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

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option : Réseaux et Systèmes Distribués (RSD)

Thème

Squirrel Search Algorithm pour l'ordonnancement des tâches dans le Cloud computing

Réaliser par :

- Oussama BELKHODJA
- Ali Amine DIF

Présenté le 24 juin 2025 devant le jury composé de :

- Mr Youcef BENMOUNA — Président
- Mr Ahmed Khalid Yassine SETTOUTI — Examinateur
- Mr Badr BENMAMMAR — Encadrant
- Mr Arslan Nedhir MALTI — Co-Encadrant

Année universitaire : 2024-2025

Remerciement

Nous remercions tout d’abord Allah, Le Tout-Puissant, de nous avoir donné la force, la patience et la volonté pour mener à bien ce travail.

Nous tenons à exprimer notre sincère gratitude à **M. BENMAMMAR Badr**, notre encadrant, pour son accompagnement constant, sa rigueur scientifique, ses conseils avisés et son soutien tout au long de ce projet, ainsi qu'à **M. MALTI Arslan Nedhir**, notre co-encadrant, pour ses orientations précieuses et son appui technique durant l’implémentation.

Je tiens à exprimer ma profonde gratitude aux membres du jury, **M. BENMOUNA Youcef** et **M. SETTOUTI Ahmed Khalid Yassine**, pour l’honneur qu’ils nous ont fait en acceptant d’évaluer ce travail et pour le temps précieux qu’ils y ont consacré.

Nous exprimons également notre reconnaissance à l’ensemble des enseignants du département d’informatique de l’Université Abou Bakr Belkaid Tlemcen, pour leur enseignement de qualité tout au long de notre parcours universitaire.

Un grand merci à nos familles, pour leur amour, leur patience et leurs encouragements sans faille.

Enfin, nos remerciements vont à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire.

Dédicace

Je dédie ce travail :

- À mes chers parents, pour leur soutien indéfectible et leur amour continu.
- À mes amis et collègues, spécialement **BELARBI Nouh Lokman**, TIZAOUI **Fatima Zohra**, **ALIANE Alaa** et **CHAREF Khadidja Fatna**, pour leurs conseils et leur aide constants.
- À mes sœurs et frères.
- À tous ceux qui me sont chers.

DIF ALI AMINE & BELKHODJA OUSSAMA

Table des matières

Remerciement i
Dédicace ii
Table des matières iii
Liste des figures v
Liste des tables vi
Liste des abréviations vii
Résumé viii
Introduction générale 1
Chapitre 1 : Cloud Computing 2
1.1 Introduction 3
1.2 Définition du Cloud computing 3
1.3 Caractéristiques du Cloud computing 4
1.4 Types de déploiement Cloud computing 4
1.4.1 Cloud public 4
1.4.2 Cloud privé 5
1.4.3 Cloud hybride 6
1.5 Services du Cloud computing 6
1.5.1 Software as a service 7
1.5.2 Platform as Services 7
1.5.3 Infrastructure as a services 8
1.6 Évolution du Cloud computing 8
1.7 Sécurité du Cloud computing 9
1.8 Avantages et inconvénients du Cloud computing 9
1.8.1 Avantages du Cloud computing 9
1.8.2 Inconvénients du Cloud computing 9
1.9 Rôle de l'optimisation dans le Cloud computing 10
1.10 Conclusion 10

Chapitre 2 : Les métaheuristiques 11
2.1 Introduction 12
2.2 Les méthodes de résolution 12
2.2.1 Les méthodes exactes 12
2.2.2 Les méthodes approchées 12
2.3 Caractéristiques des métaheuristiques 14
2.4 Quelques algorithme inspirés de phénomènes naturels 14
2.4.1 algorithmes génétiques 14
2.4.2 Optimisation par essaims de particules 14
2.4.3 Algorithme de recherche d'écureuil 15
2.5 Conclusion 23
Chapitre 3 : Implémentation de l'application et évaluation des résultats obtenus 24
3.1 Introduction 25
3.2 Environnement de développement et de simulation 25
3.2.1 Java 25
3.2.2 Netbeans 25
3.2.3 JFreeChart 25
3.2.4 CloudSim 26
3.3 Critères d'évaluation 27
3.3.1 Makespan 27
3.3.2 Coût 28
3.3.3 Énergie 28
3.4 La méthode de la somme pondérée 29
3.5 IHM développée 30
3.6 Résultats obtenus et étude comparative 37
3.6.1 Comparaison de résultats en termes de makespan 39
3.6.2 Comparaison de résultats en termes de coût 40
3.6.3 Comparaison de résultats en termes d'énergie 41
3.7 Conclusion 42
Conclusion générale 43
Bibliographie 44

