الجـــمـهوريـــة الجـــزائـرية الديمقـــراطيـــة الشـــعبيـــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التـــعليـــم العـــالي والبحـــث العلـــمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جـــــامعة أبي بكـر بلـــقـايد – تـلمســـــان –
Université Aboubakr Belkaïd – Tlemcen –
Faculté des Sciences Tidjani Haddam
Département d'informatique

MÉMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : MATHÉMATIQUES ET INFORMATIQUE
FILIÈRE : INFORMATIQUE
Spécialité : RÉSEAUX ET SYSTÈMES DISTRIBUÉS

Par : Melle CHAREF Khadidja Fatna.

Sujet

Shark Smell Optimization SSO pour l'ordonnancement des tâches dans le Cloud Computing.

Soutenu, le 28 / 06 / 2025, devant le jury composé de :

Mr. BENMAMMAR Badr	Univ. Tlemcen	Président
Mme.Benfriha Sihem	Univ. Tlemcen	Examinatrice
Mr. BENMOUNA Youcef	Univ. Tlemcen	Encadrant

Année universitaire 2024/2025

Remerciments

الحَمْدُ للهِ الَّذِي بِنِعْمَتِهِ تَتِمُّ الصَّالِحَات، وَبِهِ نَسْتَعِينُ عَلَى أُمُورِ الدُّنْيَا وَالدِّين.

Alhamdulillah, toute louange revient à **Allah**, Le Tout-Puissant, qui m'a accordé la force, la patience et la persévérance pour mener à bien ce travail.

Je remercie sincèrement **Monsieur BENMOUNA Youcef**, mon encadrant, pour son accompagnement, ses conseils précieux et sa disponibilité tout au long de ce travail.

Je remercie également les membres du jury, en particulier **Monsieur BENMAMMAR Badr**, président du jury, ainsi que **l'examinatrice**, pour l'attention qu'ils ont portée à mon travail, ainsi que pour leurs remarques pertinentes et constructives, qui témoignent de leur expertise et enrichissent la portée de cette recherche.

Je remercie également tous les enseignants de la filière pour la qualité de l'enseignement dispensé et pour leur engagement pédagogique tout au long de mon parcours universitaire.

Je souhaite ensuite exprimer ma plus profonde reconnaissance à **ma chère mère**, pour son amour inconditionnel, sa patience infinie, ses prières et son soutien moral inestimable. Que ce travail soit un humble hommage à ses sacrifices.

Mes remerciements affectueux vont également à mes sœurs **Chahra et Feriel**, et mes frères **Mohamed et Abdelali**, pour leur présence, leur encouragement constant et leur affection indéfectible.

Je tiens aussi à remercier du fond du cœur **mes amis les plus proches** : **Oussama**, **Ali**, **Nawel** et **Boubker**, pour leur soutien, leur bienveillance et leur amitié précieuse tout au long de ce parcours.

Enfin, je n'oublie pas mes collègues et camarades : **Sabah**, **Aya**, **Lamya**, **Tema** et **Khawla**, pour les moments partagés, leur esprit d'entraide et leur amitié sincère.

À toutes ces personnes qui m'ont soutenue, conseillée ou simplement encouragée, je vous adresse ma plus profonde gratitude

Dédicaces

Je dédie ce modeste travail :

À **la mémoire de mon père**,
Qu'Allah lui accorde Sa miséricorde et l'accueille dans Son vaste paradis. Ton souvenir habite mon cœur et éclaire chacun de mes pas. Ce travail est un hommage à tout ce que tu m'as transmis.

À **ma chère mère**,
Ton amour, ton courage, tes prières silencieuses et ton soutien indéfectible sont ma plus grande richesse. Dans chaque moment de doute, ta force m'a portée. Que ce mémoire reflète ne serait-ce qu'une part de ta patience infinie.

À **mes frères Mohamed et Abdelali**, et **mes sœurs Chahra et Feriel**,
Merci pour vos paroles réconfortantes, votre présence fidèle et votre affection sincère. Vous êtes mon repère.

À **mes amis les plus proches** : **Oussama**, **Ali**, **Nawel** et **Boubker**,
Merci pour votre présence dans les moments difficiles, vos encouragements constants, et votre foi en moi, même quand je doutais.

À **Sabah**, **Aya**, **Lamya**, **Tema** et **Khawla**,
Pour votre amitié sincère, votre esprit d'entraide, et les instants partagés avec bienveillance.

Avec toute ma reconnaissance, mon affection et mon respect.

TABLE DES MATIERES

LISTE DE FIGURES ..... **I**

LISTE DE TABLEAU ..... **III**

Liste des Abréviations ..... **IV**

Introduction **1**

CHAPITRE I **3**

1 Introduction 4

2 Définition et concepts du Cloud Computing 4

2.1 Définition du Cloud Computing 4

2.2 L'évolution du cloud Computing 5

2.3 Modèles de déploiement du cloud Computing 5

2.4 Cloud public 6

2.5 Cloud privé 6

2.6 Cloud Communautaire 7

2.7 Cloud hybride 8

2.8 Les modèles de services du cloud Computing 9

2.8.1 IaaS (Infrastructure as a Services) 10

2.8.2 PaaS (Platform as a Services) 10

2.8.3 SaaS (Software as a Service) 11

2.9 Comment choisir l'infrastructure cloud 12

2.10 Caractéristiques du cloud Computing 13

3 L'architecture d'une infrastructure cloud 14

4 Les technologies du Cloud Computing 15

4.1 Virtualisation 15

4.2 Automatisation 16
4.3 Les avantages et les limites du cloud Computing 16
4.3.1 Les avantages du cloud Computing 16
4.3.2 Les limites du cloud Computing 17
5 Conclusion 18
Chapitre II **15**
1 Introduction 16
2 Les Méthodes de Résolution 16
2.1 Méthodes exactes 17
2.2 Méthodes approchées 17
3 Shark Smell Optimization (SSO) 20
4 L'idée et l'inspiration de l'algorithme 20
4.1 Projection biologique vers une modélisation mathématique dans l'algorithme SSO 20
4.1.1 Source fixe de l'odeur (la proie) : Meilleure solution actuelle 21
4.1.2 Concentration d'odeur plus forte : Meilleure valeur de la fonction objective 21
4.1.3 Détection directionnelle (narine gauche vs droite) : Calcul du gradient directionnel Sdgds . 21
4.1.4 Mouvement vers la source : Mise à jour de la position 21
4.1.5 Recherche locale (spirale) : Perturbation contrôlée autour de la solution 21
4.1.6 Intensité décroissante : Diminution progressive des paramètres de recherche 21
5 Hypothèses de modélisation de l'algorithme SSO 22
5.1 Initialisation de SSO : génération des particules initiales d'odeur 22
5.2 Mouvement du requin vers la proie : 23
6 Le Pseudo-code de SSO 29
7 Conclusion 30
Chapitre III **31**
1 Introduction 32

2 Fondements des Algorithmes d'Optimisation BA et PSO 32
2.1 Description de l'Algorithme PSO (Particle Swarm Optimization) 32
2.2 Description de l'Algorithme BA (Bat Algorithm) 32
3 Outils et environnement de simulation et de développement 33
3.1 Outils de travail 33
3.2 Langage de programmation java : 33
3.3 Environnement de développement intégré : NetBeans : 33
3.4 JFreeChart 34
3.5 Le simulateur CloudSim 34
3.5.1 Définition 34
3.5.2 Architecture de CloudSim 34
3.5.3 Les différentes classes du CloudSim 36
3.6 Critères d'évaluation 37
3.6.1 Makespan 37
3.6.2 Coût : 38
3.6.3 Énergie d'exécution : 38
3.7 La méthode de la somme pondérée 39
3.7.1 Exigences essentielles pour la mise en œuvre : 39
3.7.2 Contraintes de poids : 39
3.7.3 Processus de normalisation : 40
3.8 IHM développée 40
3.9 Résultats obtenus et étude comparative 46
3.10 Comparaison de résultats en termes de makespan : 48
3.10.1 Résultat : 49
3.11 Comparaison de résultats en termes de coût : 49
3.11.1 Résultat : 50

3.12 Comparaison de résultats en termes de d'énergie : 50

3.12.1 Résultat : 51

3.12.2 Synthèse des résultats : 51

4 Conclusion 52

Conclusion **53**

Références **55**

Résumé **57**

LISTE DE FIGURES

CHAPITRE I

Figure I. 1 : Cloud Computing.....4

Figure I. 2: Les différentes technologies dans l'ère de l'informatique.....5

Figure I. 3: Les modèles de déploiement du cloud.....6

Figure I. 4: Modèle de déploiement d'un cloud public. 6

Figure I. 5: Modèle de déploiement d'un cloud privé. 7

Figure I. 6: Modèle de déploiement d'un cloud communautaire. 7

Figure I. 7: Modèle de déploiement d'un cloud hybride. 8

Figure I. 8: Les différentes services du cloud computing. 9

Figure I. 9: Les différentes services du cloud computing. 12

Figure I. 10: Caractéristiques du cloud Computing. 14

Figure I. 11: L'architecture d'une infrastructure cloud. 14

Figure I. 12: Schéma de virtualisation des serveurs utilisant un stockage partagé. 16

Figure I. 13: Les huits avantages du cloud computing. 17

CHAPITRE II

Figure II. 1: Classification des différentes méthodes pour résoudre les problèmes d'optimisation. 16

Figure II. 2: Schéma du déplacement du requin vers la source de l'odeur.....20

Figure II. 3: Mouvement de rotation d'un requin. 27

Figure II. 4: Diagramme de flux de l'algorithme (SSO). 29

Figure II. 5: Le Pseudo-code de l'algorithme (SSO). 30

CHAPITRE III

Figure III. 1: L'architecture en couches de CloudSim. 35

Figure III. 2: Interface principale. 41

Figure III. 3: choix de configuration. 42

Figure III. 4: Choix de l'algorithme d'ordonnancement. 43

Figure III. 5: Lancement des résultats de simulation. 44

Figure III. 6: Configuration des Cloudlets. 44

Figure III. 7: configuration des machines virtuelles. 45

Figure III. 8: configuration des machines physiques. 45

Figure III. 9: Lancement de simulation. 46

Figure III. 10: comparaison en termes de makespan pour 25 Vms. 48

Figure III. 11: comparaison en termes de coût pour 25 Vms. 49

Figure III. 12: comparaison en termes d'énergie pour 25 Vms. 50

LISTE DE TABLEAU

CHAPITRE I

Tableau I. 1: comparaison entre Cloud public, privé, hybride et communautaire. 9

CHAPITRE II

Tableau II. 1: les principales différences entre les heuristiques et les métaheuristiques. 19

CHAPITRE III

Tableau III. 1: les paramètres de simulation CloudSim 47

Tableau III. 2: les paramètres de longueur des Cloudlets. 47

Tableau III. 3: résultats de comparaison en termes de makespan pour 25 Vms. 48

Tableau III. 4: résultats de comparaison en termes de coût pour 25 Vms. 50

Tableau III. 5: résultats de comparaison en termes d'énergie pour 25 Vms. 51

Liste des Abréviations

BA	Bat Algorithm
CloudSim	Cloud Simulation Framework
CPU	Central Processing Unit
DE	Développement Environnement (interprété ici comme Environnement de Développement)
GHz	Gigahertz
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IHM	Interface Homme-Machine
JPEG	Joint Photographic Experts Group
JFreeChart	Java library for chart creation
JVM	Java Virtual Machine
Java SE	Java Standard Edition
Java EE	Java Enterprise Edition
Java ME	Java Micro Edition
MB	Megabyte
MI	Million Instructions
MIPS	Million Instructions Per Second
NIST	National Institute of Standards and Technology
OF	Objective Function
PaaS	Platform as a Service
PC	Personal Computer
PDF	Portable Document Format
PNG	Portable Network Graphics
PSO	Particle Swarm Optimization
QoS	Quality of Service
RAM	Random Access Memory
SaaS	Software as a Service
SSO	Shark Smell Optimization
SVG	Scalable Vector Graphics
3D	Trois Dimensions
TIC	Technologies de l'Information et de la Communication
VMM	Virtual Machine Monitor
VM / VMs	Virtual Machine(s)
WSM	Weighted Sum Method
Green IT	Green Information Technology

Introduction Générale

Introduction Générale

Le cloud computing est un paradigme qui a profondément transformé la manière dont les organisations gèrent leurs infrastructures informatiques, offrant flexibilité, évolutivité et réduction des coûts [1]. Cependant, cette transformation soulève également des défis majeurs liés à la gestion efficace des ressources, notamment dans le domaine de l'ordonnancement des tâches**.** Comme le montrent plusieurs études, l'essor du cloud computing a conduit à une augmentation significative de la complexité des environnements informatiques, rendant indispensable l'optimisation des processus d'allocation des ressources [2].

L'ordonnancement des tâches dans le cloud computing consiste à attribuer efficacement les ressources disponibles aux différentes tâches en fonction de leurs besoins spécifiques et des contraintes imposées. Dans un contexte hétérogène comme celui du cloud, où les ressources varient en termes de performances, de disponibilité et de coût, ce problème devient particulièrement difficile. De plus, lorsque plusieurs objectifs doivent être optimisés simultanément (par exemple, minimiser le temps total d'exécution ou réduire les coûts), la difficulté s'accroît encore.

Dans ce mémoire, nous explorons l'utilisation de la métaheuristique bio-inspirée **Shark Smell Optimization (SSO)** pour résoudre le problème d'ordonnancement des tâches dans le cloud computing. Inspiré du comportement des requins lorsqu'ils recherchent leur nourriture, le SSO se distingue par sa capacité à explorer efficacement des espaces de recherche complexes tout en combinant exploration et exploitation. La suite de ce mémoire est organisée comme suit :

- Le **Chapitre I** introduit les concepts fondamentaux du cloud computing et les défis liés à l'ordonnancement des tâches.
- Le **Chapitre II** détaille le fonctionnement du SSO et son adaptation au contexte de l'ordonnancement.
- Le **Chapitre III** expose les résultats obtenus à travers des simulations et des comparaisons avec d'autres approches.

CHAPITRE I *: Cloud Computing*

1 Introduction

La technologie de l'Internet a connu une croissance exponentielle depuis son apparition, transformant en profondeur les modes de communication, de travail et d'exploitation des données. Dans ce contexte évolutif, une nouvelle tendance révolutionnaire a émergé dans le domaine des Technologies de l'Information et de la Communication (TIC) : le Cloud Computing**.** Cette innovation offre des occasions aux sociétés de réduire les coûts d'exploitation des logiciels par leurs utilisations directement en ligne. En effet le cloud est un moyen par lequel nous pouvons accéder aux applications sur internet, il nous permet de créer, configurer et personnaliser des applications en ligne.

2 Définition et concepts du Cloud Computing

2.1 Définition du Cloud Computing

Le cloud computing ou l'informatique en nuage est l'exploitation de la puissance de calcul ou de stockage des serveurs informatiques distants par l'intermédiaire d'un réseau, généralement internet. Ces serveurs sont loués à la demande, le plus souvent par tranche d'utilisation selon les critères techniques (puissance, bande passante, etc) mais également au forfait. Le cloud computing se caractérise par sa grande souplesse : selon le niveau de compétence de l'utilisation client, il est possible de gérer soi-même son serveur ou de se contenter d'utiliser des applicatifs distants. Selon la définition du National Institute of Standards and Technlogy(NIST),Le cloud computing est l'accès via un réseau de télé- commutations, à la demande et libre-service, à des ressources informatiques partagées configurables .il s'agit d'une délocalisation de l'infrastructure informatique [1]**.**

Figure I. 1 : Cloud Computing.

2.2 L'évolution du cloud Computing

L'évolution de l'informatique a traversé plusieurs phases majeures. Tout a commencé avec les mainframes, des ordinateurs centraux puissants mais coûteux. Ensuite, les mini-ordinateurs ont émergé, offrant une puissance de calcul plus accessible. Puis sont venus les ordinateurs personnels (PC), qui ont démocratisé l'accès à la technologie, jusqu'à atteindre le cloud computing qui a révolutionné le domaine en externalisant les ressources informatiques et en les rendant accessibles à distance via internet [2].

La figure ci-dessous illustre les différentes technologies dans l'ère de l'information :

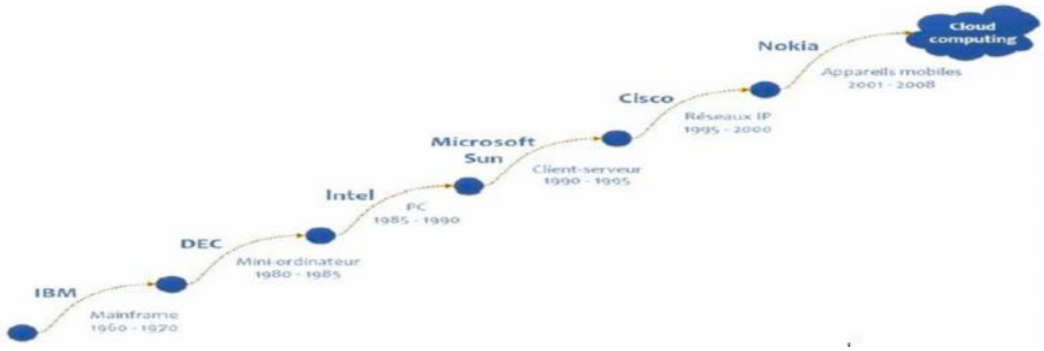

Figure I. 2: Les différentes technologies dans l'ère de l'informatique.

2.3 Modèles de déploiement du cloud Computing

Le cloud computing utilise un pool partagé de ressources informatiques (par exemple des réseaux, des serveurs, des espaces de stockage, des applications et des services) afin de fournir un accès réseau à la demande.

Comme le montre la Figure I.3, le NIST (National Institute of Standards and Technology) a défini quatre types de modèles de déploiement du cloud [3].

