mouvement vers le meilleur grand requin blanc**

Les requins peuvent maintenir leur position en avant de l'optimum, qui est proche de la proie. L'équation (16) montre comment ce phénomène est exprimé.

$$\boldsymbol{\omega}_{k+1}^{\prime i} = \boldsymbol{\omega}_{gbestk} + \boldsymbol{r}_1 \times \overrightarrow{\boldsymbol{D}_{\boldsymbol{\omega}}} \times \boldsymbol{sgn}(\boldsymbol{r}_2 - 0.5)\, \boldsymbol{r}_3 < \boldsymbol{S}_s \quad \textbf{(16)}$$

Où $\boldsymbol{\omega}_{k+1}^{\prime i}$ représente la position mise à jour du i-ème requin par rapport à la position de la proie, $\boldsymbol{sgn}(r_2 - 0.5)$ renvoie soit -1, soit 1 pour contrôler la direction de recherche, les coefficients r1, r2 et r3 sont tous des nombres aléatoires compris entre 0 et 1. $\overrightarrow{\boldsymbol{D}_{\boldsymbol{\omega}}}$ est la distance entre la proie et le requin, telle que décrite dans l'équation (17), et $\boldsymbol{S}_s$ est un coefficient proposé pour exprimer la force des sens de la vue et de l'odorat des requins lorsqu'ils poursuivent d'autres requins proches de la meilleure proie, tel que présenté dans l'équation (18).

$$\overrightarrow{\boldsymbol{D}_{\boldsymbol{\omega}}} = \left|\boldsymbol{rand} \times (\boldsymbol{\omega}_{gbest} - \boldsymbol{\omega}_{k}^{i})\right| \quad \textbf{(17)}$$

$$\boldsymbol{S}_s = \left|\boldsymbol{1} - \boldsymbol{e}^{(-a_2 \times k/K)}\right| \quad \textbf{(18)}$$

$\boldsymbol{a_2}$ est une constante positive, fixée à 0.0005, qui contrôle l'équilibre entre exploration et exploitation dans l'algorithme.

2.3.3.7 Pseudo codes de l'algorithme d'optimisation White-Shark

La figure 2.8 présente le pseudocode détaillé de l'algorithme White Shark Optimizer (WSO), illustrant l'ensemble des étapes nécessaires à son fonctionnement.

Define the problem parameters.
Define the parameters for the WSO.
Generate the initial positions of the WSO randomly.
Set the initial population's velocity.
Evaluate the initial population's positions
while $t > T$ **do**
    Sequentially revise the parameters v, p_1, p_2, μ, a, b, w_o, f, mv, and S_s, respectively.
    for $i = 1, \ldots, n$ **do**
        $v_{i_{t+1}} = \mu \left[v_{i_t} + p_1 \left(w_{g_{\text{best}_t}} - w_{i_t} \right) \times c1 + p_2 \left(w_{v_{i_t}^{\text{best}}} - w_{i_t} \right) \times c2 \right]$
    end for
    for $i = 1, \ldots, n$ **do**
        if $rand < mv$ **then**
            $w_{i_{t+1}} = w_{i_t} \cdot \neg \oplus w_o + u \cdot a + l \cdot b$
        else
            $w_{i_{t+1}} = w_{i_t} + \frac{v_{i_t}}{f}$
        end if
    end for
    for $i = 1 \ldots, n$ **do**
        if $rand \leq S_s$ **then**
            $\vec{D_w} = |rand \times (w_{g_{best_t}} - w_{i_t})|$
            if $i == 1$ **then**
                $w_{t+1}^i = w_{g_{\text{best}_t}} + r_1 \vec{D}_w \cdot \text{sgn}(r_2 - 0.5)$
            else
                $\vec{w}_{t+1}^i = w_{g_{\text{best}_t}} + r_1 \vec{D}_w \cdot \text{sgn}(r_2 - 0.5)$
                $w_{t+1}^i = \frac{w_{i_t} + \vec{w}_{t+1}^i}{2 \times \text{rand}}$
            end if
        end if
    end for
    Adjust the white sharks' positions that cross the boundary limits
    Evaluate and adjust the updated positions
end while
Return the best solution obtained up to now

Figure 2.8: Pseudo code de WSO [26].

2.4 Conclusion

Dans ce chapitre, nous avons exploré les approches de résolution de problèmes d'optimisation, en distinguant les méthodes exactes et les méthodes approchées, avec un accent particulier sur les métaheuristiques. Nous avons présenté en détail la métaheuristique White Shark Optimizer (WSO), inspirée des comportements de chasse et des structures sociales des grands requins blancs. Ses mécanismes clés, tels que l'encerclement dynamique, l'exploration adaptative et l'exploitation hiérarchique, ont été analysés, accompagnés de son pseudocode pour une compréhension opérationnelle. Contrairement aux approches mono-objectifs traditionnelles, nous avons souligné l'adaptabilité du WSO aux problématiques multi-objectif, notamment dans l'optimisation simultanée du makespan, du coût et de l'énergie, des critères critiques dans les systèmes de Cloud computing. Le chapitre suivant se concentrera sur l'application du WSO pour l'ordonnancement de tâches dans le Cloud computing, en évaluant ses performances face à des algorithmes classiques et en analysant son efficacité dans l'équilibre entre les objectifs.

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

3.1 Introduction

Le problème d'ordonnancement des tâches dans le Cloud computing est caractérisé par le fait qu'il comporte des objectifs complexes et souvent conflictuels. Il est considéré comme un problème d'optimisation multi-objectif, dont la complexité augmente avec l'augmentation des contraintes et les paramètres considérés. Dans ce contexte, trouver un bon compromis entre les objectifs reste un défi majeur.

En général, le problème d'ordonnancement est connu pour être NP-complet, où il est impossible de trouver la solution optimale en utilisant des algorithmes classiques. Par conséquent, ce problème a un enjeu majeur nécessitant de trouver une manière plus efficace pour évaluer tous ces différents objectifs en un temps raisonnable, d'où la nécessité d'utiliser les métaheuristiques. Dans ce contexte, l'objectif de l'ordonnancement consiste à une répartition des tâches sur des ressources délivrées par le fournisseur de services Cloud de façon à optimiser plusieurs métriques de qualité de services (QoS). Notons que la plupart des travaux se concentrent souvent sur l'optimisation d'une seule métrique de QoS, comme le *makespan.*

L'objectif de ce chapitre est de décrire en détail notre contribution dans le cadre de ce projet de fin d'études. Cette contribution consiste à utiliser un algorithme basé sur la métaheuristique White Shark Optimizer (WSO) pour l'ordonnancement des tâches multi-objectif dans le Cloud computing, en optimisant simultanément trois critères :

- Makespan (temps d'exécution total),
- Coût d'utilisation des ressources,
- Energie (ou efficacité énergétique).

Nous présentons également une évaluation mono-objectif de l'algorithme WSO à travers une comparaison avec une heuristique classique qui est le Round Robin.

Pour renforcer l'analyse, nous comparons également les performances du WSO avec deux métaheuristiques récentes :

- Salp Swarm Algorithm (SSA) (basé sur le comportement des salpes en essaim),
- Grey Wolf Optimizer (GWO) (mimant la hiérarchie sociale des loups gris).

Dans le reste de ce chapitre, nous commençons par présenter les outils de simulation utilisés dans le cadre de ce PFE : Le langage Java, L'IDE NetBeans, l'outil JFreeChart pour la visualisation des données et le simulateur CloudSim (architecture, classes fondamentales, et extensions personnalisées).

Nous abordons ensuite les principes de base de l'optimisation multi-objectif, notamment les méthodes d'évaluation des solutions non-dominées (e.g., *somme pondérée*). Enfin, nous détaillons l'IHM développée et les résultats obtenus, suivis d'une étude comparative approfondie mettant en lumière l'efficacité du WSO face aux autres algorithmes.

3.2 Environnement de développement et de simulation

3.2.1 Java

Java est né en juin 1991 sous la forme d'un projet appelé "Oak" sous le développement d'une petite équipe d'ingénieurs travaillant pour Sun Microsystems. Ils se sont appelé l'équipe verte : James Gosling, Mike Sheridan et Patrick Naughton. Et le mot "Chêne" a été choisi pour nommer la nouvelle technologie car le chêne est un symbole de force et de durabilité. Le temps a montré que ce nom a fini

par être tout à fait approprié et même prophétique malgré son changement en Javier 1995 en raison du fait que Oak était déjà enregistré dans le cadre d'une autre marque. James Gosling était à la tête du projet, et son objectif initial était de créer un langage de programmation orienté objet qui pourrait implémenter une machine virtuelle et serait plus simple et plus universel que C/C++, mais en même temps aurait une syntaxe similaire à C/C++ pour le rendre facile à apprendre et à utiliser par les programmeurs actuels qui connaissent bien la notation C. Le nouveau langage de programmation a été conçu à l'origine principalement pour l'industrie de la télévision numérique par câble, afin de programmer la nouvelle génération de téléviseurs avec des fonctions intelligentes et divers décodeurs [11].