Liste des figures

Figure 1.1 : Le concept du Cloud computing [1]. 3
Figure 1.2 : Modèle d'un Cloud public [2]. 5
Figure 1.3 : Modèle d'un Cloud privé [3]. 5
Figure 1.4 : Modèle d'un Cloud hybride [4]. 6
Figure 1.5 : Modèles de service Cloud [5]. 6
Figure 1.6 : Software as a service (SaaS) [6]. 7
Figure 1.7 : Platform as Services (PaaS) [7]. 8
Figure 1.8 : Infrastructure as a services (IaaS) [8]. 8

Figure 2.1: Classes des méthodes de résolutions [18]. 13
Figure 2.2: Southern flying squirrel [25]. 15
Figure 2.3: Modèle conceptuel du déplacement d'un écureuil volant d'un arbre à un autre en utilisant le vol plané [31]. 18
Figure 2.4: Un écureuil volant en équilibre [29]. 19
Figure 2.5: Modèle approximatif du comportement de vol [29]. 20
Figure 2.6: Pseudo code de l'algorithme SSA [30]. 23

Figure 3.1 : Structure de CloudSim [36] 27
Figure 3.2 : Interface principale . 31
Figure 3.3: Choix de la configuration. 32
Figure 3.4 : Lancement de la simulation. 33
Figure 3.5 : Configuration des Cloudlets. 34
Figure 3.6 : Configuration des machines virtuelles. 35
Figure 3.7 : Configuration des machines physiques. 36
Figure 3.8 : Lancement de la simulation. 37
Figure 3.9 : Comparaison en termes de makespan pour 25 Vms. 39
Figure 3.10 : Comparaison en termes de coût pour 25 Vms. 40
Figure 3.11 : Comparaison en termes d'énergie pour 25 Vms. 41

Liste des tables

Tableau 3.1 : Les paramètres de simulation CloudSim [41]. 38
Tableau 3.2 : Les paramètres de longueur des Cloudlets[41]. 38
Tableau 3.3 : Les paramètres des processeurs [38]. 39
Tableau 3.4 : Résultats de comparaison en termes de makespan pour 25 Vms. 40
Tableau 3.5 : Résultats de comparaison en termes de coût pour 25 Vms. 40
Tableau 3.6 : Résultats de comparaison en termes d'énergie (PJ) pour 25 Vms.41

Liste des abréviations

Acronyme	Signification
AWS	Amazon Web Service
CMOS	Complementary Metal Oxide Semiconductor
DVFS	Dynamic Voltage and Frequency Scaling
GA	Genetic Algorithm
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IHM	Interface Homme-Machine
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MIPS	Million instructions per second
NSGA-II	Non-dominated Sorting Genetic Algorithm
PaaS	Platform as a Service
PE	Processing Element
PDA	Personal Digital Assistant
PJ	Petajoule
PSO	Particle Swarm Optimization
RAM	Random Access Memory
SA	simulated annealing
SaaS	Software as a Service
SSA	Squirrel Search Algorithm
VM	Virtuel Machine
VMM	Virtual Machine Manager
VPN	Virtual Private Network
WSM	Weighted sum model

Résumé

Le Cloud computing est une avancée technologique qui offre un accès facile et sûr aux informations et services stockés sur des serveurs éloignés, assurant ainsi un service de haute qualité et une disponibilité continue. Cette étude porte sur l'optimisation multi-objectifs dans le Cloud computing en utilisant l'algorithme de recherche des écureuils, avec pour objectif de réduire le makespan, le coût et la consommation d'énergie. La problématique abordée concerne l'ordonnancement des tâches sur les différentes machines virtuelles. Notre méthode a démontré de meilleures performances en comparaison avec les deux méthodes approchées (NSGA-II et PSO). Nous avons utilisé CloudSim comme simulateur et nous avons créé l'interface utilisateur en Java afin de rendre les interactions avec le simulateur plus faciles.