- Public
- Privé
- Communautaire
- Hybride

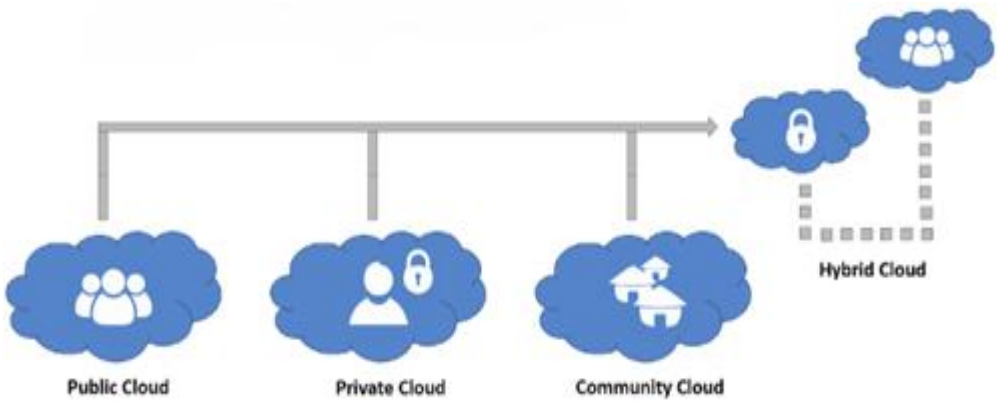

Figure I. 3: Les modèles de déploiement du cloud.

2.4 Cloud public

Le cloud public est un pool de ressources virtuelles, créées à partir de matériel détenu et géré par une entreprise externe, destiné à une utilisation par le grand public. Ces ressources sont automatiquement provisionnées et allouées à différents clients via une interface en libre-service. Ce modèle permet de faire évoluer facilement les charges de travail, même en cas de fluctuations imprévues de la demande [4].

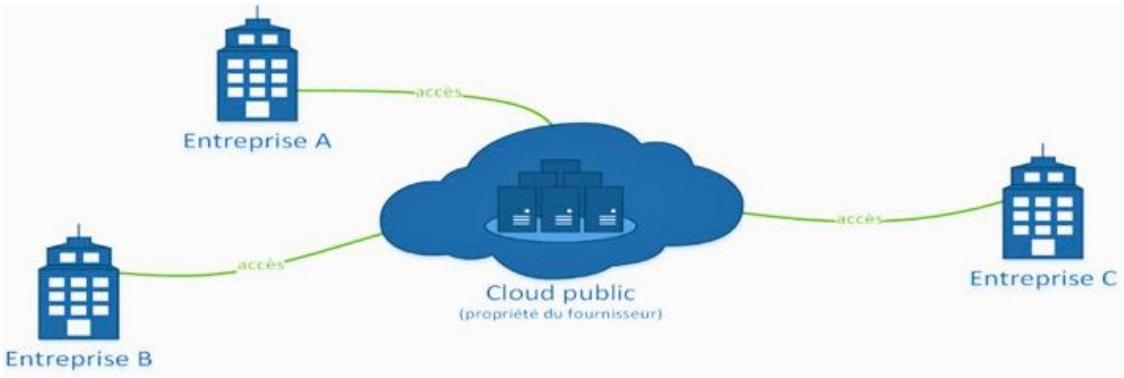

Figure I. 4: Modèle de déploiement d'un cloud public.

2.5 Cloud privé

Désigne des services cloud créés exclusivement pour une organisation unique, accessibles via Internet ou un réseau privé à un ensemble restreint d'utilisateurs. Son infrastructure peut être physiquement située sur site ou en dehors de celui-ci, et peut appartenir à un fournisseur tiers. Ce modèle offre un niveau de sécurité et de personnalisation supérieur, en ne fournissant des services qu'aux membres de cette seule organisation [5].

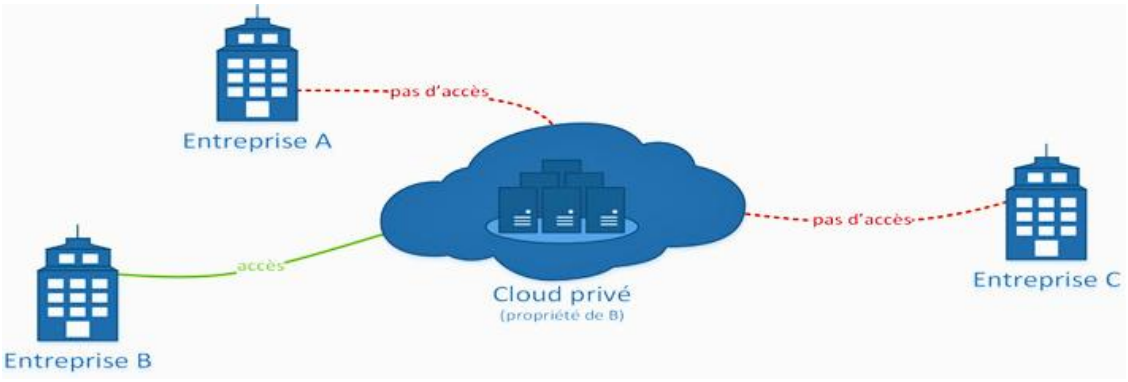

Figure I. 5: Modèle de déploiement d'un cloud privé.

2.6 Cloud Communautaire

Créé pour une utilisation exclusive par une communauté spécifique. La communauté se compose de plusieurs organisations partageant les mêmes préoccupations (par exemple en matière de mission, d'exigences de sécurité, de stratégie ou de critères de conformité). L'infrastructure peut être physiquement située sur le site ou en dehors de celui-ci, et elle peut appartenir à un fournisseur distinct ou à une ou plusieurs des organisations de la communauté. La différence entre le cloud public et le cloud communautaire se réfère aux besoins fonctionnels qui ont été personnalisés pour la communauté [6].

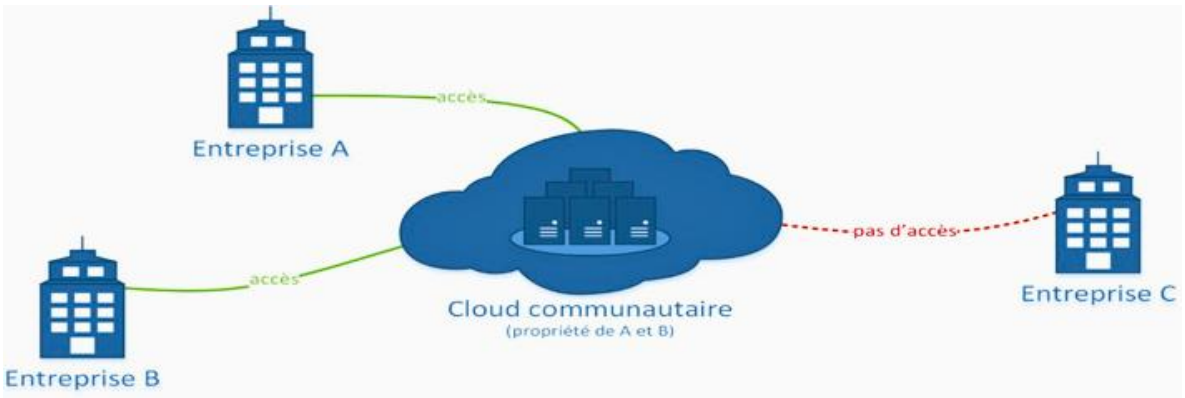

Figure I. 6: Modèle de déploiement d'un cloud communautaire.

2.7 Cloud hybride

C'est la combinaison d'au minimum deux infrastructures de cloud distinctes (cloud privé, communautaire ou public), représentant des entités uniques. Ces entités sont reliées par le biais d'une technologie permettant la portabilité des données et des applications. Cette portabilité permet à une organisation de conserver un point de vue unique en matière de solution du cloud, tout en profitant des avantages offerts par différents fournisseurs du cloud. Par exemple, la zone géographique (emplacement des utilisateurs finaux), la bande passante, les exigences en matière de stratégie ou de législation, la sécurité et le coût sont des caractéristiques susceptibles de différer d'un fournisseur à l'autre. Un cloud hybride offre une flexibilité permettant de s'adapter et de réagir aux services offerts par ces fournisseurs, et ce, à la demande [7].

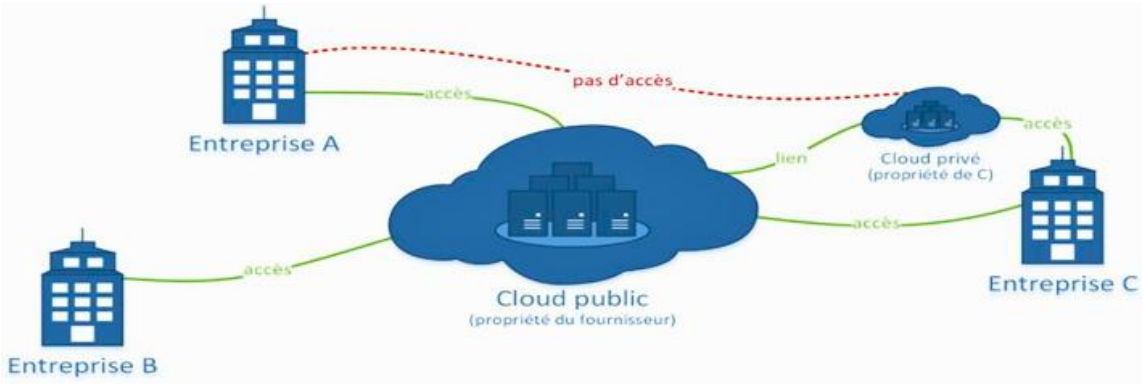

Figure I. 7: Modèle de déploiement d'un cloud hybride.

Le tableau I.1 montre une comparaison entre les types de déploiement en termes d'évolutivité, de fiabilité, de sécurité, de performances et de coût. De plus, quelques exemples sont présentés pour chaque type [7].

Tableau I. 1: comparaison entre Cloud public, privé, hybride et communautaire.

Paramètre	Cloud public	Cloud privé	Cloud communautaire	Cloud hybride
L'évolutivité	Plus élevée	Limité	Limité	Plus élevée
Fiabilité	Modérée	Plus élevée	Plus élevée	Moyenne à élevée
Sécurité	Dépend du fournisseur de service	Haut degré de sécurité	Sécurisé	sécurisé
Performance	Faible à moyenne	Bien	Très bien	Bien
Coût	Moins cher	Coût élevé	Cher	Cher
Exemples	AWS, IBM, Bluemix, Google Cloud, Windows Azure	Oracle, IBM, VMware, AmazonVirtual Private Cloud, Google Virtual Private Cloud	AWS Outposts, Azure Stack, Azure Arc, Google Anthos and VMware Cloud on AWS	Solas Community Cloud

2.8 Les modèles de services du cloud Computing

Les modèles de services sont des modèles de références sur lesquels le Cloud Computing est basé. Ceux-ci peuvent être catégorisés en trois modèles de services de base [8], comme indiqué ci-dessous :

- Infrastructure en tant que service **(IaaS)**
- Plate-forme en tant que service **(PaaS)**
- Logiciel en tant que service **(SaaS)**

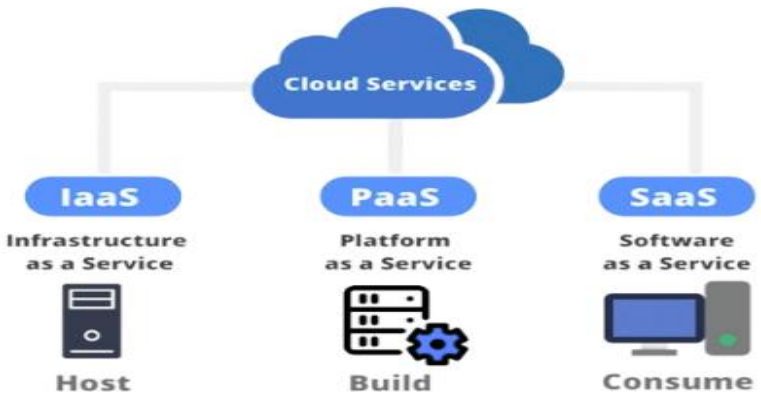

Figure I. 8: Les différentes services du cloud computing.

2.8.1 IaaS (Infrastructure as a Services)

L'Infrastructure as a Service (IaaS) est un modèle de cloud computing qui fournit, via Internet, des ressources informatiques fondamentales telles que des serveurs virtuels, du stockage de données et des composants réseau [8]. L'utilisateur a le contrôle total sur les systèmes d'exploitation, les applications et les configurations installées, tandis que le fournisseur de cloud se charge de gérer l'infrastructure physique sous-jacente. Ce modèle reproduit de manière virtualisée l'environnement informatique traditionnel, facilitant ainsi son adoption par les équipes informatiques et les développeurs [9].

- ***Avantage de (IaaS)***
  - Grande flexibilité.
  - Personnalisation complète de l'environnement selon les besoins.
- ***Inconvénient de (IaaS)***
  - Besoin d'administrateurs système comme pour les solutions de serveurs classiques sur site.
  - Besoin de sécurité.

2.8.2 PaaS (Platform as a Services)

PaaS élimine la nécessité de gérer l'infrastructure sous-jacente, comme le matériel et les systèmes d'exploitation, en permettant à l'utilisateur de se concentrer sur le déploiement et la gestion de ses applications. Cela améliore l'efficacité, car l'utilisateur n'a pas à se soucier de l'approvisionnement des ressources, de la planification de la capacité, de la maintenance des logiciels, des mises à jour ou d'autres tâches complexes liées à l'exploitation de l'application [8].

Ce service fournit un environnement complet pour le cycle de vie des applications, incluant les outils nécessaires à leur développement, déploiement et administration, sans gérer directement l'infrastructure [9].

- **Avantage de (PaaS)**
  - Le déploiement est automatisé et gestion simplifiée.
  - Pas de logiciel supplémentaire à acheter ou à installer.

- **Inconvénient de (PaaS)**
  - Limitation à une ou deux technologies (ex. Python ou Java).
  - Pas de contrôle des machines virtuelles.

  Exemple : hébergement des sites web.

2.8.3 SaaS (Software as a Service)

Le Software as a Service (SaaS) est un modèle de cloud computing qui fournit aux utilisateurs des applications prêtes à l'emploi accessibles via Internet. Contrairement aux applications Web classiques, les services SaaS offrent un haut niveau d'abstraction et peuvent être adaptés à des besoins spécifiques sans installation locale. L'utilisateur n'a pas à gérer ni l'infrastructure ni les mises à jour : tout est pris en charge par le fournisseur [9].

- **Avantage de (Saas)**
  - Plus d'installation requise sur les appareils des utilisateurs.
  - Pas besoin d'acheter de licences logicielles individuelles.
  - Migration de données.
  - Paiement à l'usage.
- **Inconvénient de (Saas)**
  - Limitation du logiciel proposé aux options proposées par le fournisseur.
  - Pas de contrôle sur le stockage et la sécurisation des données associées au logiciel.

  Exemple : messagerie, sauvegarde en ligne.

Une illustration [9] permet de visualiser clairement la répartition des responsabilités entre l'utilisateur et le fournisseur dans les modèles IaaS, PaaS et SaaS.

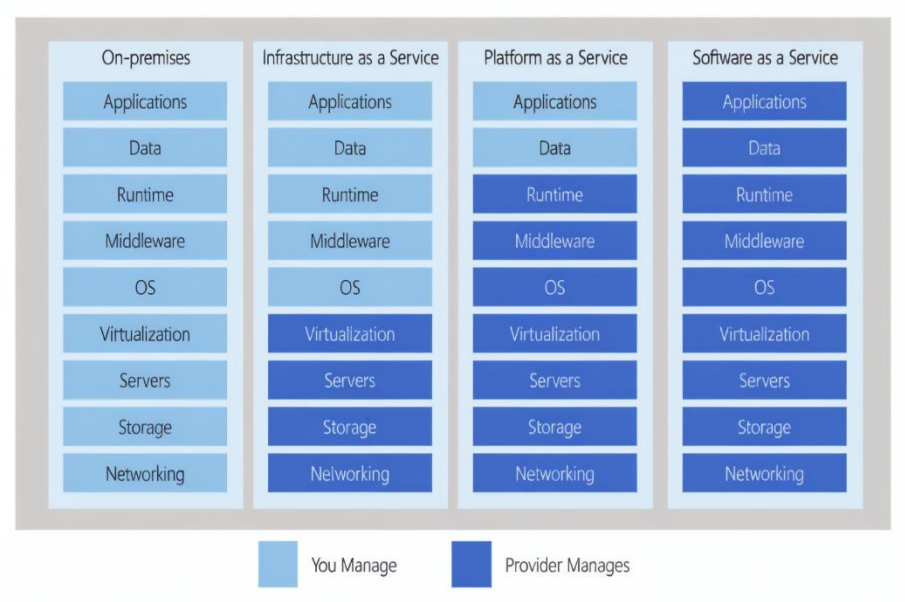

Figure I. 9: Les différentes services du cloud computing.

2.9 Comment choisir l'infrastructure cloud

Le choix entre SaaS, PaaS et IaaS dépend principalement des besoins spécifiques de l'organisation ainsi que des compétences techniques disponibles en interne. Chaque modèle offre un équilibre différent entre facilité de gestion et contrôle [10].

- ✓ **SaaS** est adapté aux organisations qui souhaitent une solution clé en main, sans avoir à gérer l'infrastructure ni les mises à jour. Il convient particulièrement aux entreprises qui privilégient la simplicité et ne disposent pas forcément d'une équipe IT dédiée.
- ✓ **PaaS** s'adresse aux organisations qui veulent développer des applications personnalisées tout en déléguant la gestion de l'infrastructure au fournisseur cloud. Cela permet de garder le contrôle sur les fonctionnalités et les données tout en réduisant la charge liée à la maintenance des ressources.
- ✓ **IaaS** convient aux entreprises disposant d'une équipe IT expérimentée capable de gérer et maintenir l'infrastructure. Ce modèle offre un contrôle total sur les systèmes et configurations, mais demande plus d'investissement en termes de gestion.