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 Développement and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web).

NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X et Open VMS. NetBeans est lui-même développé en Java, ce qui peut le rendre assez lent et gourmand en ressources mémoires [12].

3.2.3 JFreeChart

JFreeChart est un outil open source entièrement gratuit, ce qui permet son intégration dans des applications commerciales sans frais. Voici quelques raisons supplémentaires qui expliquent pourquoi JFreeChart constitue un excellent choix :

- Il est livré avec des API bien documentées, ce qui le rend assez facile à comprendre.
- Il prend en charge un large éventail de types de graphiques tels que les graphiques à secteurs, les graphiques en courbes, les graphiques à barres, les graphiques en aires et les graphiques 3D.
- JFreeChart est facile à étendre et peut être utilisé à la fois du côté client et des applications côté serveur.
- Il prend en charge plusieurs formats de sortie tels que PNG, JPEG, PDF, SVG, etc.
- Il permet des personnalisations étendues des graphiques [13].

3.2.4 CloudSim

CloudSim est un environnement de simulation avancé utilisé pour modéliser et simuler les infrastructures et services Cloud. Il offre la possibilité de représenter des éléments essentiels du Cloud, notamment :

- L'ordonnancement des tâches (jobs/tasks) ainsi que la gestion des événements.
- La création des entités fondamentales du Cloud telles que les centres de données, les courtiers (brokers) et les machines virtuelles (VMs).

De plus, CloudSim prend en charge :

- La communication entre les différentes entités du Cloud.

- L'implémentation de diverses politiques de courtage, incluant l'allocation des ressources et la gestion des charges de travail.

Parmi ses principales fonctionnalités, on retrouve :

- La possibilité de tester des services applicatifs dans un environnement contrôlé et reproductible.
- L'analyse et l'optimisation des goulots d'étranglement avant le déploiement réel dans un environnement Cloud.

L'expérimentation de divers scénarios, incluant :

- Des charges de travail dynamiques.
- Les performances des ressources soumises à différentes contraintes.
- Le développement de techniques adaptatives de provisionnement d'applications dans des conditions simulées.

Ainsi, CloudSim représente un outil essentiel pour la validation d'architectures Cloud complexes, permettant d'anticiper les défis liés à la montée en charge, au coût et à la qualité de service (QoS) avant le déploiement effectif.

Deux politiques d'ordonnancement des tâches sont disponibles dans CloudSim :

3.2.4.1 Space-shared

Une tâche est exécutée seule sur une VM jusqu'à sa complétion ; une fois terminée, une autre tâche peut démarrer.

3.2.4.2 Time-shared

Plusieurs tâches peuvent être exécutées sur une même VM, chacune dans une tranche de temps différente.

La structure logicielle de CloudSim est conçue sous forme de couches, comme illustré dans la figure 3.1 [27].

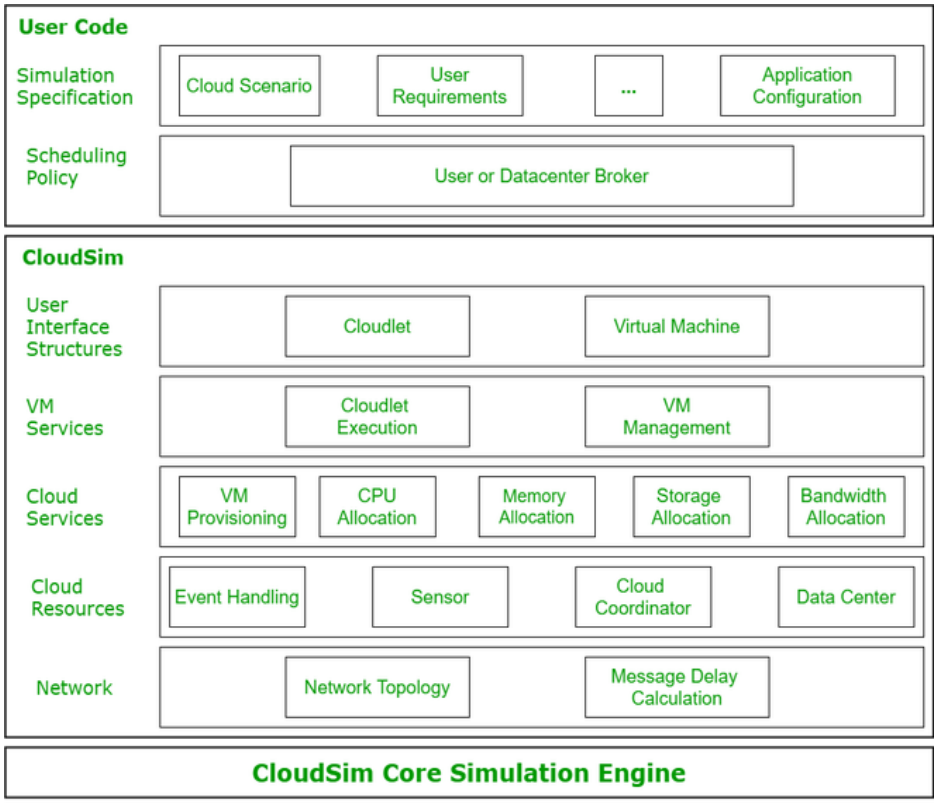

Figure 3.1: Architecture de CloudSim [14].

La figure 3.1 illustre les différentes couches de la structure du CloudSim et ses éléments architecturaux. Au niveau le plus bas est le moteur de simulation aux événements discrets Sim Java, qui implémente les fonctionnalités de base requises pour les cadres de simulation au niveau supérieur, telles que les files d'attente, le traitement des événements, la création de composants du système (services, hôte, Datacenter, Broker, les machines virtuelles), la communication entre les composants et la gestion de l'horloge de simulation.

CloudSim permet de modéliser et de simuler un environnement de centre de données basé sur le Cloud, incluant des interfaces de gestion pour les machines virtuelles (VM), la mémoire, le stockage et la bande passante. La couche CloudSim prend en charge la création et l'exécution des entités fondamentales telles que les VM, les hôtes, les centres de données et les applications tout au long de la simulation. Au sommet de la pile de simulation se trouve le code utilisateur, qui définit la configuration des éléments comme les hôtes (par exemple, leur nombre et leurs caractéristiques), les politiques d'ordonnancement du Broker, les applications (nombre de tâches et leurs exigences), les VM ainsi que le nombre d'utilisateurs

Les classes les plus couramment utilisées pendant la simulation sont :

- **Datacenter** : utilisé pour modéliser l'équipement matériel de base de tout environnement Cloud, c'est-à-dire le Datacenter. Cette classe fournit des méthodes pour spécifier les exigences fonctionnelles du Datacenter ainsi que des méthodes pour définir les politiques d'allocation des VM, etc.
- **Host** : cette classe exécute les actions liées à la gestion des machines virtuelles. Il définit également les politiques de provisionnement de la mémoire et de la bande passante aux machines virtuelles, ainsi que l'allocation des cœurs de CPU aux machines virtuelles.

- **VM** : cette classe représente une machine virtuelle en fournissant des membres de données définissant la bande passante, la RAM, les mips (millions d'instructions par seconde) d'une VM, sa taille tout en fournissant des méthodes setter et getter pour ces paramètres.
- **Cloudlet** : une classe Cloudlet représente toute tâche qui s'exécute sur une VM, comme une tâche de traitement, ou une tâche d'accès à la mémoire, ou une tâche de mise à jour de fichier, etc. Elle stocke des paramètres définissant les caractéristiques d'une tâche telles que sa longueur, sa taille en mi (million d'instructions) et fournit des méthodes similaires à la classe VM tout en fournissant des méthodes qui définissent le temps d'exécution, l'état, le coût et l'historique d'une tâche.
- **Datacenter Broker** : est une entité agissant au nom de l'utilisateur/client. Il est responsable du fonctionnement des machines virtuelles, y compris la création, la gestion, la destruction et la soumission de Cloudlets à la machine virtuelle.
- **CloudSim** : il s'agit de la classe chargée d'initialiser et de démarrer l'environnement de simulation après que toutes les entités Cloud nécessaires ont été définies et de s'arrêter plus tard après que toutes les entités ont été détruites [14].

3.3 Techniques d'optimisation multicritères

La plupart des problèmes d'optimisation réels ne se limitent pas à un seul critère, mais exigent de concilier plusieurs objectifs souvent antagonistes. Alors que l'optimisation mono-objectif cherche à résoudre un problème à l'aide d'une fonction unique (ex. minimisation du temps d'exécution), l'optimisation multi-objectif, ou *multicritère,* vise à équilibrer des critères conflictuels. Par exemple, dans le Cloud computing, un fournisseur de services doit minimiser le *makespan* (temps total d'exécution des tâches) tout en réduisant le coûts et l'énergie.