Mots clés : Cloud computing, CloudSim, SSA, NSGA-II, PSO.

Abstract

Cloud computing is a technological advancement that provides easy and secure access to information and services stored on remote servers, thereby ensuring high-quality service and continuous availability. This study focuses on multi-objective optimization in Cloud computing using the Squirrel Search Algorithm, aiming to minimize makespan, cost, and energy consumption. The addressed issue involves task scheduling across various virtual machines. Our approach demonstrated better performance compared to two existing methods (NSGA-II and PSO). We used CloudSim as the simulator and developed the user interface in Java to facilitate interactions with the simulator.

Mots clés : Cloud computing, CloudSim, SSA, NSGA-II, PSO.

ملخص

الحوسبة السحابية هي تقدم تكنولوجي يوفر وصولًا سهلاً وآمنًا إلى المعلومات والخدمات المخزنة على خوادم بعيدة، مما يضمن جودة خدمة عالية وتوفراً مستمراً. تركز هذه الدراسة على تحسين الأهداف المتعددة في بيئة الحوسبة السحابية من خلال استخدام خوارزمية البحث لدى السناجب (SSA) بهدف تقليل زمن التنفيذ، التكاليف، واستهلاك الطاقة. وتتمحور المسألة المدروسة حول جدولة المهام على مختلف الآلات الافتراضية. لقد أظهرت طريقتنا أداءً أفضل مقارنةً بالطريقتين المقاربتين (NSGA-II و PSO) استخدمنا CloudSim كمحاكي، وقمنا بتطوير واجهة المستخدم بلغة Java لتسهيل التفاعل مع المحاكي.

الكلمات الرئيسية : الحوسبة السحاب, CloudSim, SSA, NSGA-II, PSO

Introduction Générale

Le Cloud computing s'impose aujourd'hui comme un pilier central de la transformation numérique, offrant une infrastructure évolutive, flexible et économique pour répondre aux besoins croissants en traitement de données et services informatiques. En permettant un accès à la demande à des ressources virtualisées (stockage, calcul, réseaux), cette technologie a révolutionné la manière dont les entreprises et les institutions consomment et gèrent leurs services informatiques. Cependant, avec la complexité croissante des architectures Cloud, l'optimisation des ressources — notamment l'ordonnancement des tâches — constitue un défi majeur, nécessitant des approches innovantes pour concilier performance, coût et durabilité énergétique.

Dans ce contexte, les métaheuristiques émergent comme des solutions prometteuses pour résoudre des problèmes d'optimisation multi-objectifs complexes. Inspirées de phénomènes naturels ou comportementaux, ces algorithmes explorent efficacement de vastes espaces de solutions tout en évitant les optima locaux. Parmi elles, le Squirrel Search Algorithm (SSA), inspiré de la stratégie de recherche alimentaire des écureuils volants, se distingue par sa capacité à équilibrer exploration et exploitation, garantissant une convergence rapide vers des solutions quasi-optimales.

Notre projet de fin d'études propose d'appliquer le SSA à l'ordonnancement des tâches dans le Cloud computing, avec pour objectif de minimiser simultanément trois critères clés : le makespan, le coût et la consommation énergétique.

Une étude comparative avec les algorithmes NSGA-II (Non-dominated Sorting Genetic Algorithm II) et PSO (Particle Swarm Optimization) permettra d'évaluer la pertinence de cette approche. Pour ce faire, nous utilisons le simulateur CloudSim, couplé à une interface développée en Java, afin de modéliser des environnements Cloud hétérogènes et de valider expérimentalement nos résultats.