2.10 Caractéristiques du cloud Computing

Le cloud computing, tel que défini par le National Institute of Standards and Technology (NIST) dans sa publication officielle [11], repose sur cinq caractéristiques essentielles qui le distinguent des modèles informatiques traditionnels [10].

a) **Libre-service à la demande :** Les utilisateurs peuvent accéder automatiquement aux ressources informatiques (comme les serveurs, le stockage ou la puissance de calcul) sans nécessiter l'intervention humaine du fournisseur de service. Cela permet une grande autonomie dans la gestion des ressources.

b) **Accès réseau étendu :** Les services cloud sont accessibles via le réseau à travers des interfaces standards, compatibles avec divers types d'équipements tels que les ordinateurs, tablettes et smartphones, assurant une disponibilité ubiquitaire.

c) **Mutualisation des ressources :** Les ressources physiques et virtuelles du fournisseur sont regroupées afin de servir plusieurs clients selon un modèle multi-locataire. Les ressources sont allouées de manière dynamique en fonction de la demande, assurant une optimisation des coûts et des performances.

d) **Élasticité rapide :** Le cloud offre une capacité d'adaptation quasi instantanée à la variation des besoins. Les ressources peuvent être augmentées ou réduites automatiquement et rapidement, donnant l'illusion d'une disponibilité illimitée pour les utilisateurs.

e) **Service mesuré (pay-as-you-use) :** L'utilisation des ressources est automatiquement surveillée, contrôlée et enregistrée. Cette caractéristique assure une transparence entre le fournisseur et le client, permettant une facturation précise en fonction de la consommation réelle.

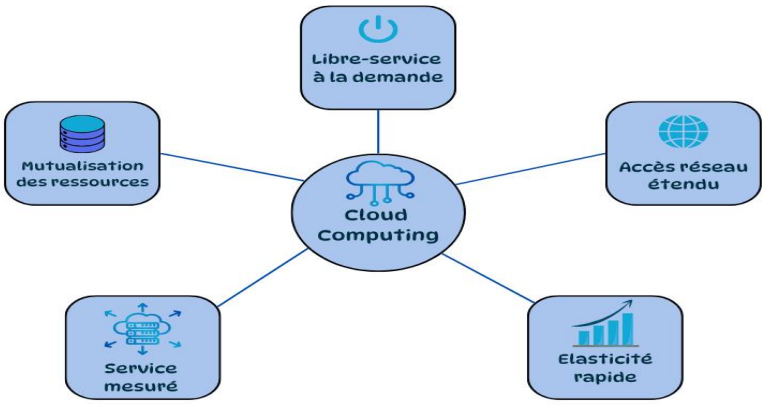

Figure I. 10: Caractéristiques du cloud Computing.

3 L'architecture d'une infrastructure cloud

L'architecture du Cloud Computing désigne la structure et l'organisation des composants nécessaires à la fourniture de services informatiques via Internet.

Elle s'appuie sur plusieurs éléments fondamentaux qui garantissent l'accès à des ressources informatiques à la demande, tout en masquant la complexité de l'infrastructure sous-jacente aux utilisateurs [12].

Ces éléments comprennent : [12]

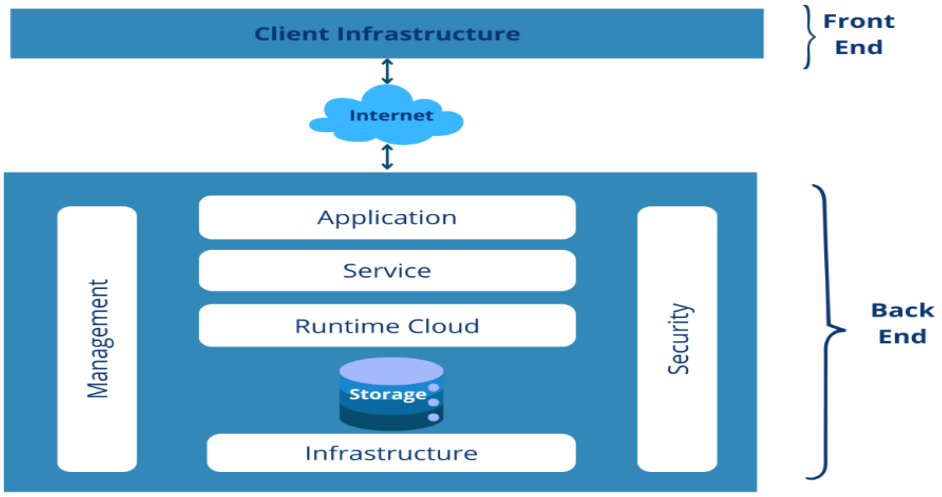

Figure I. 11: L'architecture d'une infrastructure cloud.

- **Interface utilisateur (Front-end) :**

C’est la partie visible pour les utilisateurs finaux. Elle comprend les interfaces web, applications mobiles ou tableaux de bord à travers lesquels les utilisateurs interagissent avec les services cloud.

- **Infrastructure (Back-end) :**

C’est le cœur du système cloud. Elle comprend les serveurs, bases de données, systèmes de stockage et autres composants qui traitent les demandes des utilisateurs et gèrent les ressources informatiques.

- **Réseau :**

Il permet la communication entre le front-end et le back-end, souvent via Internet. C’est par ce réseau que transitent les données et les requêtes entre les utilisateurs et l’infrastructure cloud.

- **Plateforme de livraison des services :**

C’est le composant qui assure la fourniture des services cloud comme le SaaS (Software as a Service), PaaS (Platform as a Service) ou IaaS (Infrastructure as a Service). Il gère la distribution, la sécurité et le déploiement des services vers les utilisateurs.

4 Les technologies du Cloud Computing

Certaines technologies fonctionnent derrière les plates-formes du cloud computing, rendant le cloud computing flexible, fiable et utilisable. Ces technologies sont énumérées ci-dessous :

- Virtualisation.
- Automatisation.

4.1 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 [13]. 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 [12] [14].

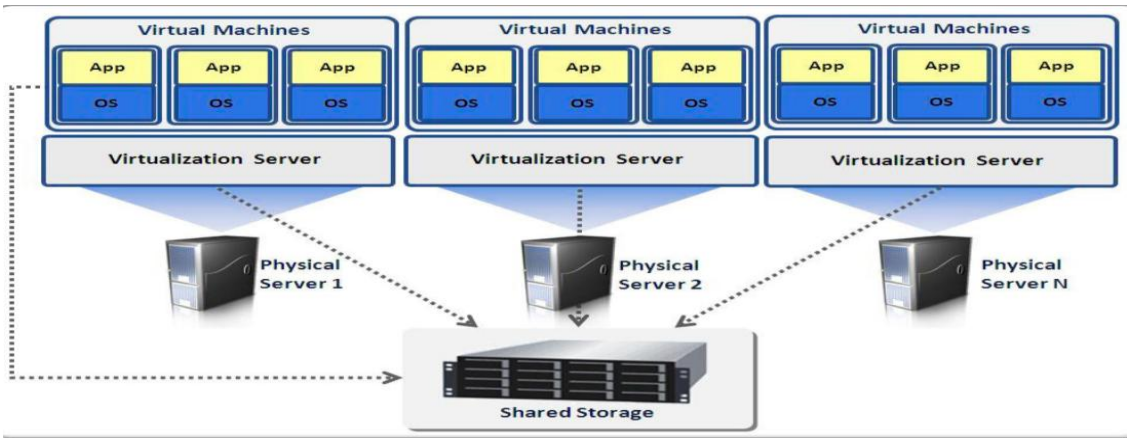

Figure I. 12: Schéma de virtualisation des serveurs utilisant un stockage partagé.

4.2 Automatisation

L'automatisation le processus consistant à fournir des ressources cloud, y compris les serveurs et le stockage connectés par les réseaux, sans intervention manuelle, permet aux utilisateurs d'accéder et de déployer ces ressources à la demande, sans obstacles techniques ou organisationnels tels que des demandes d'approbation.

Son importance réside dans la suppression des délais liés aux processus organisationnels, l'accélération du déploiement des services et l'optimisation de l'utilisation des ressources grâce à une orchestration intelligente qui adapte automatiquement les charges de travail aux capacités matérielles disponibles [15].

4.3 Les avantages et les limites du cloud Computing

4.3.1 Les avantages du cloud Computing

L'adoption du cloud computing s'explique par une série d'avantages stratégiques et opérationnels qui répondent aux exigences modernes des systèmes d'information [13] [16] [17]

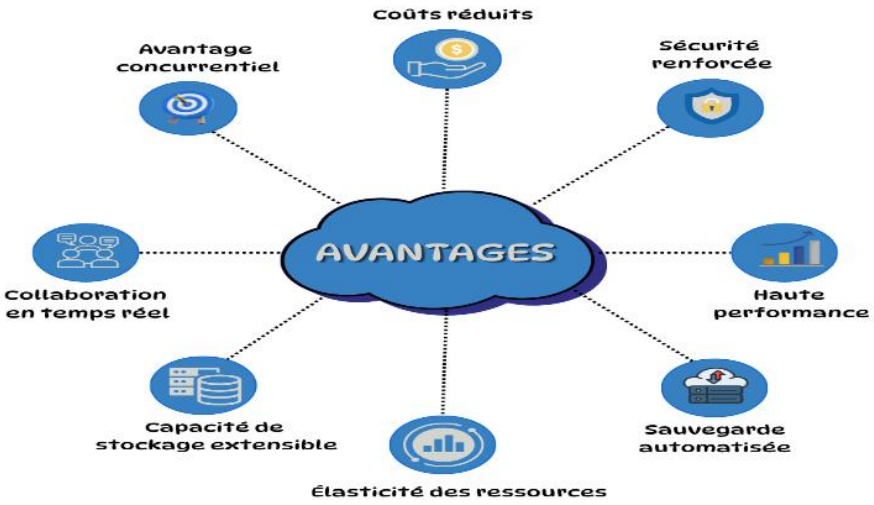

Figure I. 13: Les huits avantages du cloud computing.

1. **Coûts réduits :** Moins d'investissements en infrastructure grâce à l'accès distant aux ressources.
2. **Sécurité renforcée :** Protocoles avancés (chiffrement, pare-feu, surveillance).
3. **Haute performance :** Disponibilité élevée, faible latence, mise à jour continue.
4. **Sauvegarde automatisée :** Protection des données via des systèmes intégrés de Disaster Recovery.
5. **Élasticité des ressources :** Adaptation dynamique de la capacité selon la demande.
6. **Capacité de stockage extensible :** Volume quasi illimité, accessible à tout moment.
7. **Collaboration en temps réel :** Partage instantané des données et travail simultané.
8. **Avantage concurrentiel :** Innovation rapide et réduction du time-to-market.

4.3.2 Les limites du cloud Computing

Comme toute technologie, le Cloud Computing présente certaines limites qu'il est important de considérer pour une adoption efficace [18] [19]:

1. **Consommation énergétique élevée :** Optimisation énergétique des centres de données selon les principes du Green IT.
2. **Risques en matière de sécurité :** Exposition aux menaces (vols, fuites, cyberattaques) sur infrastructures mutualisées.

3. **Dépendance au fournisseur (Vendor Lock-in) :** Verrouillage lié aux technologies propriétaires limitant portabilité et migration des données
4. **Complexité de gestion :** Orchestration et administration des ressources cloud (VM, stockage, réseaux) nécessitant compétences avancées.
5. **Manque de personnalisation :** Offres cloud standardisées peu adaptées aux exigences spécifiques des organisations.
6. **Dépendance à la connectivité Internet :** Nécessité d'une connexion réseau stable pour garantir la disponibilité des services.

5 Conclusion

Le Cloud Computing ne se limite plus à une simple évolution technologique : il redéfinit en profondeur la manière dont les organisations conçoivent, consomment et gèrent l'informatique. Ce chapitre a mis en lumière la diversité et la richesse de ses modèles, de ses architectures et de ses services, tout en soulignant la rupture que cette approche introduit face aux paradigmes traditionnels. Au-delà de l'agilité et de la réduction des coûts, le cloud impose un nouveau rapport à l'infrastructure : la virtualisation et l'automatisation deviennent les piliers d'un écosystème où la flexibilité prime sur la possession. Cette mutation s'accompagne cependant de défis inédits, qu'il s'agisse de sécurité, de souveraineté des données ou de complexité de gestion.

À l'heure où l'informatique en nuage s'impose comme le socle des innovations futures, la question n'est plus de savoir s'il faut adopter le cloud, mais comment en exploiter tout le potentiel tout en maîtrisant ses risques. Dans cette perspective, la suite de ce travail s'attachera à explorer les stratégies d'optimisation et les solutions intelligentes, en s'appuyant aussi bien sur des méthodes exactes que sur des approches métaheuristiques avancées. Ces différentes techniques se révèlent indispensables pour faire du cloud un levier de performance durable et relever efficacement les défis liés à la gestion et à l'allocation des ressources dans des environnements de plus en plus complexes.

Chapitre II : *Méthodes de résolution*

1 Introduction

L'optimisation occupe une place centrale dans de nombreux domaines scientifiques et industriels, où la recherche de solutions efficaces à des problèmes complexes est essentielle. Face à la croissance de la taille et de la complexité des problèmes, il devient crucial de disposer de méthodes de résolution performantes et adaptées.

On distingue principalement deux grandes familles : les méthodes exactes, qui garantissent l'optimalité de la solution mais sont limitées par leur complexité, et les méthodes approchées, qui privilégient la rapidité et la flexibilité au détriment de l'optimalité garantie.

Ce chapitre présente une synthèse des principales méthodes de résolution, en mettant en lumière leurs principes, avantages et limites respectifs. Une attention particulière est portée aux métaheuristiques, notamment à l'algorithme Shark Smell Optimization (SSO), qui constitue l'axe central de ce travail.

2 Les Méthodes de Résolution

Comme mentionné précédemment, on distingue principalement deux grandes familles [11]

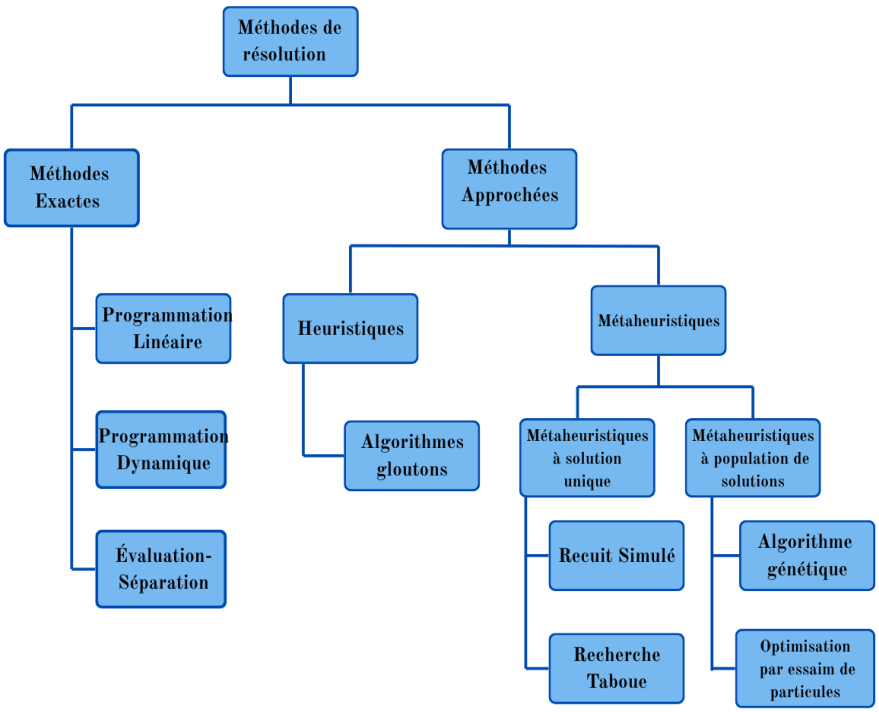

Figure II. 1: Classification des différentes méthodes pour résoudre les problèmes d'optimisation.

2.1 Méthodes exactes

Les méthodes exactes (également qualifiées de complètes) garantissent l'obtention d'une solution optimale pour un problème d'optimisation donné. Elles s'appuient principalement sur une exploration arborescente et une énumération partielle de l'espace des solutions, combinant des stratégies de séparation et d'évaluation systématiques [12].Parmi leurs implémentations les plus notables :

a) Programmation linéaire :

Modélisation polyédrale et résolution via l'algorithme du simplexe (George Dantzig)) et dualité, efficace pour les problèmes convexes à grande échelle (O ($n^{3.5}$) en pratique).

b) Programmation dynamique :

Décomposition récursive fondée sur le principe d'optimalité de Bellman, avec mémorisation des sous-solutions pour les problèmes à sous-structure optimale (ex. : sac à dos multidimensionnel).

c) Séparation-évaluation (Branch and Bound) :

Élagage de l'arbre de recherche par calcul de bornes inférieures/supérieures, souvent accéléré par des solveurs SAT modernes [13].

Le problème majeur de ces méthodes est l'explosion du nombre de combinaisons (appelée "explosion combinatoire") : plus le problème est grand, plus le temps de calcul explose. Elles ne fonctionnent bien que pour les petits problèmes.