Ces problèmes nécessitent des compromis, car l'amélioration d'un critère (ex. performance) peut détériorer un autre (ex. consommation d'énergie). Pour répondre à cette complexité, des méthodes d'aide à la décision multicritère (MCDM) ont été développées. Parmi elles, la méthode de la somme pondérée (WSM), permet de sélectionner des solutions optimales dans un contexte d'ordonnancement Cloud. Cette approche est détaillée dans la suite de ce chapitre, en les appliquant à des scénarios simulés avec CloudSim.

3.3.1 Méthode de la somme pondérée

La somme pondérée est un concept mathématique fréquemment utilisé en statistiques, l'analyse des données, et la science des données pour regrouper plusieurs valeurs en une seule valeur représentative, en tenant compte de l'importance relative de chaque valeur. Cette technique est particulièrement utile lorsqu'il s'agit de jeux de données où certains éléments ont plus d'importance que d'autres. En appliquant des pondérations à chaque composant, les analystes peuvent s'assurer que la somme finale reflète l'importance souhaitée sur des points de données spécifiques, améliorant ainsi la précision et la pertinence des résultats.

3.3.1.1 La formule mathématique

La formule pour calculer une somme pondérée est simple. On peut l'exprimer ainsi :

- ✓ Somme Pondérer = $\sum_{i=1}^{n} W_i \times X_i$
- ✓ Somme Pondérer = $W_1 \times X_1 + W_2 \times X_2 + W_3 \times X_3 + \dots\dots + W_N \times X_N$
  - $\sum_{i=1}^{n} W_i = 1 \qquad 0 < W_i < 1$

Dans cette équation, (W_i) représente le poid attribué à chaque valeur (X_i). La somme des poids égale à un, il est courant de les normaliser pour garantir que la somme pondérée est proportionnée. Cette normalisation peut être obtenue en divisant chaque poids par la somme totale des poids, permettant ainsi une interprétation plus claire des résultats [15].

3.4 Approche et démarche proposée

Dans cette section, nous décrivons notre approche utilisée dans le cadre de ce PFE de manière formelle. Dans ce contexte, l'objectif de notre ordonnancement consiste à affecter des tâches aux VMs de façon à optimiser plusieurs métriques de QoS. Pour plus de détails, nous allons utiliser deux scénarios.

3.4.1 Premier scénario : Ordonnancement mono-objectif

Dans un premier scénario, nous traitons un ordonnancement mono-objectif par rapport à une seule métrique qui est le makespan. Nous évaluons par conséquent les performances de notre algorithme WSO (White Shark Optimizer) par rapport aux heuristique Round Robin que nous présenterons par la suite.

3.4.2 Second scénario : Ordonnancement multi-objectif

Dans un second scénario, nous nous focalisons sur un ordonnancement multi-objectif par rapport à trois critères qui sont le makespan, le coût et l'énergie. Nous utiliserons également une version d'évaluation des solutions dans l'algorithme WSO qui es :

3.4.2.1 La somme pondérée

Une méthode classique pour combiner plusieurs objectifs en une seule fonction de fitness.

3.4.3 Évaluation Mono-objectif

Nous présentons dans ce qui suit, l'algorithme d'ordonnancement et d'allocation de ressources utilisé dans notre PFE pour l'optimisation du critère makespan (Round Robin).

3.4.3.1 Round robin

Cet algorithme suit une stratégie simple qui consiste à distribuer de manière équitable les tâches sur les machines virtuelles disponibles, c'est-à-dire que le nombre de tâches pour chaque machine virtuelle est le même. Cet algorithme est implémenté dans le simulateur CloudSim. L'algorithme Round robin se concentre principalement sur la distribution de la charge de manière égale à toutes les ressources.

3.4.4 Évaluation Multi-objectif

Dans ce cas, nous évaluons trois critères d'optimisation qui sont : le makespan, le coût et l'énergie.

3.4.4.1 Makespan

Le makespan est l'une des métriques les plus couramment utilisées pour évaluer les performances des algorithmes d'ordonnancement. Il correspond à l'intervalle de temps écoulé entre la soumission de la première tâche et la fin d'exécution de la dernière tâche. En d'autres termes, il représente la durée totale nécessaire pour achever toutes les tâches planifiées.

3.4.4.2 Coût

La majorité des fournisseurs de services Cloud établissent des tarifs fixes pour l'utilisation de leurs ressources. Dans notre cas, le coût global d'exécution est calculé comme la somme des frais engendrés par l'utilisation des machines virtuelles (VMs) pour toutes les tâches.

3.4.4.3 Energie

L'énergie est l'une des métriques clés dans l'évaluation des performances d'un système. Elle représente la consommation totale d'énergie des machines virtuelles (VMs) pendant l'exécution des tâches. Dans notre cas, l'énergie totale est calculée en deux composantes :

- Energy active (EC) :

Consommée lorsque les VMs exécutent des tâches. Elle dépend de la tension, de la fréquence et du temps d'utilisation.

$$\boldsymbol{E_c = \sum_{i=1}^{n} AC_{ef} v_i^2 f_i wr_i^* = \sum_{i=1}^{n} \gamma v_i^2 f_i wr_i^*} \quad \textbf{(1)}$$

$\gamma = AC_{ef}$: Constante pour une machine donnée.

v_i , f_i : tension d'alimentation et fréquence du processeur sur lequel la tâches est exécutée, respectivement.

wr_i^* : le temps de calcul réel de la tâche sur le processeur qui lui est affecté.

- Energy inactive (Ei) :

Consommée lorsque les VMs sont allumées mais inactives. Elle est estimée en fonction de la tension minimale et de la durée d'inactivité.

La somme de ces deux composantes donne l'énergie totale, qui est utilisée pour évaluer l'efficacité énergétique des solutions proposées.

$$\boldsymbol{E_i = \sum_{i=1}^{m} \sum_{idle_{jk} \in IDLE_j} \gamma\, v_{minj}^2 f_{minj} L_{jk}} \quad \textbf{(2)}$$

Où :

$\boldsymbol{IDLE_j}$: l'ensemble de toutes les périodes d'inactivités sur la machine p_j .

$\boldsymbol{V_{minj}}$ ($\boldsymbol{f_{minj}}$): la plus basse tension d'alimentation (fréquence) de la machine p_j .

$\boldsymbol{L_{jk}}$: la durée d'une période d'inactivité sur la machine p_j .

Finalement, notre modèle d'énergie consommée par les ressources du Cloud pour l'achèvement du workflow est donné par l'équation suivante : [21]

$$\boldsymbol{E_{total} = E_c + E_i} \quad \textbf{(3)}$$

3.4.5 Adaptation de la métaheuristique WSO

La figure 3.2 illustre le fonctionnement détaillé de l'algorithme White Shark Optimizer (WSO) appliqué à l'ordonnancement des Cloudlets.

Définition des paramètres :

- VMs : Bandwidth, CPU count, VM count, Main memory, MIPS, OS, Policy type, Size, VMM
- Cloudlets : Job type, MI range, Distribution (%)
- Objectifs : minimiser makespan, coût, énergie

Paramètres WSO :

$n, T, v, p_1, p_2, \mu, a, b, w_0, f, nv, S_s$

Initialisation :

Générer n positions w_i (ordonnancements)

Initialiser vitesses v_i

Évaluation initiale :

Pour $i = 1$ à n :

Calcul makespan, coût, énergie

Fitness = $w_1 \times \text{makespan}_i + w_2 \times \text{coût}_i + w_3 \times \text{énergie}_i$

Mettre à jour $w_{\text{Spest},i}$, w_{gbest} (min fitness)

Tant que $t < T$:

Mettre à jour paramètres $v, p_1, p_2, \ldots$

Pour chaque $i \in [1, n]$:

$v_{i,t+1} = \mu[v_{i,t} + p_1(w_{\text{Spest},i} - w_{i,t})c_1 + p_2(w_{\text{gbest}} - w_{i,t})c_2]$

Si rand < nv :

$w_{i,t+1} = w_{i,t} \oplus w_0 + u \cdot a + l \cdot b$

Sinon :

$w_{i,t+1} = w_{i,t} + \frac{v_{i,t}}{f}$

Si rand ≤ S_s :

$D_w = |\text{rand} \times (w_{\text{Spest},i} - w_{i,t})|$

$w'_{i,t+1} = w_{\text{Spest},i} + r_1 D_w \cdot \text{sgn}(r_2 - 0.5)$

$w_{i,t+1} = \frac{w_{i,t} + w'_{i,t+1}}{2 \times \text{rand}}$

Corriger positions invalides

Évaluer fitness des nouvelles solutions

Mettre à jour $w_{\text{Spest},i}$, w_{gbest} si amélioré

Retourner w_{gbest} (meilleur ordonnancement)

Figure 3.2: Adaptation de WSO en termes des Vms et Cloudlets.