2.2 Méthodes approchées

Les méthodes approchées s'opposent aux méthodes exactes en sacrifiant l'optimalité garantie au profit d'une résolution rapide, offrant des solutions de qualité raisonnable (souvent proches de l'optimum) en temps polynomial, ce qui les rend indispensables pour les problèmes complexes de grande échelle [11].

a) **Heuristique :**

Le mot heuristique (du grec ***εὑρίσκειν/heuriskein***, « trouver »). Une heuristique est un algorithme qui permet de trouver dans un temps polynomial une solution réalisable, tenant en compte d'une fonction objectif, pas nécessairement optimale (approchée) ou exacte pour un problème d'optimisation difficile.Ce type de méthodes traduit une stratégie (une manière de penser) en s'appuyant sur la connaissance du problème. Une heuristique est spécifique au problème et ne peut pas être généralisée [14]. Parmi les heuristiques les plus utilisées figurent : les **algorithmes gloutons** qui construisent la solution étape par étape en faisant à chaque itération le choix local optimal (ex. : sélectionner l'objet au meilleur rapport valeur/poids pour le sac à dos), sans jamais revenir en arrière (*sans backtracking*).

b) Métaheuristique :

Le terme métaheuristique combine deux racines grecques : (Méta(μετά) : Au-delà ou niveau supérieur et Heuristique(εὑρίσκειν/heuriskein) : Trouver.) [15]. Les métaheuristiques sont des méthodes généralement inspirées de la nature. Contrairement aux heuristiques, elles s'appliquent à plusieurs problèmes de nature différentes. Pour cela on peut dire qu'elles sont des heuristiques modernes, de plus haut niveau, dédiées particulièrement à la résolution des problèmes d'optimisation. Leur but est d'atteindre un optimum global tout en échappant les optima locaux. Les métaheuristiques regroupent des méthodes qui peuvent se diviser en deux classes [16] :

1. **Métaheuristiques à solution unique :** Ces méthodes traitent une seule solution à la fois, afin de trouver la solution optimale. Les exemples les plus connus sont :

   - **Recuit simulé** (*Simulated Annealing*) : inspiré du processus de refroidissement des métaux, il permet d'accepter temporairement des solutions moins bonnes pour échapper aux optima locaux.
   - **Recherche taboue** (*Tabu Search*) : utilise une mémoire à court terme pour éviter de revisiter des solutions déjà explorées et ainsi diversifier la recherche.

2. **Métaheuristiques à population de solutions :** Ces méthodes utilisent une population de solutions à chaque itération jusqu'à l'obtention de la solution globale. Les exemples les plus connus sont :

- **Algorithme génétique** : s'inspire des mécanismes de sélection naturelle (croisement, mutation, sélection) pour faire évoluer la population vers de meilleures solutions.
- **Optimisation par essaim de particules** (*Particle Swarm Optimization*) : modélise le comportement collectif d'animaux (oiseaux, poissons) pour guider la recherche vers les zones prometteuses de l'espace des solutions.

Le tableau II.1 ci-dessous synthétise les principales différences entre les heuristiques et les métaheuristiques [17]**:**

Tableau II. 1: les principales différences entre les heuristiques et les métaheuristiques.

Critère	Heuristiques	Métaheuristiques
Spécificité	Conçues pour un problème unique	Génériques, adaptables à plusieurs problèmes
Optimalité	Solution acceptable, non optimale	Recherche d'un optimum global
Mécanismes	Règles déterministes	Stochastique + apprentissage
Complexité	Temps polynomial	Temps plus long (mais contrôlé)
Inspiration	Connaissance métier	Nature (essaims, évolution)

Parmi les différentes métaheuristiques exposées précédemment, notre choix s'est porté sur l'algorithme **Shark Smell Optimization (SSO)**, que nous avons retenu pour l'étudier et l'appliquer au problème d'ordonnancement des tâches dans un environnement de cloud computing. Cette sélection s'appuie sur les atouts que présente cet algorithme, notamment sa simplicité conceptuelle, sa capacité à éviter les optima locaux, ainsi que son efficacité dans l'exploration de l'espace de recherche.

3 Shark Smell Optimization (SSO)

L'optimisation par odeur de requin (Shark Smell Optimization, SSO) est un nouvel algorithme métaheuristique qui reproduit la stratégie de chasse des requins guidée par leur olfaction ultra-sensible : chaque solution est assimilée à un « requin » qui suit un gradient de concentration chimique (valeur de la fonction objectif) pour se rapprocher de la meilleure zone, tout en effectuant des rotations locales en spirale afin d'explorer finement son voisinage et d'éviter les minima locaux. Grâce à ce double mécanisme d'exploitation directionnelle et d'exploration locale, enrichi d'une mise à jour adaptative des vitesses et intensités de déplacement, le SSO garantit à la fois une convergence rapide vers l'optimum global et une robustesse face aux pièges du paysage de recherche [18].

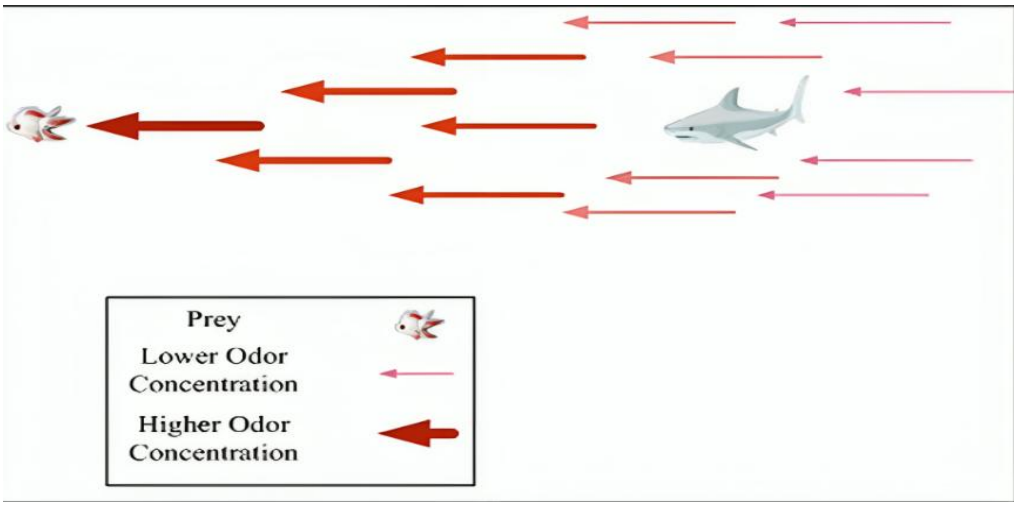

Figure II. 2: Schéma du déplacement du requin vers la source de l'odeur.

4 L'idée et l'inspiration de l'algorithme

4.1 Projection biologique vers une modélisation mathématique dans l'algorithme SSO

L'algorithme Shark Smell Optimization (SSO) trouve son origine dans un comportement naturel bien précis : la capacité des requins à localiser une proie grâce à leur système olfactif très développé. Cette base biologique est projetée dans un modèle mathématique pour construire un mécanisme d'optimisation efficace. Voici comment cette analogie est formalisée :

4.1.1 Source fixe de l'odeur (la proie) : Meilleure solution actuelle

Biologiquement, un requin suit l'odeur d'une proie blessée, supposée immobile. En SSO, cela correspond à la meilleure solution connue dans l'espace de recherche, vers laquelle les autres agents sont attirés.

4.1.2 Concentration d'odeur plus forte : Meilleure valeur de la fonction objective

La concentration chimique perçue par le requin est assimilée à la valeur numérique de la fonction objectif. Plus cette valeur est « bonne » (selon qu'il s'agit d'un minimum ou d'un maximum), plus elle attire les autres agents.

4.1.3 Détection directionnelle (narine gauche vs droite) : Calcul du gradient directionnel Sdgds

Le requin utilise la différence d'intensité entre ses deux narines pour ajuster sa direction. Cette idée est traduite en un vecteur directionnel orienté vers la meilleure solution connue.

4.1.4 Mouvement vers la source : Mise à jour de la position

Chaque solution (requin) met à jour sa position dans l'espace en fonction de sa vitesse actuelle et de la direction de la meilleure solution.

4.1.5 Recherche locale (spirale) : Perturbation contrôlée autour de la solution

La rotation du requin autour de la cible pour affiner sa position est modélisée mathématiquement par un mouvement circulaire ou aléatoire local contrôlé autour de la solution actuelle.

4.1.6 Intensité décroissante : Diminution progressive des paramètres de recherche

À mesure que le requin s'approche de la source, ses mouvements deviennent plus fins. De manière équivalente, les paramètres comme la vitesse et l'intensité diminuent progressivement à chaque itération pour affiner la recherche.

Cette projection biologique vers la modélisation mathématique constitue le socle du fonctionnement de l'algorithme SSO. Elle permet de capturer un comportement adaptatif naturel et de l'exploiter dans des processus d'optimisation complexes, tout en conservant une structure algorithmique simple et efficace.

5 Hypothèses de modélisation de l'algorithme SSO

Afin d'établir la formulation mathématique de l'algorithme Shark Smell Optimization (SSO), certaines hypothèses simplificatrices sont posées pour modéliser l'environnement et le comportement du requin dans l'espace de recherche :

Source fixe : La proie, représentée par un poisson blessé, est considérée comme immobile. Sa vitesse est négligeable face à celle du requin, ce qui permet de la modéliser comme une source statique dans l'espace de recherche.

Diffusion régulière de l'odeur : L'injection de particules odorantes (sang) dans l'eau est supposée continue et homogène. Les effets de dispersion dus aux courants marins sont ignorés. La concentration est plus élevée à proximité de la source, facilitant la progression du requin vers la proie en suivant le gradient de concentration.

Source unique : On considère l'existence d'une seule proie blessée, donc d'une unique source d'odeur. Cette simplification permet de modéliser un seul objectif global dans l'environnement de recherche.

5.1 Initialisation de SSO : génération des particules initiales d'odeur

Le processus de recherche débute lorsque le requin détecte une odeur. En réalité, les particules odorantes diffusées par un poisson blessé (proie) ont une intensité faible et progressive.

Pour modéliser ce phénomène, une population de solutions initiales est générée aléatoirement dans l'espace de recherche faisable.

Chaque solution représente une particule odorante, c'est-à-dire une position potentielle du requin au début du processus.

Cette population initiale peut être représentée sous la forme suivante :

$$\left[x_1^1, x_2^1, \dots, x_{NP}^1 \right] \quad \textbf{(1)}$$

Où $x_{i,1}^1$1représente la position initiale de la i ième solution, et **NP** désigne la taille de la population. Le problème d'optimisation associé peut être représenté comme suit :

$$x_i^1 = \left[x_{i,1}^1, x_{i,2}^1, \dots, x_{i,ND}^1 \right] \quad i = 1, 2, \dots, NP \qquad (2)$$

ND représente le nombre total de variables de décision dans le problème d'optimisation

$x_{i,1}^1$ Désigne la **j** *ième* dimension de la position du ***i*** *ième* requin (ou encore la **j** *ième* variable de décision associée a cette position)

L'intensité de l'odeur perçue à chaque position traduit la proximité du requin par rapport à la proie.

Ce phénomène est modélisé dans l'algorithme SSO (Shark Smell Optimization) à l'aide d'une fonction objectif.

Dans le cas d'un problème de maximisation, une valeur plus élevée de la fonction objective indique une odeur plus forte, c'est-à-dire une position plus proche de la proie.

5.2 Mouvement du requin vers la proie :

Le requin à chaque position se déplace avec une vitesse pour se rapprocher de la proie.

En se basant sur les vecteurs de position, le vecteur de vitesse initial peut être défini comme suit :

$$\left[V_1^1, V_2^1, \cdots, V_{NP}^1 \right] \quad (3)$$

Comme indiqué dans l'équation **(3)**, les vecteurs vitesse possèdent des composantes dans chaque dimension.

$$V_i^1 = \left[V_{i,1}^1, V_{i,2}^1, \dots, V_{i,ND}^1 \right] \quad i = 1, \dots, ND \qquad (4)$$

Le requin suit l'odeur, et la direction de son déplacement est déterminée par l'intensité de cette odeur.

Plus l'odeur est concentrée, plus la vitesse du requin augmente.

D'un point de vue optimisation, ce déplacement est modélisé par le gradient de la fonction objectif.

Le gradient indique la direction dans laquelle la fonction croît le plus rapidement.

Cette dynamique est représentée par l'équation suivante :

$$\begin{cases} V_i^k = \eta_k \cdot R1 \cdot \nabla(OF)|_{x_i^k} \\ i = 1, \dots, NP \quad k = 1, \dots, k_{max} \end{cases} \quad \textbf{(5)}$$

V_i^k: vitesse du requin, considérée comme approximativement constante

OF: fonction objective.

∇: gradient de la fonction objectif

η_k: facteur de réduction

$R1$: valeur aléatoire uniformément distribuée

k_{max}: nombre maximal d'itérations

- Le paramètre η_k est compris entre 0 et 1, car il se peut qu'un requin ne puisse pas atteindre la vitesse déterminée uniquement par le gradient.
- Le paramètre aléatoire ***R1*** permet d'introduire davantage de hasard dans la recherche, ce qui est une caractéristique propre à l'algorithme SSO.
- L'idée d'utiliser ***R1*** est inspirée de l'algorithme de recherche gravitationnelle (GSA).

La vitesse dans chaque dimension peut être calculée selon l'équation suivante :

$$\begin{cases} V_{i,j}^k = \eta_k \cdot R1 \cdot \frac{\partial(OF)}{\partial x_j}\Big|x_{i,j}^k \\ i = 1, \dots, NP \quad j = 1, \dots, ND \quad k = 1, \dots, k_{max} \end{cases} \quad \textbf{(6)}$$

En raison de l'existence d'une inertie, l'accélération du requin est limitée, et sa vitesse dépend de sa vitesse précédente.

Ce comportement est modélisé par une version modifiée de l'équation **(6),** donnée comme suit :

$$\begin{cases} V_{i,j}^{k} = \eta_k \cdot R1 \cdot \frac{\partial(OF)}{\partial x_j}\Big|x_{i,j}^{k} + \alpha_k \cdot R2 \cdot V_{i,j}^{k-1} \\ i = 1, \dots, NP \quad j = 1, \dots, ND \quad k = 1, \dots, k_{max} \end{cases} \quad (7)$$

Où :

α_k: coefficient d'inertie ou de momentum, appartenant à l'intervalle [0,1], et fixé pour chaque étape k.

R2 : variable aléatoire uniformément distribuée dans [0,1], utilisée pour pondérer le terme d'inertie.

- Une valeur élevée de α_k signifie une plus grande inertie, ce qui rend la vitesse actuelle plus influencée par la vitesse précédente.

Sur le plan mathématique, l'introduction du momentum permet de rendre les trajectoires de recherche plus fluides dans l'espace de solutions.

Le terme ***R2*** contribue à augmenter la diversité de la recherche.

Pour la vitesse au premier pas ($V_{i,j}^{1}$) , il est possible soit de négliger la vitesse initiale du requin, soit de lui attribuer une valeur très faible ($V_{i,j}^{0}$).

La vitesse du requin peut être augmentée jusqu'à une certaine limite.

Contrairement aux poissons, les requins ne possèdent pas de vessie natatoire pour rester en flottaison.

Ils ne peuvent donc pas rester immobiles et doivent nager en direction ascendante, même à faible vitesse.

Ce mouvement est rendu possible grâce à leur nageoire caudale puissante, qui joue le rôle de propulseur.

En moyenne, la vitesse normale d'un requin est d'environ 20 km/h, et elle peut atteindre 80 km/h lorsqu'il attaque.

Le rapport entre la vitesse maximale et la vitesse minimale est donc limité (par exemple : 80/20=4).

Pour tenir compte de cette contrainte dans l'algorithme SSO, une limitation de vitesse est appliquée à chaque étape. Elle est modélisée par l'expression suivante :

$$\begin{cases} |V_{i,j}^{k}| = \min\left[\, |\eta_k \cdot R_1 \cdot \left(\frac{\partial(OF)}{\partial x_j}\right)| X_{i,j}^{k} + \alpha_k \cdot R_2 \cdot V_{i,j}^{k-1}| \,, |\beta_k \cdot V_{i,j}^{k-1}| \right] \\ i = 1, \dots, NP \quad j = 1, \dots, ND \quad k = 1, \dots, k_{max} \end{cases} \quad \textbf{(8)}$$

β_k Est le coefficient de limitation de vitesse à l'étape k.

- La valeur de $V_{i,j}^{k}$ est calculée par l'équation **(8)** et conserve le même signe que le terme sélectionné par l'opérateur min dans cette même équation.
- Lors du mouvement vers l'avant, la nouvelle position du requin à l'étape **k+1** notéeY_i^{k+1}, est déterminée à partir de sa position précédente et de sa vitesse :

$$\begin{cases} Y_i^{k+1} = X_i^{k} + V_i^{k} \cdot \Delta t_k \\ i = 1, \dots, NP \quad k = 1, \dots, k_{max} \end{cases} \quad \textbf{(9)}$$

Où :

Δt_k**:** correspond à l'intervalle de temps de l'étape k, supposé égal à 1 pour toutes les étapes, afin de simplifier le modèle.

- Chaque composante $V_{i,j}^{k}$ du vecteur de vitesse est calculée à l'aide de l'équation **(8)** avec **j=1,…,ND**.

En plus de son mouvement vers l'avant, le requin effectue également un mouvement rotatif dans sa direction, afin de détecter des sources d'odeur plus intenses.

Ce comportement améliore sa progression, comme illustré dans la figure II.3

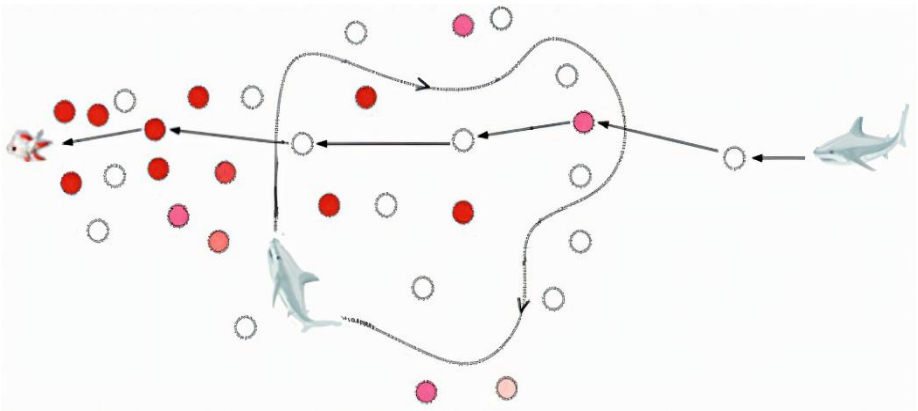

Figure II. 3: Mouvement de rotation d'un requin.

Ce mouvement rotatif s'effectue autour d'un contour fermé, qui n'est pas nécessairement un cercle.

Du point de vue de l'optimisation, ce mouvement correspond à une recherche locale effectuée à chaque étape pour explorer de meilleures solutions.

Ce mécanisme de recherche locale dans l'algorithme SSO est modélisé comme suit :

$$\begin{cases} Z_i^{k+1,m} = Y_i^{k+1} + R3 \cdot Y_i^{k+1} \\ m = 1, \dots, M \quad i = 1, \dots, NP \quad k = 1, \dots, k_{max} \end{cases} \qquad \textbf{(10)}$$

Où :

$Z_i^{k+1,m}$ Désigne la position du point m dans la recherche locale.

R3 : est un facteur aléatoire introduisant une légère perturbation autour de la position, Y_i^{k+1} simulant ainsi l'exploration locale.

M : représente le nombre de points évalués dans la recherche locale à chaque étape.

- Étant donné que cet opérateur effectue une recherche locale autour de Y_i^{k+1} , la valeur de ***R3*** est limitée à l'intervalle [−1,1].
- Les ***M*** points de la recherche locale $Z_i^{k+1,m}$ se trouvent à proximité de Y_i^{k+1}
  - Par exemple, si le générateur aléatoire retourne zéro, alors$Z_i^{k+1,m} = Y_i^{k+1}$
- Ces ***M*** points forment un contour fermé en se connectant les uns aux autres, simulant ainsi le mouvement rotatif du requin.

- Pendant cette phase, si le requin découvre un point associé à une odeur plus intense, il poursuivra sa recherche à partir de cette nouvelle position.

Ce comportement est modélisé dans l'algorithme SSO par l'équation suivante :

$$\begin{cases} \boldsymbol{x_i^{k+1} = \arg\max \left\{ OF\left(Y_i^{k+1}\right), OF\left(Z_i^{k+1,i}\right), \ldots, OF\left(Z_i^{k+1,M}\right) \right\}} \\ \boldsymbol{i = 1, 2, \ldots, NP} \end{cases} \quad \textbf{(11)}$$

- Autrement dit, on sélectionne, parmi les $\boldsymbol{M}$ points (et le point initial), celui qui maximise la valeur de la fonction objective.
- Dans l'équation **(11)**, la fonction objectif (OF) doit être maximisée. D'une autre façon, parmi les positions obtenues par le déplacement vers l'avant $\boldsymbol{Y_i^{k+1}}$ et celles issues du mouvement rotatif $\boldsymbol{Z_i^{k+1,i}}$pour (m=1,2,…,M), On sélectionne celle qui donne la plus haute valeur de la fonction objectif.
- Cette solution devient alors la nouvelle position du requin $\boldsymbol{X_i^{k+1}}$**.**
- Le cycle entre le mouvement vers l'avant et le mouvement rotatif se répète jusqu'à ce que $\boldsymbol{k = k_{max}}$
- Comme d'autres méthodes d'optimisation méta-heuristiques, l'algorithme SSO utilise plusieurs paramètres à définir par l'utilisateur : $\boldsymbol{NP}$, $\boldsymbol{k_{max}}$, $\boldsymbol{\eta_k}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta_k}$,Ces paramètres peuvent être modifiés dynamiquement durant l'évolution de l'algorithme grâce à un mécanisme adaptatif, efficace dans de nombreuses applications.

La figure II.4 illustre le déroulement général de l'algorithme SSO sous forme de diagramme de flux, mettant en évidence les étapes principales du processus d'optimisation inspiré du comportement olfactif des requins [18].

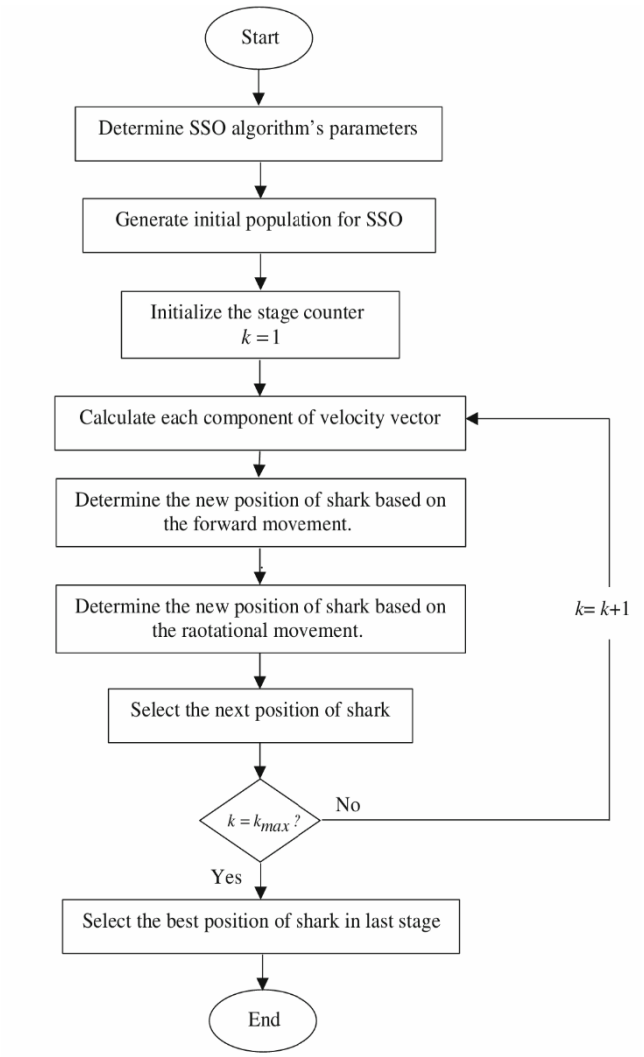

Figure II. 4: Diagramme de flux de l'algorithme (SSO).

6 Le Pseudo-code de SSO

Le pseudocode ci-dessous illustre de manière claire et structurée les étapes principales de l'algorithme Shark Smell Optimization (SSO) [15]. Il permet de comprendre le déroulement séquentiel du processus d'optimisation, depuis l'initialisation de la population jusqu'à la mise à jour itérative des positions, en passant par la gestion des paramètres adaptatifs. Ce formalisme facilite la compréhension de la logique sous-jacente et sert de référence pour toute implémentation pratique de l'algorithme [18].

Begin

Step 1. Initialization

Set parameters NP, k_{max}, η_k, α_k, and β_k $(k=1, 2, ..., k_{max})$

Generate initial population with all individuals

Generate each decision randomly within the allowable range

Initialize the stage counter $k = 1$

For $k = 1 : k_{max}$

Step 2. Forward movement

Calculate each component of the velocity vector, $v_{i,j}$ $(i = 1, ..., NP,\ j = 1, ..., ND)$

Obtain new position of shark based on forward movement, Y_i^{k+1} $(i = 1, ..., NP)$

Step 3. Rotational movement

Obtain position of shark based on rotational movement, $Z_i^{k+1,m}$ $(m = 1, .., M)$

Select next position of shark based on the two movements X_i^{k+1} $(i = 1, ..., NP)$

End for ***k***

Set $k = k+1$

Select the best position of shark in the last stage which has the highest OF value

End

Figure II. 5: Le Pseudo-code de l'algorithme (SSO).

7 Conclusion

Dans ce chapitre, nous avons exploré les différentes approches de résolution appliquées aux problèmes d'optimisation, en distinguant les méthodes exactes des méthodes approchées. Une attention particulière a été portée aux métaheuristiques, qui se révèlent particulièrement efficaces face à la complexité et à la nature non linéaire de nombreux problèmes. Parmi celles-ci, nous avons étudié en détail l'algorithme Shark Smell Optimization (SSO), en présentant son principe biologique d'inspiration, sa modélisation mathématique, ainsi que son fonctionnement

Le prochain chapitre sera consacré à l'application de cet algorithme au problème d'ordonnancement des tâches dans un environnement Cloud.

Chapitre III *: Implémentation de l'application et évaluation des résultats obtenus*

1 Introduction

Dans ce troisième volet, nous présentons la mise en œuvre du Shark Smell Optimization (SSO) couplé à deux méta-heuristiques « la Particle Swarm Optimization (PSO)» et « l'algorithme Bat (BA) ». Ainsi qu'à une méthode de somme pondérée, pour optimiser l'ordonnancement des tâches dans un environnement de cloud computing.

L'étude compare les approches selon trois critères de performance : le temps d'exécution, le coût et la consommation énergétique.

Nous décrivons ensuite les outils et environnements choisis pour ce projet : le langage Java et l'IDE NetBeans pour le développement, la bibliothèque JFreeChart pour la visualisation des résultats, et le simulateur CloudSim pour modéliser l'infrastructure cloud.

Enfin, ce chapitre détaille l'interface utilisateur conçue pour simplifier l'interaction avec le système, avant de proposer une analyse comparative approfondie des résultats obtenus par chacune des méthodes multicritères.

2 Fondements des Algorithmes d'Optimisation BA et PSO

2.1 Description de l'Algorithme PSO (Particle Swarm Optimization)

L'algorithme PSO (Particle Swarm Optimization) est une métaheuristique inspirée du comportement collectif des oiseaux et des bancs de poissons. Il a été introduit en 1995 par Kennedy et Eberhart. PSO utilise une population de solutions potentielles appelées particules, qui se déplacent dans l'espace de recherche pour trouver la meilleure solution.

Chaque particule ajuste sa position en fonction de sa propre expérience (meilleure position personnelle) et de celle des autres particules (meilleure position globale). Ce mécanisme de collaboration permet un bon équilibre entre exploration (découverte de nouvelles zones) et exploitation (raffinement des solutions prometteuses) [19].

2.2 Description de l'Algorithme BA (Bat Algorithm)

L'algorithme BA (Bat Algorithm) est une métaheuristique inspirée du comportement d'écholocation des chauves-souris. Proposé par Xin-She Yang en 2010, il simule la manière dont les chauves-souris localisent leurs proies en émettant des signaux ultrasonores.

Chaque chauve-souris représente une solution potentielle qui se déplace dans l'espace de recherche en ajustant sa vitesse, sa fréquence et son taux d'impulsion. Le mouvement est influencé par la meilleure solution trouvée et par des mécanismes aléatoires. L'algorithme équilibre exploration (découverte de nouvelles zones) et exploitation (amélioration des bonnes solutions), ce qui en fait un outil efficace pour résoudre divers problèmes d'optimisation [20].

3 Outils et environnement de simulation et de développement

3.1 Outils de travail

- ✓ Une machine avec un processeur Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz.

3.2 Langage de programmation java :

Java est un langage orienté objet, portable grâce à la machine virtuelle Java (JVM), et reconnu pour sa simplicité, robustesse et sécurité. Il combine une compilation partielle et une interprétation, assurant ainsi un bon compromis entre performance et flexibilité [21].

L'écosystème Java comprend trois éditions principales :

- ✓ Java ME pour les dispositifs mobiles et embarqués.
- ✓ Java SE pour les applications de bureau,
- ✓ Java EE pour le développement d'applications d'entreprise distribuées

Depuis son passage à l'open source en 2006-2008 via le projet OpenJDK, Java bénéficie d'une large communauté et d'une grande disponibilité de ses sources, favorisant l'adaptabilité et l'innovation [22] [23].

3.3 Environnement de développement intégré : NetBeans :

NetBeans est un environnement de développement intégré (IDE) open source, largement utilisé pour le développement d'applications Java. Il se distingue par sa simplicité d'utilisation et son intégration native avec les technologies Java, ce qui facilite la création, l'édition, le débogage et la gestion de projets.

NetBeans offre un ensemble complet de fonctionnalités, incluant des outils graphiques, la prise en charge de multiples langages, et un support étendu pour les frameworks Java

modernes. Sa communauté active contribue régulièrement à son amélioration et à la mise à jour des fonctionnalités.

L'IDE NetBeans est disponible gratuitement et peut être téléchargé depuis son site officiel : https://netbeans.apache.org

3.4 JFreeChart

hJFreeCart est une bibliothèque Java open source et gratuite, conçue pour la création d'une large gamme de graphiques professionnels. Elle supporte divers types de représentations graphiques, incluant les graphiques à secteurs, les graphiques à barres (horizontaux, verticaux, empilés et avec effets 3D), ainsi que les graphiques linéaires, entre autres.

Parmi ses fonctionnalités principales, JFreeChart offre le zoom interactif, la gestion des événements, les infobulles (tooltips) et un accès structuré aux données via des interfaces dédiées. De plus, cette bibliothèque permet l'exportation des graphiques dans plusieurs formats, tels que JPEG, PNG, SVG et PDF, en tirant parti de toute implémentation conforme à l'interface Graphics2D [24].

3.5 Le simulateur CloudSim

3.5.1 Définition

CloudSim est un simulateur open-source conçu pour modéliser des infrastructures et services complexes de Cloud computing. Développé par le laboratoire CLOUDS Lab de l'Université de Melbourne (Australie), il est entièrement écrit en Java. Cet outil fournit un environnement contrôlé et reproductible permettant de tester et valider différentes stratégies telles que le déploiement, l'ordonnancement, le provisionnement et la répartition de charge [25].

CloudSim supporte également la virtualisation, la gestion des connexions réseau et la fédération de plusieurs clouds, ce qui facilite l'évaluation des performances des systèmes Cloud avant leur déploiement réel.

3.5.2 Architecture de CloudSim

Son architecture repose sur une structure en couches [26]:

- ✓ **Couche SimJava (base) :**Cette couche fournit le moteur d'événements discrets. Elle gère la simulation du temps, la communication entre les entités, et le traitement des événements. SimJava est la fondation sur laquelle repose tout le simulateur.
- ✓ **Couche CloudSim Core (cœur) :**Cette couche contient les entités essentielles du cloud, telles que les machines virtuelles (VMs), les hôtes (hosts), les centres de données (datacenters) et les courtiers (brokers). Elle implémente la logique de gestion des ressources, la modélisation de l'infrastructure cloud et les mécanismes d'allocation.
- ✓ **Couche Utilisateur (user layer) :**La couche supérieure permet à l'utilisateur de configurer la simulation, notamment en définissant le nombre de machines, les tâches à exécuter, ainsi que les stratégies d'ordonnancement et de gestion des ressources. C'est ici que sont développés les scénarios spécifiques à chaque expérimentation.

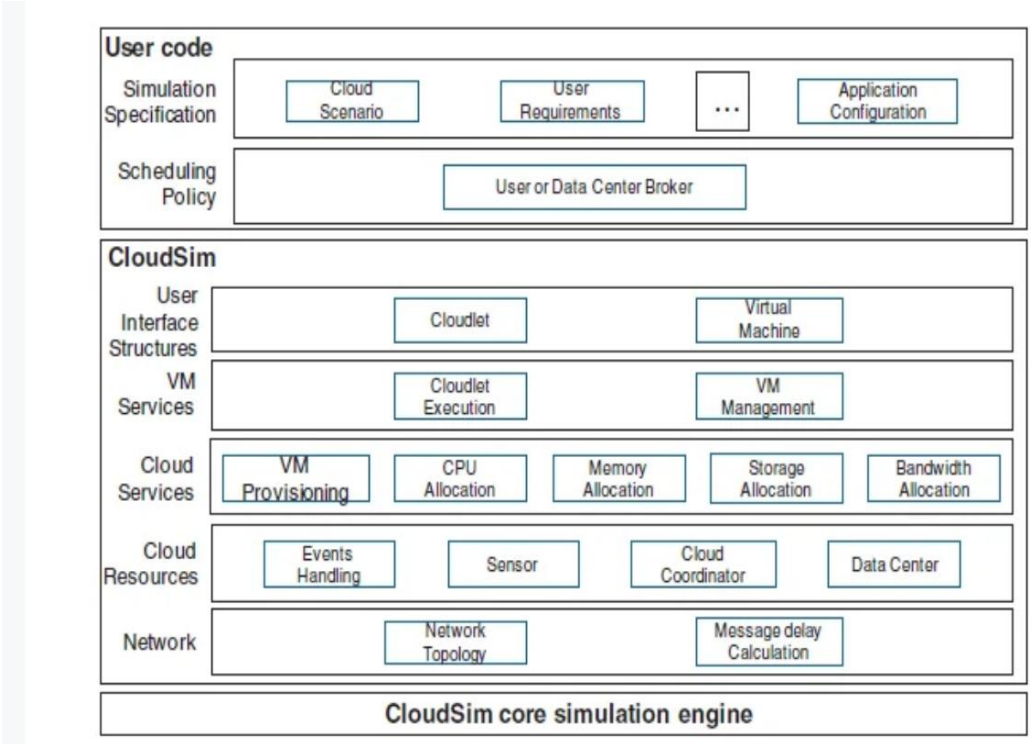

Figure III. 1: L'architecture en couches de CloudSim.