3.5 IHM développée

CloudSim s'exécute uniquement en mode console et ne dispose pas d'interface graphique. Par conséquent, j'ai développé une interface homme-machine (IHM) personnalisée pour faciliter l'accès et la manipulation de la simulation. Pour examiner le comportement de mon algorithme, je dois d'abord passer par l'étape de configuration afin de définir les paramètres de la simulation. Les interfaces suivantes illustrent les différentes étapes que j'ai mises en place pour paramétrer la simulation :

3.5.1 Interface principal

Dès le lancement de mon application, la fenêtre d'accueil illustrée dans la figure 3.3 apparait, elle contient les informations de mon PFE :

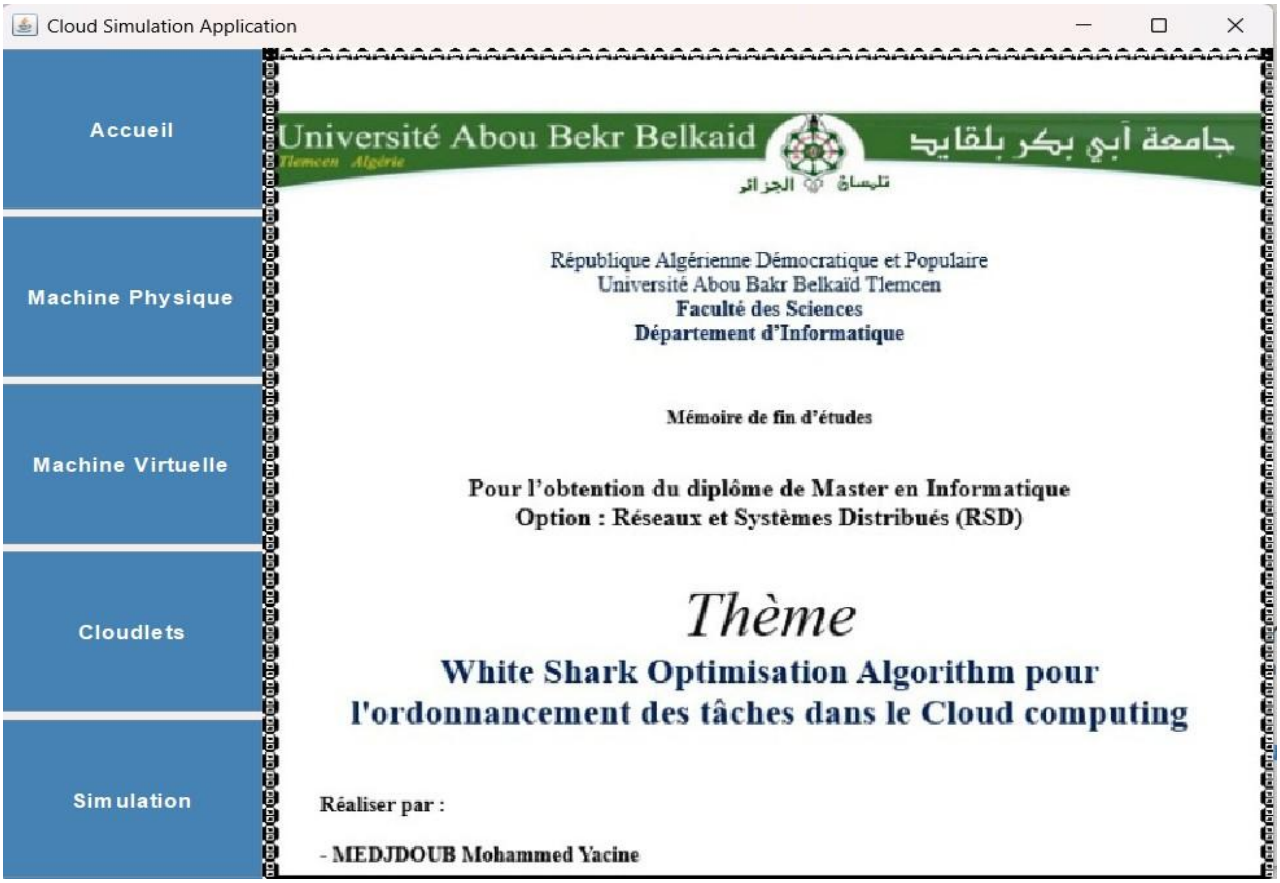

Figure 3.3: Interface principale.

3.5.2 Configuration des machines physiques

Pour configurer le Cloud, je commence par saisir les paramètres nécessaires aux machines physiques. Ces paramètres incluent le nombre de DataCenters, le nombre d'hôtes présents dans chaque DataCenter, le nombre de processeurs (PE) dans chaque hôte, la vitesse de chaque processeur (MIPS), ainsi que la RAM, la bande passante et la capacité de stockage (voir la figure 3.4).

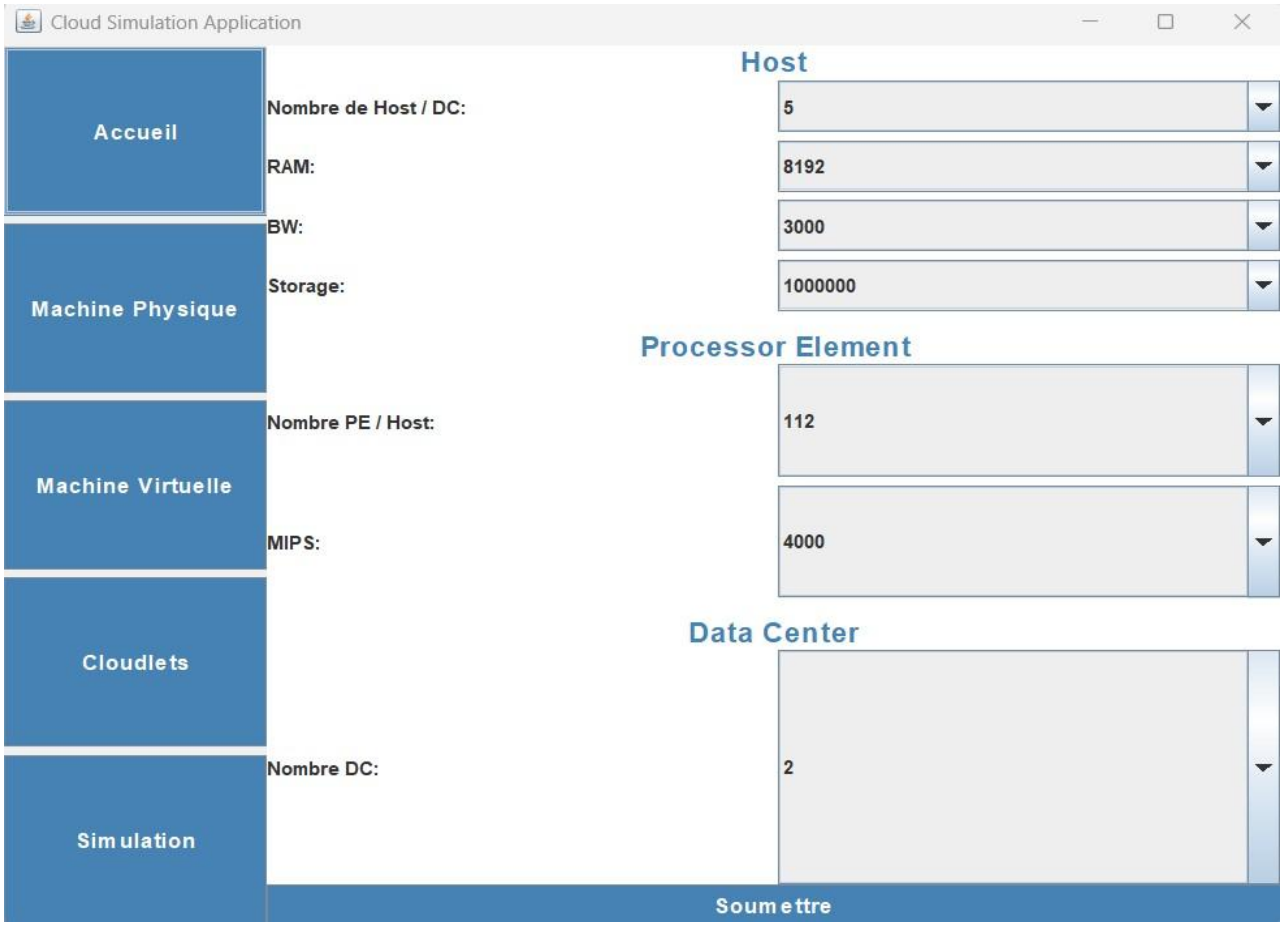

Figure 3.4: Création des machines physiques.

3.5.3 Configuration des machines virtuelles

Une fois que j'ai configuré les différentes caractéristiques des machines physiques, je devrai configurer les machines virtuelles (VMs), de telle sorte qu'un hôte peut être alloué à un ensemble de machines virtuelles.

Une machine virtuelle est caractérisée par sa vitesse de traitement (MIPS), son coût d'utilisation, son taux d'échec, sa capacité de RAM, sa bande passante ainsi que son stockage, comme indiqué dans la figure 3.5.

Figure 3.5: Création des machines virtuelles.

3.5.4 Configuration des Cloudlets

Cette interface permet de spécifier les propriétés des Cloudlets, notamment leur quantité, leur taille et leurs relations de dépendance. Ces relations servent à étendre la gestion de l'ordonnancement des tâches individuelles à celle des groupes de tâches (clusters) au sein d'une application voire la figure 3.6.

Exemple :

Considérons une application composée de quatre groupes de tâches (T1, T2), (T3), (T4, T5) et (T6), respectivement :

- Les tâches T1 et T2 s'exécutent en premier, sans ordre particulier entre elles.

- Une fois T3 terminée, les tâches T4 et T5 peuvent être exécutées dans n'importe quel ordre.

- Enfin, la tâche T6 ne peut débuter qu'après la fin des tâches T4 et T5.

Dans cet exemple, le nombre total de relations de dépendance pour l'ensemble des 6 tâches est égal à 4.

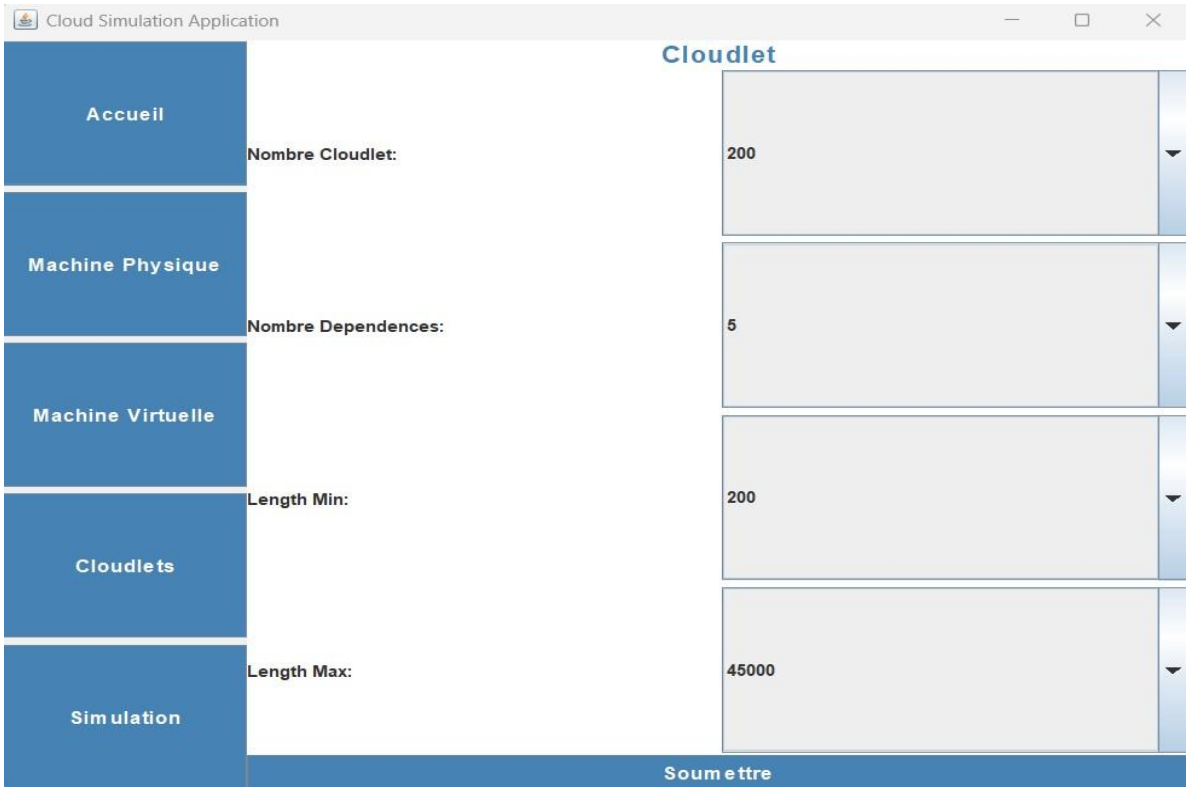

Figure 3.6: Création des Cloudlets.

3.5.5 Simulation

A partir de l'interface présentée dans la figure 3.7, l'exécution de la simulation peut être lancée, mais nous devons d'abord choisir le type d'ordonnancement à utiliser pendant la simulation (ordonnancement mono-objectif par rapport au makespan ou ordonnancement multi-objectif par rapport aux 3 métriques (makespan, coût et énergie). La simulation est lancée en cliquant sur le une des deux buttons radio (ordonnancement mono-objectif ou ordonnancement multi-objectif).

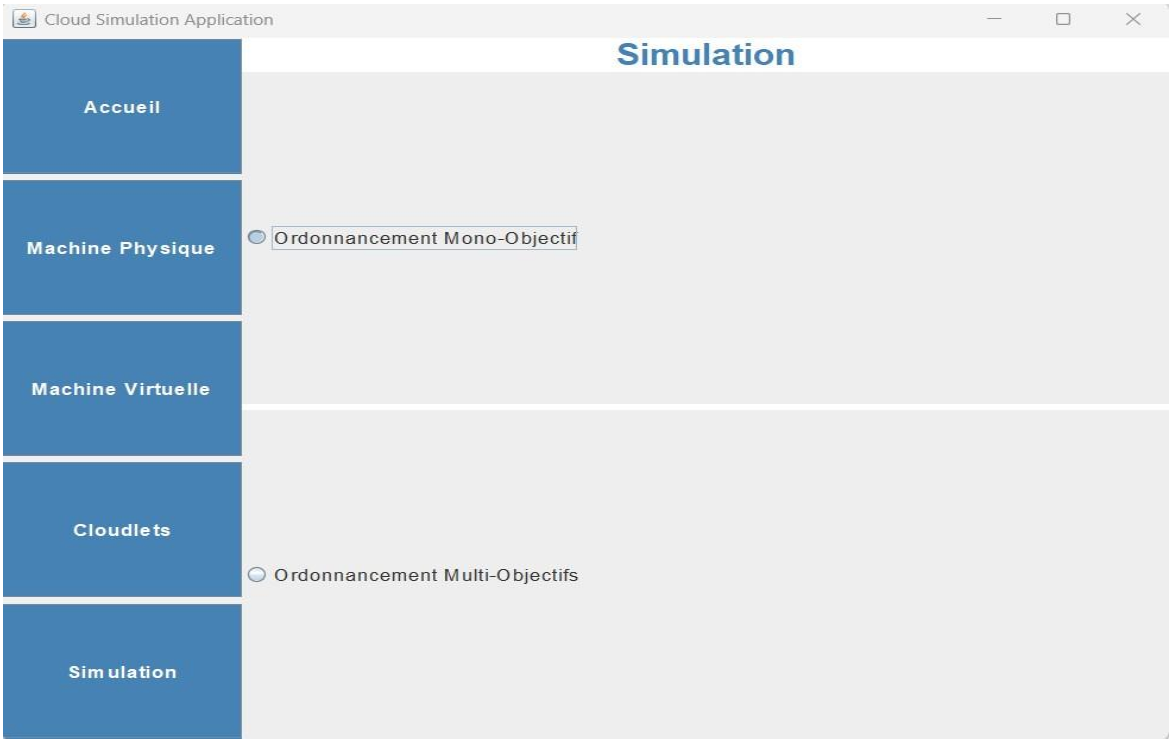

Figure 3.7: Interface de la simulation.

Après avoir terminé les deux phases de configuration et de pré-simulation, la planification des tâches dans les différentes machines virtuelles est envoyée au broker afin de lancer la simulation et afficher les résultats obtenus.

3.6 Résultats obtenus et étude comparative

Cette section présente la configuration des simulations réalisées ainsi que les résultats obtenus après l'exécution de l'algorithme White Shark Optimiser (WSO) pour l'ordonnancement des tâches dans le Cloud computing. Une comparaison avec d'autres algorithmes d'optimisation, à savoir Grey Wolf Optimiser et Salp Swarm Optimiser est également présentée afin d'évaluer les performances relatives des approches en fonction des trois critères suivants : makespan, coût et énergie.

3.6.1 Configuration expérimentale

Les simulations ont été exécutées sur une machine sous Windows 11 (64 bits), équipée d'un processeur Intel(R) Core (TM) i7-8650U CPU @ 1.90GHz (vitesse de 2.11 GHz) et d'une capacité mémoire de 16.0 GB.

Les expériences ont été menées dans un environnement hétérogène, dont les paramètres sont détaillés dans le tableau 3.1.

Sr No	Item	Details	Value
1	Virtual machine	Bandwidth	1000
		CPU count	01
		Virtual machine count	25
		Main memory	1 GB
		MIPS	100 – 1000
		Operating system	Linux
		Policy type	Time Shared
		Size	10000
		VMM	Xen
2	Physical machine	Bandwidth	2800
		Hard disk drive	1 TB
		Hosts count	05
		Main memory	8 GB
		Policy	Time shared
		Power	4 Dual core (4000 MIPS), 26 Quad core (4000 MIPS)
3	Data center	Data center count	02

Sr no	Job type	MI range	Distribution
1	Tiny	200	35 %
2	Small	1,000	40 %
3	Medium	5,000	5 %
4	Large	15,000	15 %
5	Extra large	45,000	5 %

Tableau 3.1: Les paramètres de simulation de CloudSim [28].