3.5.3 Les différentes classes du CloudSim

La couche CloudSim comprend un ensemble varié de classes essentielles à la modélisation des environnements cloud. Parmi celles-ci [27]:

- ✓ **Centre de données (Data Center) :** Le centre de données constitue une infrastructure virtualisée centralisée, regroupant des ressources physiques telles que les serveurs, le stockage et les réseaux, ainsi que des ressources logicielles, notamment les hyperviseurs et les orchestrateurs. Il représente la fondation matérielle du cloud computing, permettant la mutualisation dynamique des ressources à l'aide d'algorithmes sophistiqués d'allocation (processeur, mémoire vive, bande passante) et de stratégies avancées de gestion énergétique. Les hôtes physiques au sein du centre de données hébergent des machines virtuelles (VM) selon des politiques de provisionnement élastique, intégrant des mécanismes de résilience et d'assurance de haute disponibilité.
- ✓ **DatacenterBroker :** Le DatacenterBroker agit en tant qu'intermédiaire intelligent entre les utilisateurs et les fournisseurs de services cloud. Il analyse les exigences des utilisateurs, telles que la performance ou le coût, et négocie avec les centres de données afin d'allouer les ressources nécessaires aux machines virtuelles. Grâce à l'intégration de techniques d'optimisation, le broker veille à ce que les tâches (cloudlets) soient exécutées de manière efficiente, tout en garantissant le respect des accords de qualité de service (QoS).
- ✓ **Host :** La classe Host représente une machine physique au sein du centre de données et encapsule des informations essentielles, telles que la capacité mémoire, l'espace de stockage, le type et le nombre de processeurs (illustrant la notion de multi-cœur), ainsi que les politiques d'allocation des ressources. Elle définit également les règles de partage de la puissance de calcul, de la mémoire et de la bande passante entre les différentes machines virtuelles hébergées.
- ✓ **VM (Machine Virtuelle) :** La machine virtuelle (VM) simule, au niveau logiciel, le fonctionnement d'un ordinateur autonome exécuté sur un hôte physique. Elle dispose de ressources virtuelles (processeur, mémoire, stockage)

attribuées dynamiquement en fonction des besoins. Les VMs offrent un environnement d'exécution flexible et isolé pour les tâches, avec une tarification basée sur l'utilisation effective des ressources (temps de calcul, consommation CPU). Elles peuvent être temporaires (supprimées après usage) ou persistantes (conservées sur la durée).

- ✓ **Cloudlet :** La classe Cloudlet modélise toute opération exécutée sur une machine virtuelle, qu'il s'agisse de traitement, d'accès mémoire ou d'autres tâches. Elle encapsule des paramètres décrivant les caractéristiques spécifiques de la tâche, tels que la durée, la taille et le nombre de millions d'instructions (MI). De plus, elle fournit des méthodes permettant de déterminer le temps d'exécution, l'état, le coût et l'historique d'exécution de chaque tâche.

3.6 Critères d'évaluation

3.6.1 Makespan

Dans le cadre de l'optimisation des ressources en environnement cloud, le concept de makespan correspond à la durée totale nécessaire pour achever l'exécution d'un ensemble de tâches, cette durée étant déterminée par la machine virtuelle (VM) la plus sollicitée. Cette mesure constitue un indicateur fondamental de la performance des algorithmes d'ordonnancement, car elle reflète l'efficacité de la distribution des charges de travail entre les différentes VMs.

Un makespan élevé traduit une répartition déséquilibrée, susceptible d'engendrer des goulets d'étranglement, tandis qu'un makespan minimisé témoigne d'une allocation plus homogène et optimisée des ressources [28].

Cette notion peut être formalisée par l'équation **(12)** suivante :

$$\boldsymbol{Makespan = max}_{\,1 \le i \le m} \left(\sum_{j=1}^{n} \boldsymbol{T_{ij}}\right) \qquad \textbf{(12)}$$

Où :

$\boldsymbol{T_{ij}}$ Désigne le temps d'exécution de la tâche j sur la machine virtuelle i.

m représente le nombre total de machines virtuelles utilisées,

n correspond au nombre total de tâches à planifier.

3.6.2 Coût :

Les fournisseurs de services cloud adoptent des modèles tarifaires qui incluent des frais pour le transfert de données (généralement facturés par unité de volume, comme le mégaoctet) ainsi que des coûts liés à la consommation des ressources de calcul, souvent facturés en fonction du temps d'utilisation (par exemple, à l'heure). Ces composantes tarifaires constituent la base de la facturation et influencent directement la gestion économique des ressources cloud [29].

Dans le cadre de cette étude, le coût total associé à l'exécution d'un ensemble de tâches réparties sur plusieurs machines virtuelles (VMs) est défini par la somme des coûts individuels de chaque VM, calculés comme le produit du temps d'utilisation de la VM par son tarif unitaire. Cette relation est formalisée par l'équation **(13)** suivante :

$$\textbf{Coût } \boldsymbol{totale} = \sum_{i=1}^{m} \boldsymbol{c_i} \times \boldsymbol{T_i} \qquad \textbf{(13)}$$

Où :

$\boldsymbol{m}$ est le nombre total de machines virtuelles.

$\boldsymbol{c_i}$ est le coût unitaire horaire ou à l'utilisation de la machine virtuelle i.

$\boldsymbol{T_i}$ est la durée totale d'utilisation de la machine virtuelle i.

3.6.3 Énergie d'exécution :

L'évaluation de la consommation énergétique dans les environnements cloud constitue un enjeu majeur pour la conception de systèmes durables et économes en énergie. L'énergie d'exécution représente la quantité d'énergie consommée par les ressources informatiques lors du traitement effectif des tâches. Contrairement au makespan, qui mesure la durée totale d'achèvement des tâches, ou au coût, qui quantifie la dépense financière, l'énergie d'exécution se concentre sur l'impact environnemental et l'efficacité énergétique des infrastructures.

Le calcul de cette énergie repose généralement sur l'agrégation de la consommation énergétique de chaque machine virtuelle (VM), laquelle dépend à la fois du temps d'utilisation et de la puissance moyenne consommée par la VM durant l'exécution des tâches. Formellement, l'énergie totale s'exprime par la relation **(14)** suivante :

$$\text{Énergie } \boldsymbol{totale} = \sum_{i=1}^{m} \boldsymbol{P_i} \times \boldsymbol{T_i} \quad \textbf{(14)}$$

Où :

$\boldsymbol{m}$ désigne le nombre total de machines virtuelles.

$\boldsymbol{T_i}$ est la durée totale d'utilisation de la machine virtuelle i.

$\boldsymbol{P_i}$ est la puissance moyenne consommée par la VM i.

$\boldsymbol{T_i}$ correspond au temps total d'utilisation de la VM i.

3.7 La méthode de la somme pondérée

La méthode de somme pondérée (WSM) est une technique simple d'aide à la décision multicritères, couramment utilisée dans des contextes académiques et pratiques. Sa simplicité le rend adapté à une utilisation quotidienne dans les tâches d'optimisation [30].

3.7.1 Exigences essentielles pour la mise en œuvre :

Critères quantitatifs : Tous les critères doivent être quantifiables numériquement.

Cohérence des unités : Les critères doivent soit utiliser la même unité de mesure, soit être normalisés sur une échelle commune.

Le WSM regroupe les objectifs en une seule fonction à l'aide de la formule :

$$\boldsymbol{f(x) = \Sigma_{i=1}^{n}\, wi.fi} \quad \textbf{(15)}$$

Où wireprésente le poids attribué au critère $\boldsymbol{fi}$

3.7.2 Contraintes de poids :

Non-négativité : Les poids doivent respecter la condition $0 \leq \boldsymbol{wi}_i \leq 1$.

Somme à un : L'addition de tous les poids doit être égale à 1, c'est-à-dire que

$$\boldsymbol{\sum_{i=1}^{n} wi = 1.} \quad \textbf{(16)}$$

3.7.3 Processus de normalisation :

Étant donné que des critères tels que le temps de réalisation (temps), le coût (monnaie) et l'utilisation des ressources (pourcentage) ont souvent des unités incompatibles, leur normalisation s'avère indispensable. L'approche varie selon les objectifs d'optimisation :

Critères de minimisation :

$$\mathbf{X_{normalisé}} = \frac{x - x_{min}}{x_{\max - x_{min}}} \qquad \textbf{(17)}$$

Critères de Maximisation :

$$\mathbf{X_{normalisé}} = \frac{x_{max} - x}{x_{max} - x_{min}} \qquad \textbf{(18)}$$

Fonction d'optimisation multi-objectifs :

La fonction objectif agrégée combinant des critères normalisés est :

$$\boldsymbol{F = w1 * Makespan + w2 * Cout + w3 * energie} \qquad \textbf{(19)}$$

La fonction objectif agrégée combinant des critères normalisés est :

Les priorités de l'utilisateur sont exprimées par le vecteur de poids w1 = {0.4, 0.3, 0.3}.

Dans cette configuration, la minimisation du temps de production est considérée comme la priorité principale (40%), tandis que l'optimisation de l'énergie et la réduction des coûts revêtent une importance égale (30% chacun).

3.8 IHM développée

Afin de faciliter l'utilisation et le contrôle de la simulation, nous avons conçu une Interface Homme-Machine (IHM) dédiée. En effet, CloudSim, la plateforme utilisée pour la simulation des environnements cloud, ne propose qu'une interface en mode console, ce qui peut s'avérer complexe et peu intuitive pour certains utilisateurs.

L'IHM que nous avons développée permet donc d'interagir avec la simulation de manière plus accessible et visuellement structurée. Elle guide l'utilisateur à travers les différentes étapes nécessaires à la configuration du scénario de simulation.

Interface principale : L'interface principale constitue l'écran d'accueil de l'application. Elle s'affiche dès l'ouverture du logiciel et présente de manière synthétique l'ensemble des informations relatives à notre projet de fin d'études. La Figure III.2 illustre cette interface et met en évidence les éléments clés mis à disposition de l'utilisateur.

Figure III. 2: Interface principale.

Choix de configuration L'IHM propose deux modes de configuration distincts : la configuration par défaut et la configuration personnalisée

Ces deux options sont illustrées dans la Figure III.3 et visent à s'adapter à différents profils d'utilisateurs et objectifs de simulation.

- Configuration par défaut : Le mode par défaut est destiné aux utilisateurs souhaitant effectuer rapidement une simulation sans avoir à modifier manuellement les paramètres techniques. Lorsque cette option est sélectionnée, l'application redirige directement vers l'interface de simulation. À ce niveau, l'utilisateur peut spécifier le nombre de cloudlets et de machines virtuelles (VM), ce qui permet d'initialiser une simulation basique mais fonctionnelle.

- Configuration personnalisée : Le mode personnalisé offre quant à lui un accès détaillé aux paramètres avancés de la simulation. Il permet à l'utilisateur de modifier les caractéristiques des cloudlets (comme la taille, l'instruction count, la puissance requise), celles des machines virtuelles (CPU, RAM, bande passante, etc.), ainsi que les propriétés des machines physiques (hôtes) telles que la capacité de calcul et les ressources allouées.

Ce mode est particulièrement utile pour les utilisateurs souhaitant tester des scénarios spécifiques, comparer des politiques d'ordonnancement ou évaluer les performances de l'infrastructure cloud selon des critères précis.

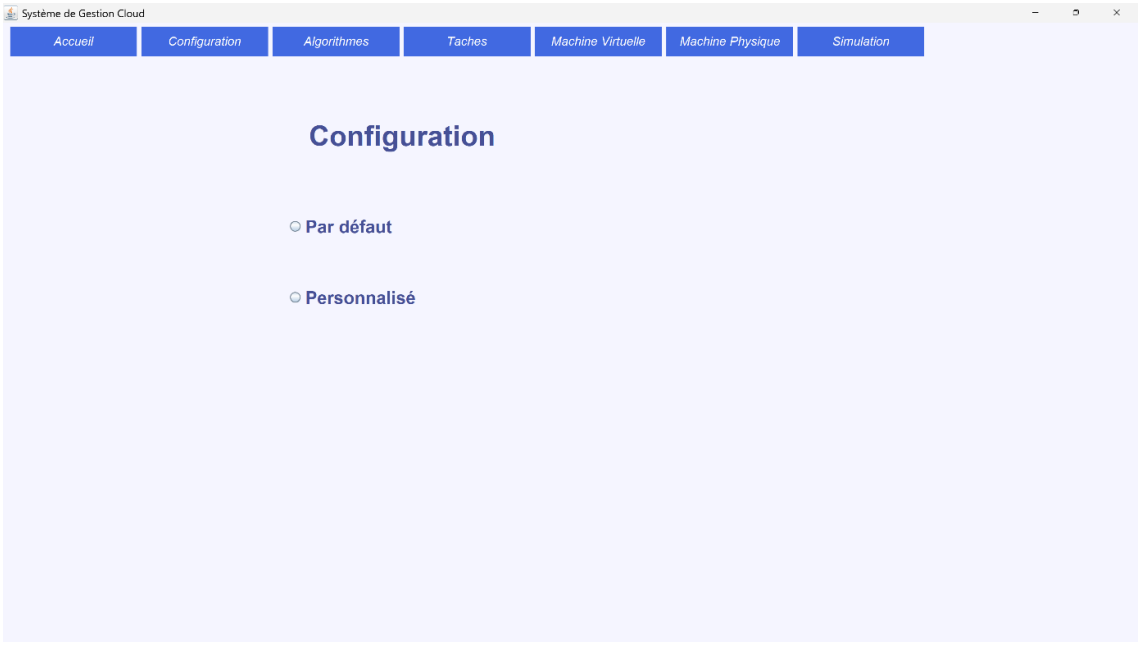

Figure III. 3: choix de configuration.

Choix de l'algorithme d'ordonnancement : L'interface illustrée dans la FigureIII.4 apparaît une fois que l'utilisateur a opté pour le mode de configuration par défaut elle permet à l'utilisateur de sélectionner l'algorithme d'ordonnancement à utiliser lors de la simulation, parmi les options suivantes : Bat Algorithm (BA) , Particle Swarm Optimization (PSO) , ou encore une comparaison globale de tous les algorithmes .

Ces méthodes sont intégrées au sein de notre approche basée sur l'algorithme Shark Smell Optimization (SSO) , utilisé comme méthode principale pour l'optimisation de l'ordonnancement des tâches cloud. Cette fonctionnalité permet ainsi de comparer les performances du SSO avec celles des autres algorithmes connus dans la littérature, dans des conditions similaires de simulation.

Par défaut, l'option « Tous » est sélectionnée afin de faciliter une analyse comparative globale, offrant à l'utilisateur une vue synthétique des résultats obtenus par chaque algorithme sous les mêmes paramètres de configuration. Pour permettre une comparaison globale des performances.

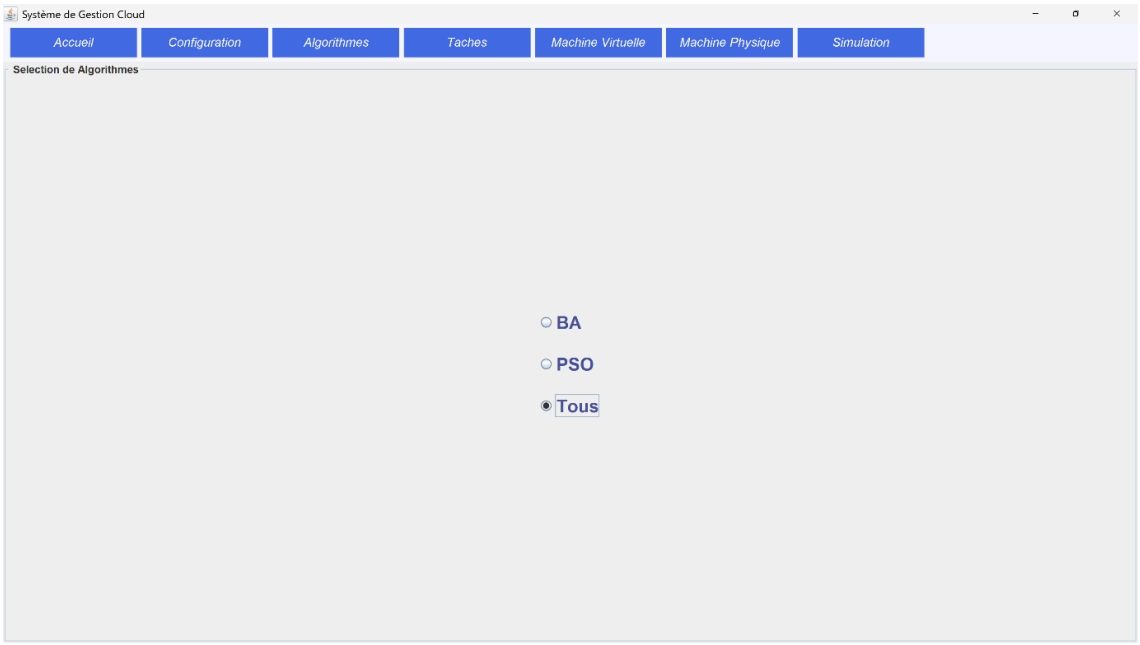

Figure III. 4: Choix de l'algorithme d'ordonnancement.

Lancement des résultats de simulation : L'interface illustrée dans la Figure III.5 permet de définir à la fois le nombre de tâches (cloudlets) ainsi que le nombre de machines virtuelles (VMs) à déployer lors de la simulation surtout en mode de configuration par defaut. Après avoir ajusté ces paramètres, l'utilisateur peut cliquer sur le bouton « Démarrer Simulation » pour initier l'exécution et accéder ensuite aux résultats obtenus.

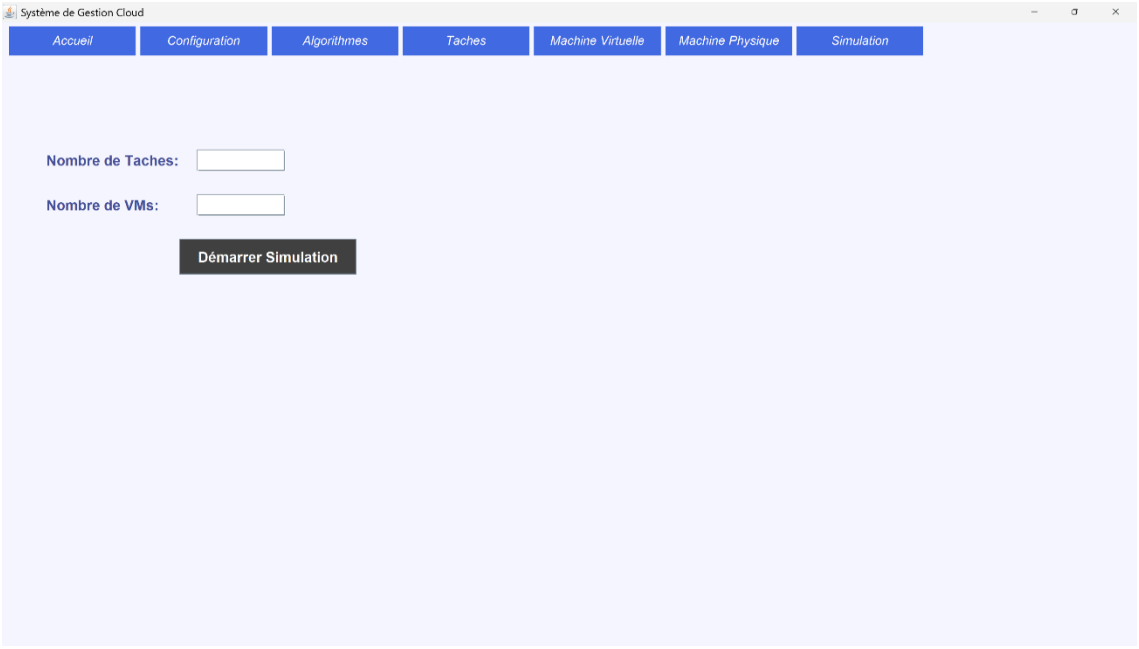

Figure III. 5: Lancement des résultats de simulation.