Le critère d'arrêt des simulations a été défini par la convergence de l'algorithme WSO. Pour garantir la robustesse des résultats, chaque simulation a été répétée 10 fois pour chaque type d'ordonnancement, et les résultats présentés correspondent à la moyenne obtenue sur ces exécutions. Cette approche permet de minimiser l'impact des fluctuations dues au caractère non déterministe des algorithmes.

3.6.2 Comparaison avec d'autres algorithmes

Outre le WSO, nous avons implémenté et testé les algorithmes suivants : Grey Wolf Optimiser (GWO) qui s'inspire de la hiérarchie sociale et du comportement de chasse des loups gris dans la nature. Le Salp Swarm Algorithm (SSA) qui est inspiré du comportement de navigation et d'alimentation des salpes (méduses) en formation de chaîne dans l'océan.

i. Makespan : La durée total nécessaire pour exécuter toutes les tâches.

ii. Coût : Le coût global associé à l'utilisation des ressources Cloud.

iii. Énergie : La consommation énergétique totale pendant l'exécution des tâches.

3.6.3 Evaluation de l'ordonnancement mono-objectif

Pour la validation de l'ordonnancement mono-objectif, une comparaison a été réalisée entre White Shark Optimiser et l'heuristique de Round Robin (pondération {1, 0, 0} afin de minimiser le

makespan). La figure 3.8 représente la comparaison entre le white Shark optimiser et round robin par rapport au makespan (mesuré en seconds) pour un nombre de tâches égal à 200, 400, 400 600, 800 et 1000 respectivement.

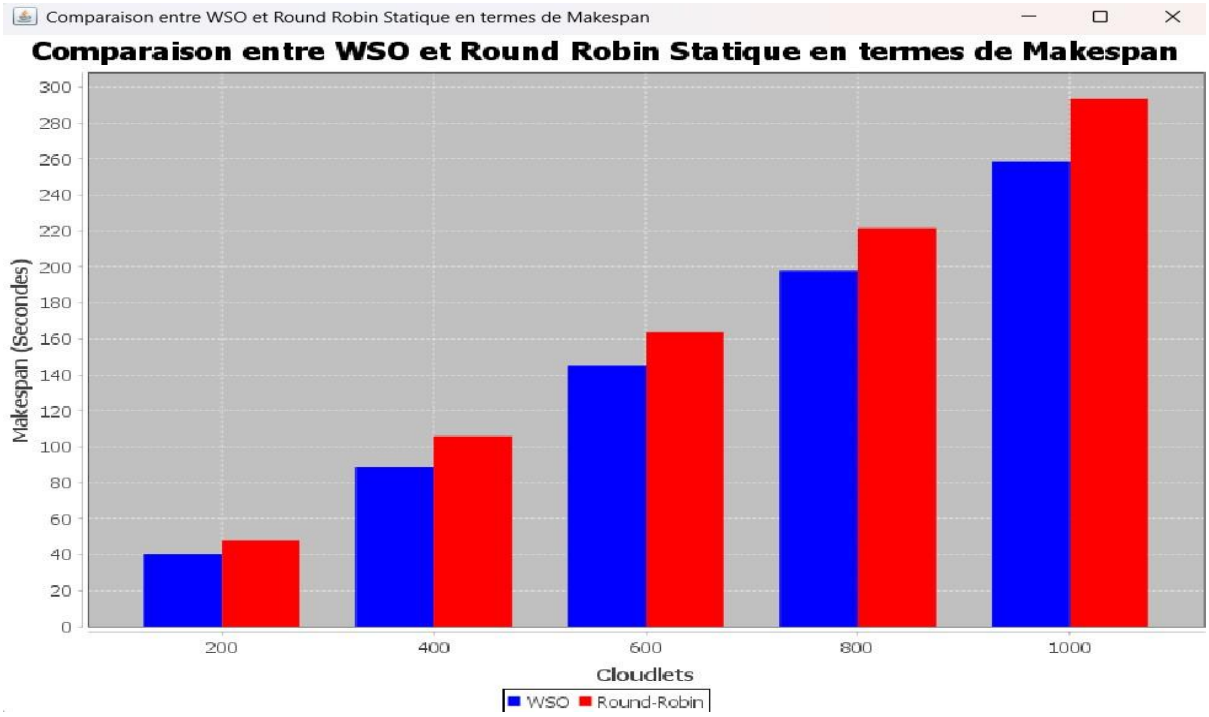

Figure 3.8: Comparaison entre WSO et l'heuristique Round Robin en termes de makespan.

Résultat : Les résultats obtenus montrent que l'algorithme d'optimisation White Shark Optimiser (WSO) présente des performances significativement supérieures à celles de l'algorithme Round Robin en termes de minimisation du makespan. Cette Conclusion est basée sur une analyse systématique des temps d'exécution observés pour différents nombres de tâches (voir le tableau 3.2).

Cloudlets	WSO-makespan(S)	RR-Makespan(S)
200	38.32	47.95
400	92.60	105.90
600	144.46	163.80
800	198.20	221.72
1000	261.00	293.72

Tableau 3.2: Les Résultats de WSO et RR en termes de makespan.

3.6.4 Evaluation de l'ordonnancement multi-objectif

Afin de valider l'ordonnancement multi-objectif, une autre comparaison a été effectuée en intégrant une version de l'algorithme WSO utilisant le principe de la somme pondérée pour l'évaluation des solutions. Cette version a été comparée aux métaheuristiques Grey Wolf Optimizer et Salp Swarm Optimizer, en agrégeant les différentes métriques de QoS (makespan, coût, énergie), afin d'identifier l'algorithme offrant la meilleure minimisation simultanée de ces trois critères.

3.6.4.1 Comparaison par rapport au makespan

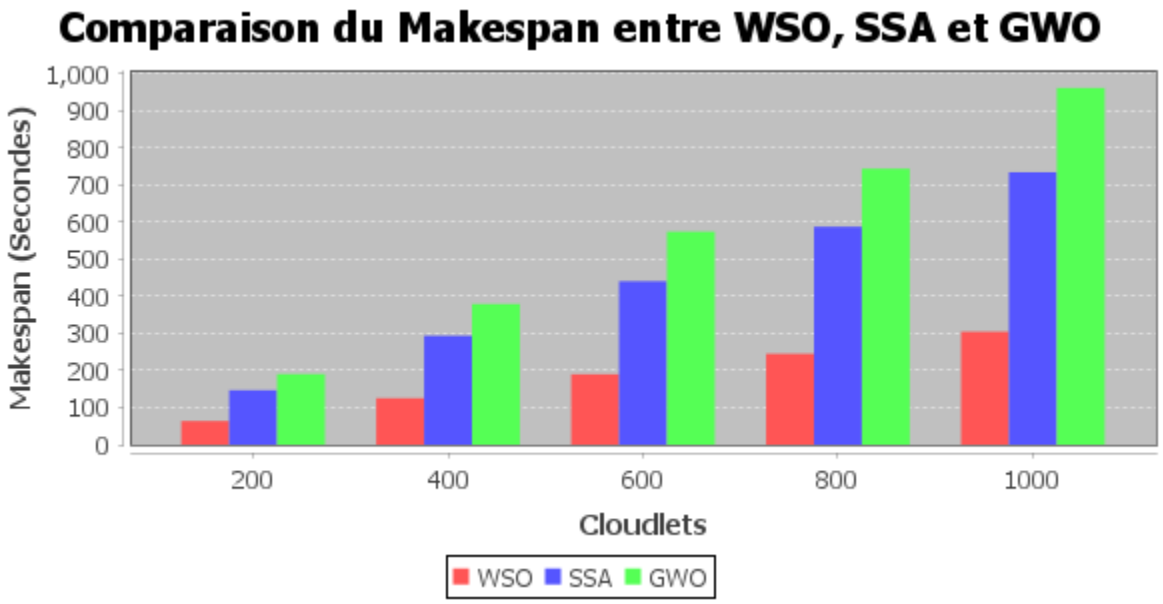

Figure 3.9: Comparaison en termes de makespan entre WSO et GWO et SSA.

Résultat : L'analyse des résultats présentés sur la figure 3.9 montre que le White Shark Optimizer (WSO) offre des performances nettement supérieures à celles du Salp Swarm Optimizer (SSA) et du Grey Wolf Optimizer (GWO), en termes de makespan. Le SSA et le GWO présentent des comportements très similaires, tandis que le WSO se distingue par son efficacité dans l'optimisation multi-objectif (voir tableau 3.3).

Cloudlets	WSO-Makespan(S)	SSA-Makespan(S)	GWO-Makespan(S)
200	63.90	147.00	190.56
400	125.60	294.00	379.75
600	190.10	441.00	574.89
800	245.40	588.00	744.71
1000	305.00	735.00	962.47

Tableau 3.3: Résultats entre WSO SSA et GWO en termes de makespan.

3.6.4.2 Comparaison par rapport au coût

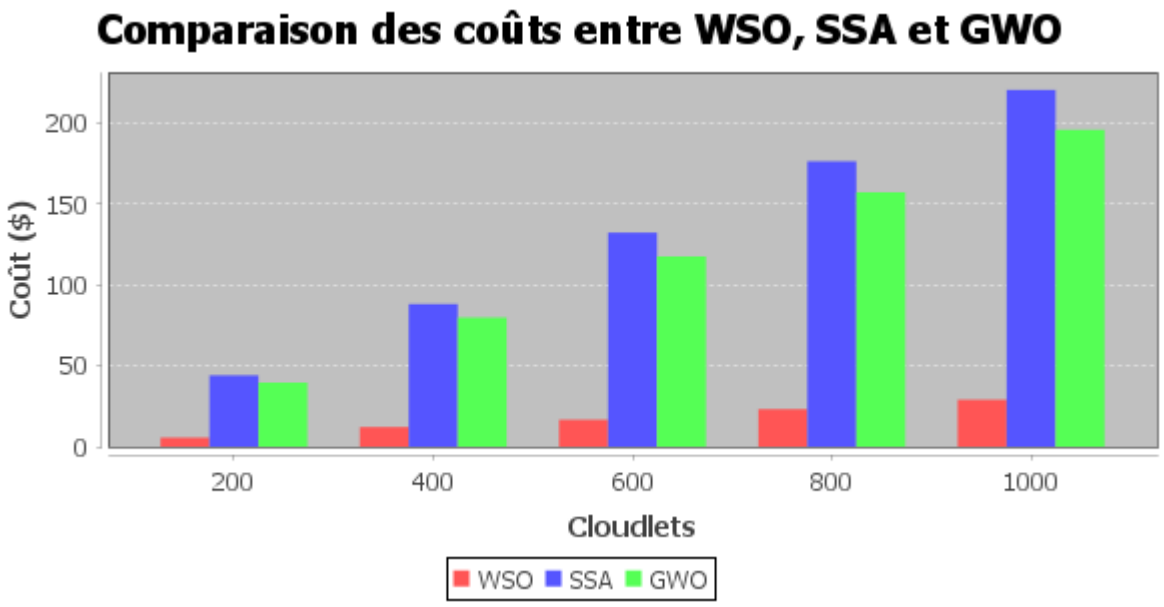

Figure 3.10: Comparaison en termes de coût entre WSO et GWO et SSA.

Résultat : L'analyse des résultats présentés sur la figure 3.10 montre que le White Shark Optimizer (WSO) offre des performances nettement supérieures à celles du Salp Swarm Optimizer (SSA) et du Grey Wolf Optimizer (GWO), en termes de coût. Le SSA et le GWO présentent des comportements très similaires, tandis que le WSO se distingue par son efficacité dans l'optimisation multi-objectif (voir tableau 3.4).

Cloudlets	WSO-Coût ($)	SSA-Coût ($)	GWO-Coût ($)
200	5.62	44.10	39.63
400	12.14	88.20	79.84
600	16.69	132.30	117.60
800	23.19	176.40	157.24
1000	29.05	220.50	195.86

Tableau 3.4: Résultats entre WSO SSA et GWO en termes de Coût.

3.6.4.3 Comparaison par rapport à l'Energie

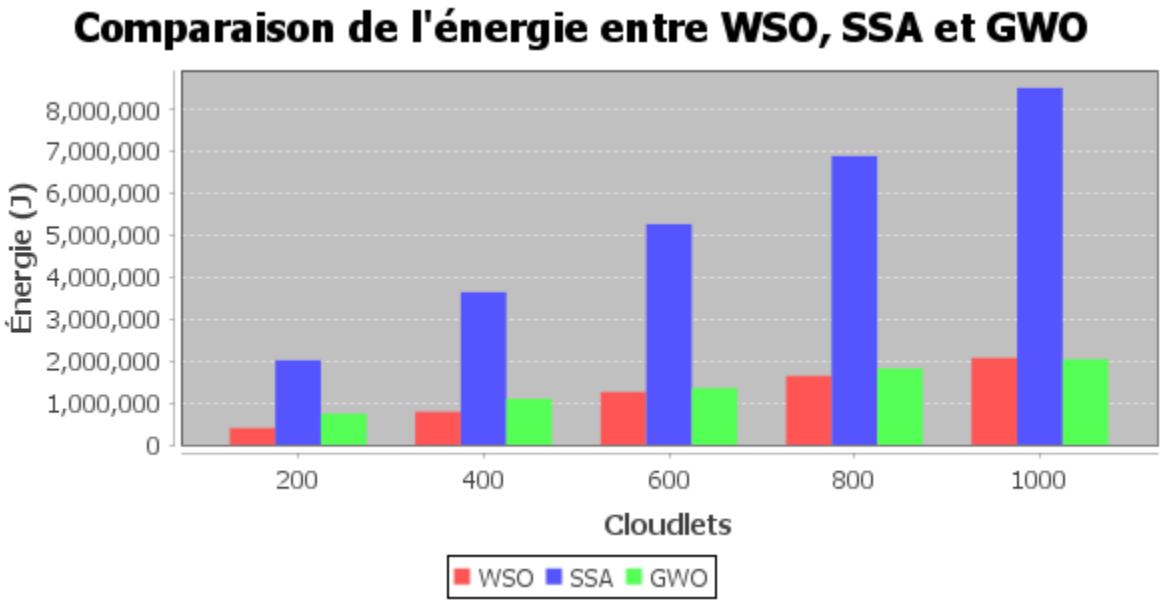

Figure 3.11: Comparaison en termes d'énergie entre les métaheuristiques.

Résultat : Le WSO (White Shark Optimiser) présente des performances énergétiques nettement supérieures à celles du SSA et du GWO (voir tableau 3.5).

Cloudlets	WSO-Energie (Joule)	SSA-Energie (Joule)	GWO-Energie (Joule)
200	398494.41	2017043.77	743297.30
400	786132.99	3642746.18	1094814.03
600	1258953.38	5268448.58	1354848.19
800	1643392.12	6894150.98	1820898.55
1000	2071355.58	8519853.37	2040255.47

Tableau 3.5: Résultats entre WSO SSA et GWO en termes d'énergie.

3.7 Conclusion

Ce chapitre a été consacré à l'évaluation comparative de l'algorithme White Shark Optimiser (WSO) par rapport à d'autres algorithmes d'optimisation, notamment SSA et GWO, dans le cadre de l'ordonnancement multi-objectif. Les simulations ont été réalisées en utilisant des outils robustes tels que le langage JAVA, l'IDE NetBeans, l'outil JFreeChart, et le simulateur CloudSim, qui ont permis une analyse approfondie des performances selon trois critères clés : makespan, coût, et énergie. Les résultats obtenus montrent que le WSO surpasse significativement les autres algorithmes dans tous les critères évalués. Concernant le makespan, mesurée en secondes, le WSO affiche des temps d'exécution plus courts, ce qui témoigne de son efficacité pour réduire les délais globaux d'achèvement des tâches. En termes de coût, le WSO se distingue par une réduction drastique des dépenses opérationnelles, offrant des économies jusqu'à dix fois supérieures par rapport aux autres méthodes. Enfin, pour la consommation d'énergie, mesurée en joules (J), le WSO démontre une efficacité énergétique remarquable, consommant nettement moins d'énergie que ses homologues, même lorsque le nombre de Cloudlets augmente.

Cette comparaison met en lumière la polyvalence et la robustesse du WSO dans des environnements Cloud complexes, où la minimisation simultanée du makespan, du coût, et de la consommation d'énergie est cruciale. Ces résultats confirment que le WSO constitue une solution optimale pour répondre aux exigences modernes de l'ordonnancement dans les systèmes Cloud, tout en garantissant un équilibre performant entre ces objectifs souvent contradictoires. Cette étude ouvre ainsi la voie à des applications pratiques prometteuses pour améliorer l'efficacité des infrastructures Cloud.

Conclusion générale

Le Cloud computing représente une avancée technologique majeure, caractérisée par l'utilisation d'un large pool de ressources informatiques accessibles à la demande via Internet. La gestion efficace de ces ressources, sous forme de tâches dépendantes, nécessite des techniques d'ordonnancement sophistiquées pour mapper les tâches sur les machines virtuelles appropriées tout en répondant aux exigences de qualité de service (QoS) spécifiées dans les contrats. Cependant, la nature conflictuelle des métriques de QoS, telles que le makespan, le coût, et la consommation d'énergie, rend ce problème particulièrement complexe et relève de l'optimisation multi-objectif.

Dans cette étude, nous avons évalué les performances du White Shark Optimizer (WSO), une métaheuristique récente, pour résoudre le problème d'ordonnancement dans le domaine du Cloud computing. Notre analyse a porté sur trois critères clés de QoS : le makespan (temps d'exécution total), le coût (dépenses opérationnelles), et la consommation d'énergie (mesurée en joules). Les résultats obtenus montrent que le WSO surpasse significativement les algorithmes classiques tels que GWO et SSA dans tous les scénarios testés. En termes de makespan, le WSO minimise les délais d'exécution, garantissant ainsi une meilleure réactivité du système. Concernant le coût, il réduit drastiquement les dépenses, offrant des économies jusqu'à dix fois supérieures par rapport aux autres méthodes. Enfin, pour la consommation d'énergie, le WSO démontre une efficacité énergétique remarquable, consommant nettement moins d'énergie même lorsque le nombre de Cloudlets augmente.