Configuration des Cloudlets : dans cette section qui illustrée à la Figure III.6 l'utilisateur a la possibilité de définir le nombre total de cloudlets à inclure dans la simulation. De plus, il propose des sections pour indiquer la répartition en pourcentage de ces cloudlets dans les catégories de tailles suivantes : minuscule, petite, moyenne, grande et très grande. Cette configuration permet de créer un environnement de simulation plus réaliste et flexible.

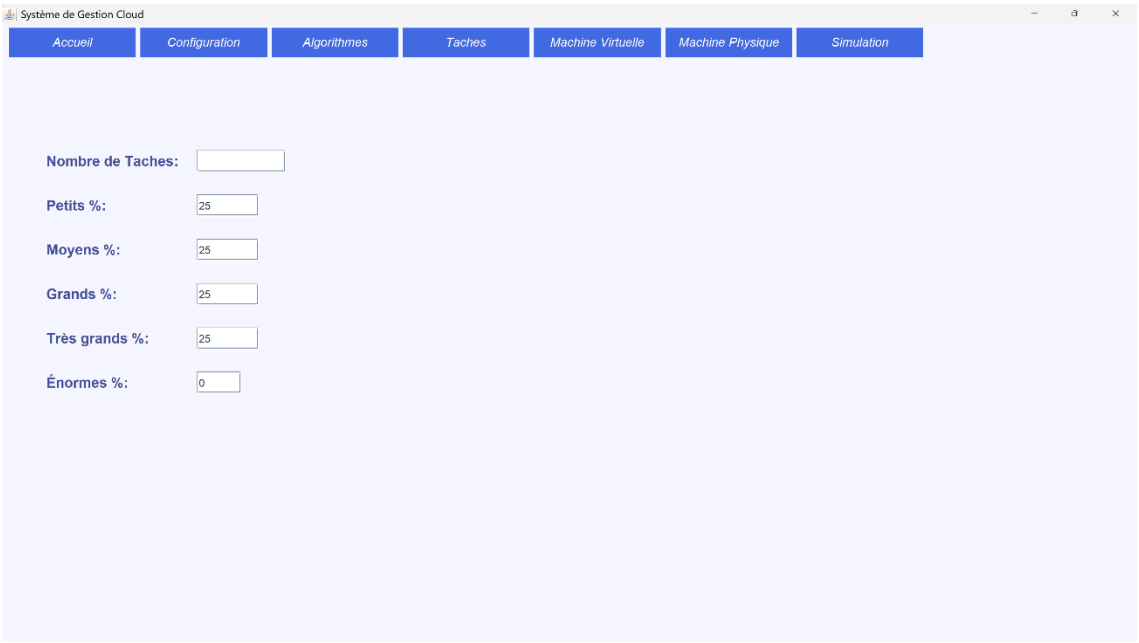

Figure III. 6: Configuration des Cloudlets.

Configuration des machines virtuelles : une fois que nous avons terminé la configuration des tâches, la prochaine étape consiste à configurer les machines virtuelles (VMs)

Comme démontré dans la figure III.7, chaque machine virtuelle a une vitesse de traitement (MIPS) maximum et minimum, une capacité de mémoire vive, une bande passante et une capacité de stockage spécifiques.

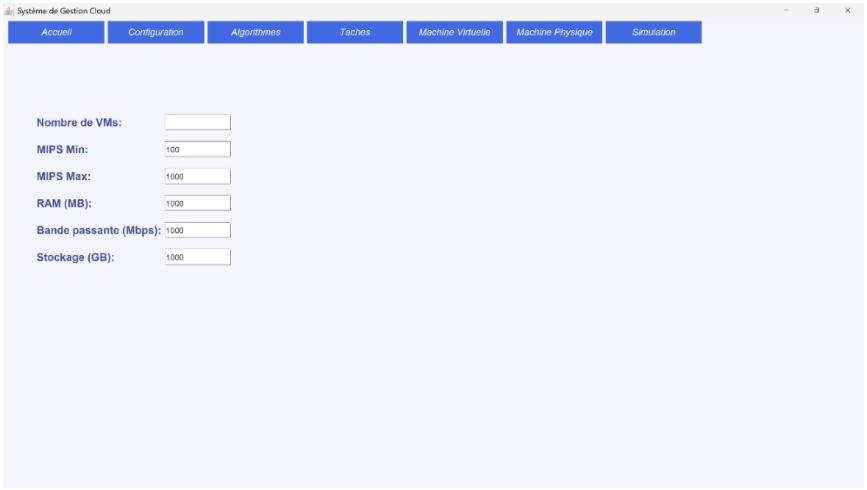

Figure III. 7: configuration des machines virtuelles.

Configuration des machines physiques : l'insertion des données essentielles concernant les machines physiques est la première étape pour configurer l'environnement cloud. Comme illustré dans la Figure III. 8, cela implique de déterminer le nombre de centres d'hôtes, la bande passante, le stockage d'hôte par mégaoctet, la mémoire d'hôte.

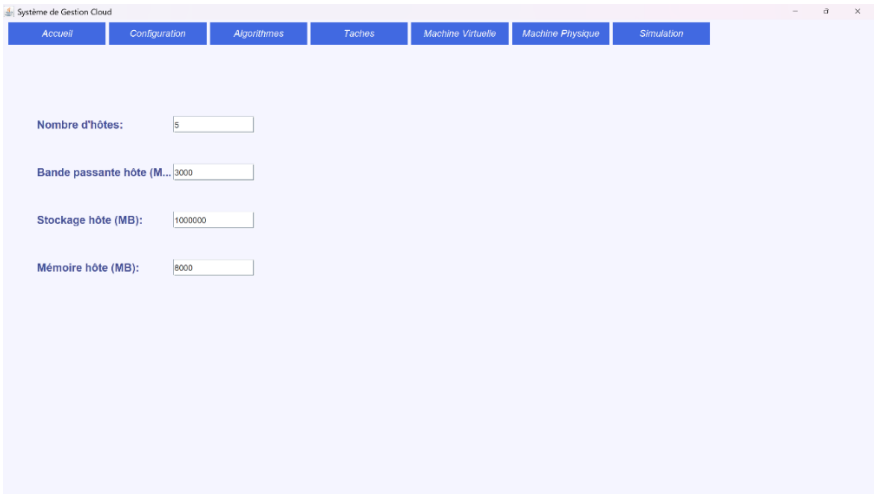

Figure III. 8: configuration des machines physiques.

Simulation : Cette interface affiche dans le cas que l'utilisateur voulais personnaliser la configuration, il peut maintenant spécifier le nombre des taches et des machines virtuelles. Lancement de la simulation se fait en cliquant sur le bouton «Demarrer Simulation » dans l'interface montrée à la Figure III.9 Au cours de la simulation, une approche de planification multi-objectifs a été utilisée, se basant sur l'optimisation du temps d'exécution, des coûts et de la consommation énergétique.

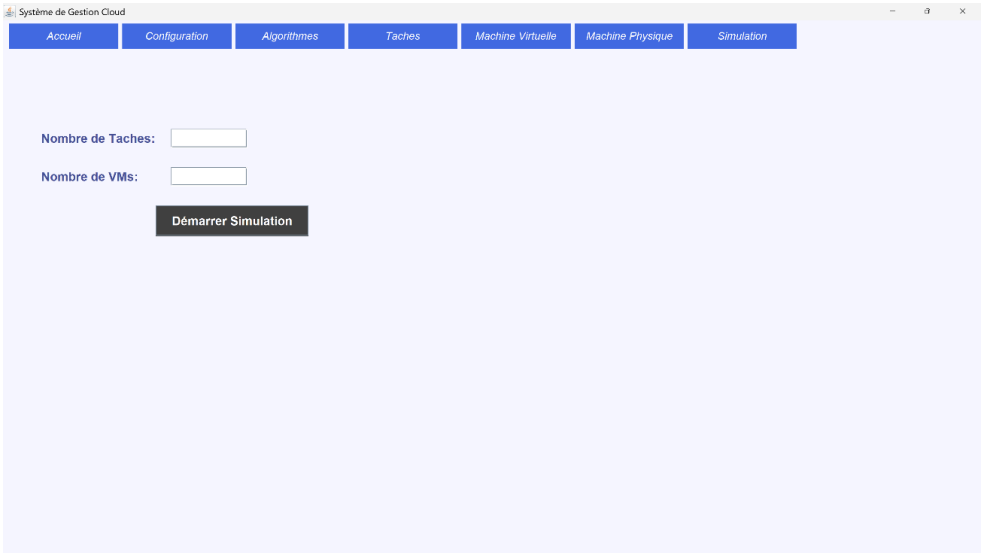

Figure III. 9: Lancement de simulation.

Après l'accomplissement des étapes de mise en place et de pré-simulation, la distribution des tâches entre les différentes machines virtuelles est envoyée au courtier afin de lancer la simulation et afficher les résultats.

3.9 Résultats obtenus et étude comparative

Dans cette partie, nous allons analyser la configuration des simulations et les résultats obtenus suite à l'application de l'algorithme SSO pour le planifiation des tâches.Les simulations ont été effectuées dans un environnement hétérogène, comme précisé dans les tableaux III.1et III.2, qui affichent les valeurs attribuées à chaque paramètre.

Tableau III. 1: les paramètres de simulation CloudSim .

Objet	Détails	Valeurs
Machines virtuelles	Bonde passante	1000 Mb
	Nombre de processeurs	1
	Nombre de machines virtuelles	25
	RAM	1000
	Vitesse d'exécution (MIPS)	100-1000
	System d'exploitation	Linux
	Type de politique	Time shared
	Stockage	10000 MB
	VMM	Xen
Machines physiques	Bonde passante	3000 MB
	Stockage Nombre d'hôtes	1000000 MB
	Nombre d'hôtes	5
	RAM	8000 MB
	politique	Time shared
	processeurs	4 Dual core (4000 mips) 26 quad core (4000 mips)
Centres de données	Nombre	2

Pour les Cloudlets, nous avons choisis pour une variation de la longueur conformément au tableau III.2

Tableau III. 2: les paramètres de longueur des Cloudlets.

Type de Cloudlet	Distribution
Petite	35%
Moyenne	40%
Grande	5%
Extra large	15%
Enorme	*5%*

3.10 Comparaison de résultats en termes de makespan :

La figure III.10 illustre la comparaison de makespan entre SSO et les diverses méthodes examinées pour 25 Vms.

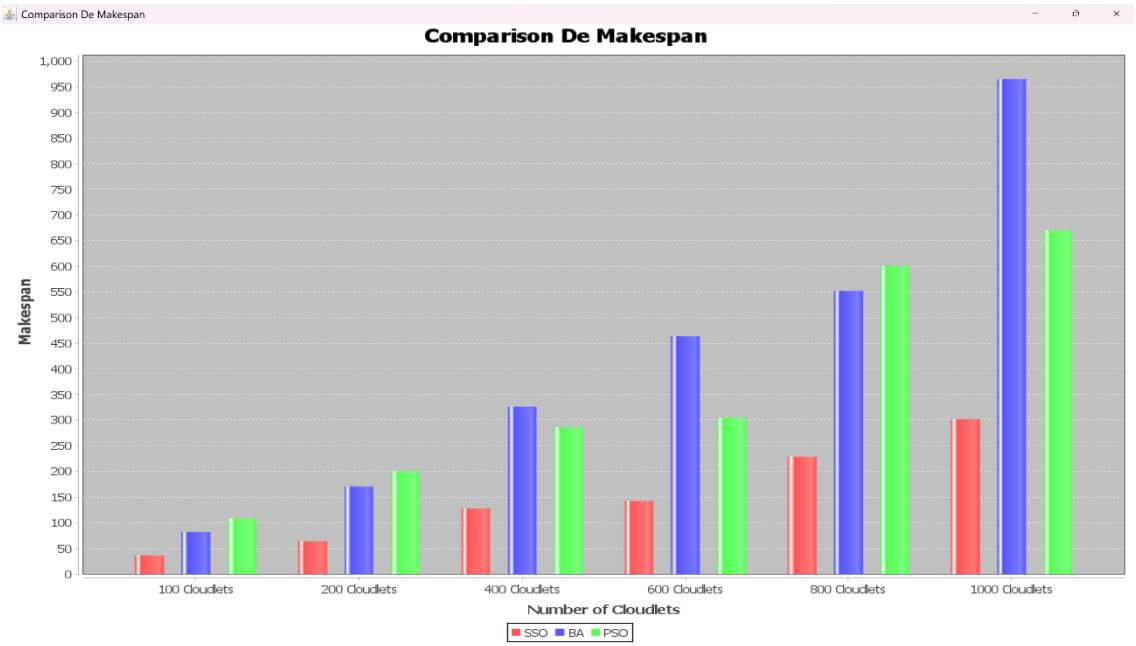

Figure III. 10: comparaison en termes de makespan pour 25 Vms.

Le tableau 3.4 montre le détail des valeurs.

Tableau III. 3: résultats de comparaison en termes de makespan pour 25 Vms.

Nombre de cloudlets	100	200	400	600	800	1000
SSO	**36.631**	**64.084**	**127.721**	**142.661**	**228.64**	**301.435**
BA	81.547	170.652	326.174	463.797	551.786	964.516
PSO	108.018	200.358	286.34	303.774	601.717	669.528

3.10.1 Résultat :

L'analyse comparative des techniques d'ordonnancement met en évidence la performance remarquable de la technique SSO en matière de réduction du makespan. Selon la figure III.10 et le tableau III.3 , cette technique parvient à maintenir un temps d'exécution globalement faible, quelle que soit la charge de travail. Elle se distingue par une croissance maîtrisée du makespan à mesure que le nombre de cloudlets augmente, traduisant ainsi une meilleure capacité d'adaptation. À l'inverse, la technique BA affiche une dégradation plus importante des performances, tandis que la technique PSO se positionne dans une zone intermédiaire, avec des résultats acceptables mais globalement inférieurs à ceux de SSO.

3.11 Comparaison de résultats en termes de coût :

La figure III.11 illustre la comparaison de coût entre SSO et les diverses méthodes examinées pour 25 Vms.

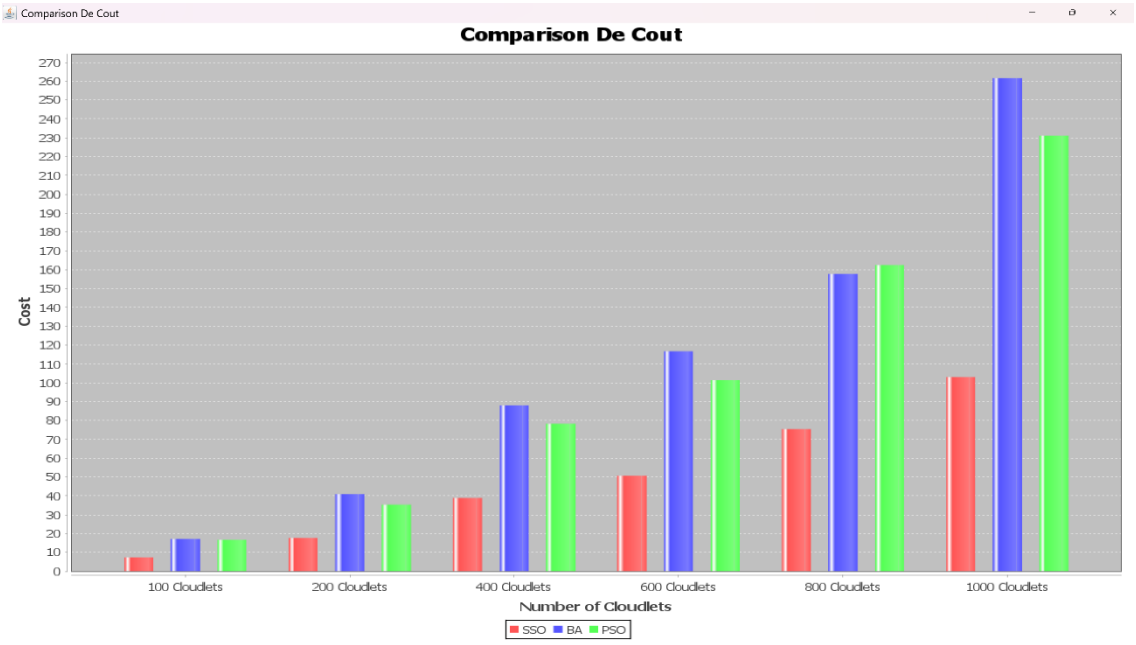

Figure III. 11: comparaison en termes de coût pour 25 Vms.

Le tableau III.4 montre le détail des valeurs.

Tableau III. 4: résultats de comparaison en termes de coût pour 25 Vms.