Ces résultats mettent en lumière la polyvalence et la robustesse du WSO dans des environnements Cloud complexes, où la minimisation simultanée de plusieurs objectifs souvent contradictoires est cruciale. Cette étude ouvre ainsi la voie à des applications pratiques prometteuses pour améliorer l'efficacité des infrastructures Cloud.

En perspective, il serait intéressant d'étendre cette étude à d'autres métriques de QoS, telles que l'équilibrage de charge, la tolérance aux pannes, ou encore l'empreinte carbone. Enfin, appliquer d'autres métaheuristiques permettrait de comparer leurs performances avec celles du WSO et d'identifier les meilleures stratégies pour des environnements Cloud toujours plus exigeants.

Bibliographie

[1] Qu'est-ce que le cloud computing ? Disponible sur : https://blog.blaisethirard.com/qu-est-ce-que-le-cloud-computing/. Consulté le 20/05/2025

[2] The Cloud Architecture Consulting Services for Improving your Architecture. Disponible sur : https://memprize.com/the-cloud-architecture-consulting-services-for-improving-your-architecture/. Consulté le 20/05/2025

[3] Cloud Computing Technologies. Disponible sur : https://www.tutorialride.com/cloud-computing/cloud-computing-technologies.htm. Consulté le 21/05/2025.

[4] Types of Cloud Computing. Disponible sur : https://www.interviewbit.com/blog/types-of-cloud-computing/. Consulté le 21/05/2025.

[5] Cloud computing : définition, intérêts et exemples d'applications. Disponible sur : https://www.geekmaispasque.com/2018/04/cloud-computing-definition-interets-applications/?cn-reloaded=1. Consulté le 22/05/2025.

[6] DÉFINITION ET EXPLICATIONS : LA MÉTAHEURISTIQUE. Disponible sur : https://hawassib.com/algorithme-la-metaheuristique/. Consulté le 23/05/2025.

[7] Great white shark. Disponible sur: https://seamonsters.fandom.com/wiki/Great_white_shark. Consulté le 23/05/2025.

[8] C'était le meilleur endroit au monde pour observer les grands requins blancs. Disponible sur : https://www.nationalgeographic.fr/animaux/conservation-animaux-preservation-requins-c-etait-le-meilleur-endroit-au-monde-pour-observer-les-grands-requins-blancs-il-n-en-reste-plus-aucun. Consulté le 23/05/2025.

[9] A bit on sharks – from Shark Week 2021. Disponible sur: https://ocean-noise.com/2021/07/a-bit-on-sharks-from-shark-week-2021/. Consulté le 23/05/2025.

[10] Organs used for Sensing. Disponible sur : https://arundivysharks.weebly.com/sensitivity.html. Consulté le 23/05/2025.

[11] Histoire de Java. Une histoire complète du développement Java, de 1991 à 2021. Disponible sur : https://codegym.cc/fr/groups/posts/fr.594.histoire-de-java-une-histoire-complete-du-developpement-java-de-1991-a-2021. Consulté le 23/05/2025.

[12] NetBeans – Définition. Disponible sur : https://www.techno-science.net/definition/5346.html. Consulté le 23/05/2025.

[13] JFreeChart – Présentation. Disponible sur : https://tutoriels.edu.lat/pub/jfreechart/jfreechart-overview/jfreechart-presentation . Consulté le 23/05/2025.

[14] What is CloudSim? Disponible sur : https://www.geeksforgeeks.org/what-is-cloudsim/. Consulté le 24/05/2025.

[15] Qu'est-ce que c'est : le modèle de somme pondérée Disponible sur : https://fr.statisticseasily.com/glossaire/qu%27est-ce-que-le-modèle-de-somme-pondérée/. Consulté le 24/05/2025.

[16] Naziha ALI SAOUCHA (2020). Exploitation des techniques de l'intelligence artificielle pour l'optimisation de la QoS et l'efficacité spectrale dans les réseaux de radio cognitive. Thèse de doctorat en Informatique. Université de Tlemcen.

[17] Estele Glize. Méthodes exactes pour les problèmes combinatoires bi-objectif: Application sur les problèmes de tournées de véhicules. Automatique / Robotique. INSA de Toulouse, 2019.

[18] Ibrahim Moussa. Modèles de résolution approchée et efficace pour les problèmes des réseaux de transport et de télécommunication. Université de Picardie Jules Verne, 2015. Français.

[19] Sidi Mohamed Douiri, Souad Elbernoussi, Halima Lakhbab. Cours des Méthodes de Résolution Exactes, Heuristiques et Métaheuristiques. MASTER CODES, CRYPTOGRAPHIE ET SÉCURITÉ DE L'INFORMATIONDE. Université Mohammed V, Faculté des Sciences de Rabat.

[20] Ghoumari, Asmaa. Métaheuristiques adaptatives d'optimisation continue basées sur des méthodes d'apprentissage. Thèse de doctorat en Informatique. Université Paris-Est, 2018.

[21] Yassa, Sonia. "Allocation optimale multicontraintes des workflows aux ressources d'un environnement cloud computing." Thèse de doctorat en Informatique. Université de Cergy Pontoise, 2014.

[22] Zhou, J., Huang, S., Wang, M., & Qiu, Y. (2022). Performance evaluation of hybrid GA–SVM and GWO–SVM models to predict earthquake-induced liquefaction potential of soil: a multi-dataset investigation. *Engineering with Computers*, 1-19.

[23] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61.

[24] Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in engineering software, 114, 163-191.

[25] Braik, M., Hammouri, A., Atwan, J., Al-Betar, M. A., & Awadallah, M. A. (2022). White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. Knowledge-Based Systems, 243, 108457.

[26] Lepe-Silva, Fernando, Broderick Crawford, Felipe Cisternas-Caneo, José Barrera-Garcia, and Ricardo Soto. "A Binary Chaotic White Shark Optimizer." Mathematics 12, no. 20 (2024) : 3171.

[27] Nguyen, Hoa T., Prabhakar Krishnan, Dilip Krishnaswamy, Muhammad Usman, and Rajkumar Buyya. Quantum cloud computing: review, open problems, and future directions. arXiv preprint arXiv:2404.11420 (2024).

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

Résumé

L'ordonnancement des tâches joue un rôle essentiel dans l'optimisation des performances d'un environnement de Cloud computing. En effet, les fournisseurs et les utilisateurs des services Cloud poursuivent des objectifs souvent divergents, rendant la gestion des ressources complexe. Un ordonnanceur efficace doit ainsi proposer un équilibre satisfaisant entre ces différentes contraintes. Ce travail s'inscrit dans cette problématique en cherchant à optimiser l'affectation des tâches aux machines virtuelles. Pour cela, nous avons exploité la métaheuristique White Shark Optimizer (WSO) en intégrant trois métriques de qualité de service (QoS) : le makespan, le coût et l'énergie. L'évaluation des performances a été réalisée à l'aide de CloudSim, et les résultats obtenus démontrent l'efficacité de l'approche proposée.

Mots clés : Cloud computing, optimisation multi-objectif, QoS, WSO, CloudSim.

Abstract

Task scheduling plays a crucial role in optimizing the performance of a Cloud computing environment. Cloud service providers and users often have conflicting objectives, making resource management a complex challenge. An efficient scheduler must find a balance that satisfies these different constraints. This work focuses on optimizing task allocation to virtual machines. To achieve this, we utilized the White Shark Optimizer (WSO) while incorporating three quality of service (QoS) metrics: makespan, cost, and energy. Performance evaluations were conducted using CloudSim, and the results obtained demonstrate the effectiveness of the proposed approach.

Keywords: Cloud computing, multi-objective optimization, QoS, WSO, CloudSim.

الملخص

تلعب جدولة المهام دورًا حاسمًا في تحسين أداء بيئة الحوسبة السحابية. غالبًا ما تكون لدى مزوّدي الخدمات السحابية والمستخدمين أهداف متعارضة، مما يجعل إدارة الموارد تحديًا معقدًا. يجب على جدول زمني فعّال أن يجد توازنًا يحقق هذه الشروط المختلفة. يركّز هذا العمل على تحسين تخصيص المهام إلى الآلات الافتراضية. ولتحقيق ذلك، استخدمنا خوارزمية القرش الأبيض (WSO) مع دمج ثلاث مقاييس لجودة الخدمة (QoS): مدة الإنجاز، التكلفة، والطاقة. أجريت تقييمات الأداء باستخدام CloudSim (بيئة محاكاة)، وأظهرت النتائج التي تم الحصول عليها فعالية النهج المقترح.

الكلمات المفتاحية: الحوسبة السحابية، التحسين متعدد الأهداف، جودة CloudSim, ،WSO ،QoS.