Nombre de cloudlets	100	200	400	600	800	1000
SSO	**7.23**	**17.695**	**38.852**	**50.617**	**75.482**	**103.085**
BA	17.136	40.844	87.879	116.736	157.813	261.659
PSO	16.685	35.416	78.272	101.456	162.382	231.25

3.11.1 Résultat :

En ce qui concerne l'optimisation des coûts, les résultats présentés dans la figure III.11 et le tableau III.4 mettent en évidence la supériorité de la technique SSO. Grâce à une gestion plus efficiente des ressources, cette technique permet d'exécuter les tâches à un coût réduit. En comparaison, la technique PSO engendre un coût plus élevé que SSO, mais demeure globalement plus économique que la technique BA. Cela traduit une utilisation des ressources plus optimisée que BA, mais moins performante que SSO. L'efficacité de SSO à cet égard se révèle particulièrement pertinente dans un environnement cloud, où la facturation dépend directement de l'utilisation des ressources.

3.12 Comparaison de résultats en termes de d'énergie :

La figure III.12 illustre la comparaison d'énergie entre SSO et les diverses méthodes examinées pour 25 Vms.

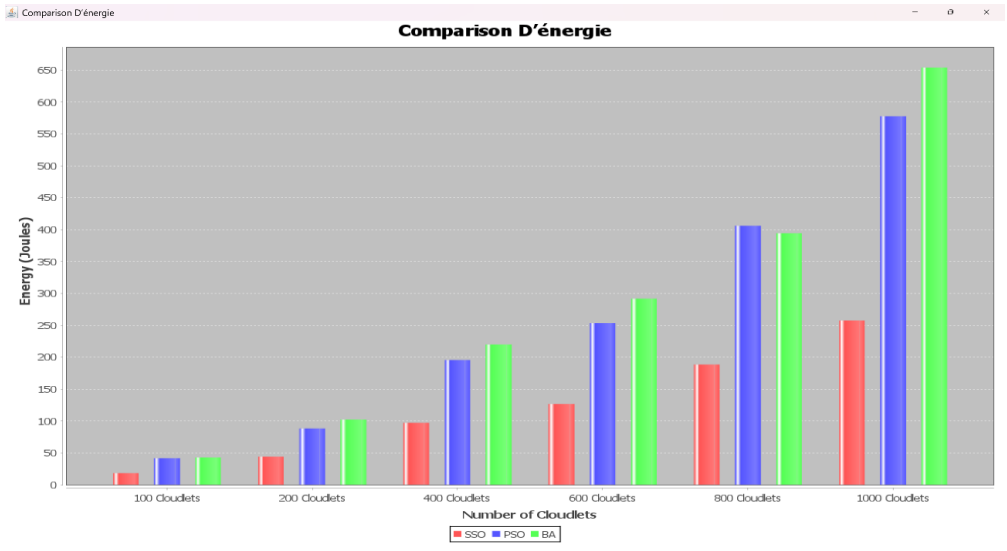

Figure III. 12: comparaison en termes d'énergie pour 25 Vms.

Le tableau III.5 montre le détail des valeurs en petajoule.

Tableau III. 5: résultats de comparaison en termes d'énergie pour 25 Vms.

Nombre de cloudlets	100	200	400	600	800	1000
SSO	**18.075**	**44.238**	**97.129**	**126.542**	**188.705**	**257.713**
BA	41.712	88.54	195.679	253.641	405.956	578.126
PSO	42.84	102.109	219.697	291.84	394.533	654.148

3.12.1 Résultat :

Sur le plan de la consommation énergétique, les résultats présentés dans la figure III.12 et le tableau III.5 confirment que la technique SSO est la plus efficiente. Elle maintient une consommation stable et maîtrisée, même en présence d'une charge élevée, ce qui reflète une allocation optimale des ressources. La technique PSO, bien qu'elle présente une consommation plus élevée que SSO, demeure globalement plus performante que BA. Cette dernière enregistre la consommation la plus importante, ce qui limite son adéquation avec des environnements sensibles aux enjeux de durabilité énergétique. Ainsi, SSO se distingue comme l'approche la plus appropriée dans des contextes où la réduction de l'empreinte énergétique constitue une priorité.

3.12.2 Synthèse des résultats :

L'analyse globale des résultats présentés dans les figures III.10 à III.12 et les tableaux III.3 à III.5 met en évidence la supériorité de la technique d'optimisation par odeur de requin (SSO) en matière d'ordonnancement de tâches dans un environnement cloud. Sur les trois critères d'évaluation : makespan, coût d'exécution et consommation énergétique SSO se distingue par une efficacité accrue et une robustesse face à l'augmentation de la charge de travail. Cette technique permet une allocation optimale des ressources, réduisant à la fois les temps de traitement, les coûts opérationnels et l'empreinte énergétique. Comparativement aux techniques PSO et BA, elle offre un compromis performant entre efficacité computationnelle et efficience énergétique. Ces résultats positionnent la technique SSO comme une approche particulièrement adaptée aux systèmes distribués, en répondant aux exigences croissantes d'optimisation, de scalabilité et de durabilité dans le domaine du cloud computing.

4 Conclusion

Dans le dernier volet de ce mémoire, nous avons exposé la mise en œuvre technique de l'algorithme Shark Smell Optimization (SSO) pour l'ordonnancement des tâches dans le cloud. Cette phase a consisté à intégrer la technique proposée dans l'environnement de simulation CloudSim, tout en développant une interface graphique facilitant la configuration et le suivi des expérimentations.

Les résultats obtenus à travers les simulations ont permis de comparer la technique SSO à d'autres approches métaheuristiques telles que PSO et BA. L'analyse a mis en évidence la supériorité de SSO sur les critères de performance étudiés, en particulier le makespan, le coût d'exécution et la consommation énergétique. Ces performances traduisent la capacité de la technique à exploiter efficacement les ressources disponibles dans un environnement cloud dynamique.

Ainsi, cette étude expérimentale vient consolider la pertinence de la technique proposée et met en lumière son potentiel pour des environnements réels nécessitant une optimisation fine des ressources.

Conclusion Générale

Conclusion Générale

Ce mémoire a porté sur l'étude et la mise en œuvre d'une solution d'ordonnancement de tâches dans le cloud computing, fondée sur l'algorithme métaheuristique Shark Smell Optimization (SSO). Face à la croissance constante des besoins en ressources dans les environnements cloud, l'optimisation de l'allocation des tâches constitue un enjeu majeur tant en termes de performance que de consommation de ressources.

Dans une première phase, nous avons posé le cadre théorique de notre travail en explorant les concepts clés liés au cloud computing, aux techniques d'ordonnancement, ainsi qu'aux méthodes de résolution, qu'elles soient exactes ou approchées. Nous avons ensuite présenté en détail le fonctionnement de l'algorithme SSO et sa pertinence dans le contexte de l'ordonnancement.

La partie expérimentale a permis de mettre en œuvre cette technique dans l'environnement de simulation CloudSim, avec le développement d'une interface graphique facilitant la configuration et l'analyse des résultats. Des tests comparatifs ont été menés entre SSO et deux techniques de référence (PSO et BA), sur des critères tels que le makespan, le coût d'exécution et la consommation énergétique. Les résultats obtenus ont mis en évidence la supériorité de SSO en termes d'efficacité, de stabilité et d'optimisation des ressources.

Ainsi, ce travail démontre la capacité de l'algorithme SSO à répondre efficacement aux défis de l'ordonnancement dans le cloud. Il constitue une base solide pour le développement de solutions plus intelligentes et adaptatives dans des environnements distribués.

Toutefois, certaines limites demeurent, notamment l'absence de prise en compte de contraintes spécifiques liées aux utilisateurs ou aux types de tâches. Des perspectives intéressantes pourraient consister à enrichir le modèle avec des objectifs multi-critères, à tester la robustesse de SSO sur des plateformes réelles ou à envisager des approches hybrides combinant plusieurs techniques d'optimisation.

En définitive, cette étude met en lumière l'importance d'exploiter les techniques métaheuristiques modernes pour concevoir des solutions performantes, évolutives et économes dans le domaine du cloud computing.

Références

[1] A. A. e. al., Above the Clouds: A Berkeley View of Cloud Computing, EECS Department, University of California, Berkeley ech. Rep, 2009.

[2] S. S. a. M. D. S. S. K. Singh, Survey of task scheduling algorithms in cloud computing, International Journal of Computer Applications : vol. 49, no. 15, pp. 7-12, 2012.

[3] Red Hat, «Un cloud public, qu'est-ce que c'est ?,» [En ligne]. Available: https://www.redhat.com/fr/topics/cloud-computing/what-is-public-cloud. [Accès le 12 01 2025].

[4] Antares, «Les principales caractéristiques et capacités du cloud privé,» 2024. [En ligne]. Available: https://www.antares.fr/cloud-prive/les-principales-caracteristiques-et-capacites/. [Accès le 13 01 2025].

[5] «wikipedia,» [En ligne]. Available: https://en.wikipedia.org/wiki/Community_cloud. [Accès le 14 01 2025].

[6] [En ligne]. Available: https://www.editions-eni.fr/livre/cloud-prive-hybride-et-public-quel-modele-pour-quelle-utilisation-un-etat-de-l-art-et-des-bonnes-pratiques-2e-edition-9782409042676. [Accès le 20 01 2025].

[7] GHOMARI, Mohammed El-Amine, «Ordonnancement_des_taches_dans_le_Cloud_computing_avec_SFLA__synthese_de_quelques_methodes_d_aide_multicriteres_a_la_decision.pdf,» 25-jui-2023. [En ligne]. Available: http://dspace.univ-tlemcen.dz/handle/112/22488.

[8] «amazon,» [En ligne]. Available: https://aws.amazon.com/what-is/iaas/. [Accès le 25 01 2025].

[9] «oracle,» [En ligne]. Available: https://www.oracle.com/fr/cloud/definition-saas/. [Accès le 30 01 2025].

[10] «ibm,» [En ligne]. Available: https://www.ibm.com/think/topics/iaas-paas-saas. [Accès le 02 02 2025].

[11] [En ligne]. Available: https://elearning-facsci.univ-annaba.dz/pluginfile.php/9177/mod_resource/content/1/Chapitre%203%20-%20M%C3%A9thodes%20de%20r%C3%A9solution.pdf. [Accès le 10 02 2024].

[12] U. A. B. B. –. T. F. d. S. D. d. 2. Application des méthodes exactes pour l'optimisation.

[13] [En ligne]. Available: http://www.fsr.ac.ma/DOC/cours/maths/Souad%20Bernoussi/Cours%20C2SI.pdf. [Accès le 15 02 2025].

[14] «technoscience,» 27 02 2025. [En ligne]. Available: https://www.techno-science.net/glossaire-definition/Heuristique.html.

[15] «technoscience,» [En ligne]. Available: https://www.techno-science.net/glossaire-definition/Metaheuristique.html. [Accès le 27 02 2025].

[16] C. d.-l. (. C. 3. -. P. 1.-2. [. P. Université de Mila. [En ligne]. Available: https://view.officeapps.live.com/op/view.aspx?src=https%3A%2F%2Felearning.centre-univ-mila.dz%2Fa2024%2Fpluginfile.php%2F85407%2Fmod_resource%2Fcontent%2F1%2FChapitre%25203%2520-%2520parties%252012.pptx&wdOrigin=BROWSELI.

[17] E. –. H. e. métaheuristiques. [En ligne]. Available: https://www.eurodecision.com/algorithmes/recherche-operationnelle-optimisation/heuristiques-meta-heuristiques. [Accès le 01 03 2025].

[18] O. Z.-A. B. &. C. X. (. Bozorg-Haddad, « Advanced Optimization by Nature-Inspired Algorithms. Springer, Studies in Computational Intelligence,» 2018 ;vol. 720. DOI: 10.1007/978-981-10-5221-7. [En ligne]. Available: http://ndl.ethernet.edu.et/bitstream/123456789/66756/1/1.pdf.

[19] Mahboubi, S., & Themer, H., «Une approche intelligente pour un problème d'ordonnancement avec PSO [Mémoire de master, Université 8 Mai 1945 - Guelma]. DSpace Université de Guelma,» 2019. [En ligne]. Available: https://dspace.univ-guelma.dz/jspui/handle/123456789/4280.

[20] Yang, X.-S., & Slowik, A, «The Bat Algorithm: An Introduction. In A. Slowik (Ed.), Nature-Inspired Optimizers (pp. 1–22). Springer,» 2020. [En ligne]. Available: https://www.researchgate.net/publication/340540442_The_Bat_Algorithm_An_Introduction.

[21] J. Farrell, Java™ Programming. Cengage Learning, Inc, 2019.

[22] O. (n.d.), «Java Platform, Standard Edition Documentation,» Oracle Corporation. Retrieved June 5, 2025. [En ligne]. Available: https://docs.oracle.com/en/java/.

[23] O. C. (n.d.), «OpenJDK: The Open Java Development Kit. Retrieved June 5, 2025,» [En ligne]. Available: https://openjdk.org/.

[24] J. T. (n.d.), «JFreeChart - A free Java chart library. Retrieved June 5, 2025,» [En ligne]. Available: https://www.jfree.org/jfreechart/.

[25] S. C. (n.d.), «CloudSim Architecture. Retrieved June 5, 2025,» [En ligne]. Available: https://ce.snscourseware.org/files/166918s3152.pdf.

[26] Eddine, K. , «Cloud Computing: CloudSim. SlideShare. Retrieved,» (2019). [En ligne]. Available: https://www.slideshare.net/slideshow/cloud-computing-cloud-sim/191837387.

[27] C. Plus, « Documentation de l'API Java – Version 5.2.3. Consulté le 6 juin 2025,» 2023. [En ligne]. Available: https://javadoc.io/doc/org.cloudsimplus/cloudsim-plus/5.2.3/allclasses-index.html.

[28] Geeta Singh, Shiva Prakash, Santosh Kumar, «Minimizing Makespan Time in Cloud Computing using Heuristic Elasticity based Dynamic Task Scheduling Algorithms, Journal of System and Management Sciences,» Vol. 11 (2021) No. 2, pp. 29-47, DOI:10.33168/JSMS.2021.0203. [En ligne]. Available: https://www.aasmr.org/jsms/Vol11/vol.11.2.3.pdf.

[29] Chaisiri, S., Lee, B.-S., & Niyato, D., «Cost in Cloud Computing. IEEE Transactions on Services Computing, 5(2), 164–177, https://doi.org/10.1109/TSC.2011.7,» 2012 . [En ligne]. Available: http://www.intercloudtestbed.org/uploads/2/1/3/9/21396364/optimization_of_resource_provisioning_cost_in_cloud_computing.pdf.

[30] P. Ghomari, Ordonnancement des tâches dans le Cloud computing avec SFLA : synthèse de quelques méthodes d'aide multicritères à la décision, Mémoire de Master, Univ. Tlemcen, Algérie, 2016..

Abstract

Cloud computing has emerged as a fundamental paradigm for delivering scalable, flexible and on-demand IT services over the internet. Among the critical challenges it presents are effective task planning, which significantly affects overall system performance, operational costs and energy efficiency. This problem is inherently a multi-objective optimization task. In this study, we study the use of the shark smell optimization (SSO) algorithm to solve this problem by targeting the minimization of three essential metrics: time required, running cost and energy consumption. The approach was implemented using CloudSim simulation framework and compared to two widely adopted metaheuristics: particle swarm optimization (PSO) and Bat algorithm (BA). The results indicate that the SSO-based strategy offers superior performance and better resource management, confirming its potential for efficient task scheduling in cloud environments.

Keywords: Cloud computing, task planning, multi-objective optimization, CloudSim, Shark Smell optimization, PSO, BA.

Résumé

L'informatique en nuage s'est imposée comme un paradigme fondamental pour fournir des services informatiques évolutifs, flexibles et à la demande sur Internet. Parmi les défis critiques qu'il présente, citons la planification efficace des tâches, qui affecte considérablement la performance globale du système, les coûts opérationnels et l'efficacité énergétique. Ce problème est intrinsèquement une tâche d'optimisation multi-objectif. Dans cette étude, nous étudions l'utilisation de l'algorithme d'optimisation de l'odeur de requin (SSO) pour résoudre ce problème en ciblant la minimisation de trois métriques essentielles : le temps nécessaire, le coût d'exécution et la consommation d'énergie. L'approche a été mise en œuvre à l'aide du cadre de simulation CloudSim et comparée à deux métaheuristiques largement adoptées : l'optimisation par essaim de particules (PSO) et l'algorithme Bat (BA). Les résultats indiquent que la stratégie basée sur le SSO offre des performances supérieures et une meilleure gestion des ressources, confirmant son potentiel pour une planification efficace des tâches dans les environnements cloud.

Mots-clés : Cloud computing, planification des tâches, optimisation multi-objectif, CloudSim, optimisation Shark Smell, PSO, BA.

الملخص

اصبحت الحوسبة السحابية نموذجًا أساسيًا لتوفير خدمات حوسبة مرنة، قابلة للتوسع، وتُقدَّم عند الطلب عبر الإنترنت. من أبرز التحديات في هذا المجال جدولة المهام بكفاءة، لما لها من تأثير مباشر على أداء النظام العام، وتكلفة التشغيل، وكفاءة استهلاك الطاقة. وتُعد هذه المشكلة من مسائل التحسين متعددة الأهداف. في هذه الدراسة، قمنا باعتماد خوارزمية تحسين رائحة القرش (SSO) لمعالجة هذه المشكلة، وذلك من خلال استهداف تحسين ثلاث مؤشرات رئيسية: مدة التنفيذ (makespan)، تكلفة التنفيذ، واستهلاك الطاقة. تم تنفيذ المقترح باستخدام بيئة المحاكاة CloudSim، ومقارنته بخوارزميتين شائعتين هما: خوارزمية سرب الجسيمات (PSO) وخوارزمية الخفاش (BA). وقد أظهرت النتائج أن خوارزمية SSO توفر أداءً متفوقًا وإدارة أفضل للموارد، مما يؤكد فعاليتها في جدولة المهام في بيئات الحوسبة السحابية.

الكلمات المفتاحية: الحوسبة السحابية، جدولة المهام، تحسين متعدد الأهداف، CloudSim، خوارزمية رائحة القرش، PSO، BA