Structuré en trois chapitres, ce travail débute, dans le premier chapitre, par une analyse approfondie du Cloud computing, de ses modèles de service, de ses avantages et de ses défis. Le Chapitre 2 explore les métaheuristiques, en mettant l'accent sur le SSA et ses mécanismes bio-inspirés. Enfin, le chapitre 3 présente l'implémentation pratique, l'interface développée et les résultats obtenus, démontrant la supériorité du SSA dans l'optimisation multi-objectifs.

À travers ce travail, nous visons à contribuer à l'amélioration des stratégies d'allocation des ressources Cloud, en adaptant une méthode à la fois robuste, économe en énergie et adaptée aux exigences dynamiques des infrastructures modernes.

Chapitre 1 : Cloud Computing

1.1 Introduction

Ces dernières années, la transformation numérique des industries et la croissance exponentielle de la génération de données ont nécessité le développement d'environnements informatiques plus flexibles, évolutifs et rentables. L'informatique en nuage est apparue comme un paradigme révolutionnaire dans le domaine des technologies de l'information, offrant un accès à la demande à un pool partagé de ressources informatiques configurables. En éliminant la nécessité pour les organisations d'investir massivement dans une infrastructure physique, l'informatique en nuage a redéfini la manière dont les services informatiques sont consommés et fournis [1].

La dépendance croissante aux systèmes basés sur le cloud a créé de nouveaux défis en termes de gestion des ressources, de planification des tâches et d'optimisation, surtout avec la complexité croissante des infrastructures cloud. Pour résoudre ces problèmes, les chercheurs et les ingénieurs se sont tournés vers des solutions intelligentes et adaptatives, parmi elles, les algorithmes métaheuristiques. Ces algorithmes fournissent des mécanismes efficaces pour traiter les problèmes d'optimisation multi-objectifs et ont montré des résultats prometteurs dans l'amélioration de la performance et de la fiabilité des systèmes cloud [2].

Ce chapitre présente un aperçu approfondi de l'informatique en nuage, de ses principales caractéristiques, de ses modèles de service et de déploiement, de ses avantages et des défis critiques auxquels elle est confrontée. En outre, il introduit le rôle essentiel de l'optimisation dans l'informatique en nuage et jette les bases de l'application de l'algorithme de recherche d'écureuils (SSA), une métaheuristique inspirée de la nature dans ce domaine.

1.2 Définition du Cloud computing

Le Cloud computing est une technologie qui s'appuie sur Internet et des serveurs centraux distants pour stocker et gérer des données et des applications (voir Figure 1.1). Cela offre aux utilisateurs et aux entreprises la possibilité d'exploiter des applications sans besoin d'installation, ainsi que d'accéder à leurs documents privés sur tout ordinateur possédant une connexion Internet. Cette technologie facilite un calcul beaucoup plus performant en centralisant le stockage, la mémoire, le traitement et la largeur de bande [2].

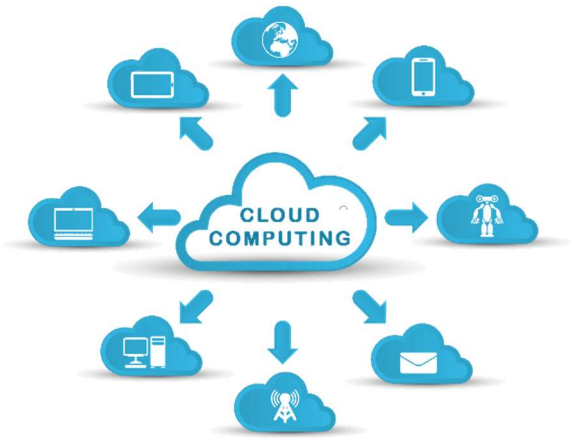


Figure 1.1 : Le concept du Cloud computing [9].

1.3 Caractéristiques du Cloud computing

Pour comprendre les concepts de base et les technologies du Cloud, nous déterminons les cinq caractéristiques majeures [3].

Service à la demande : Les ressources peuvent être obtenues dès que le client en fait la requête, généralement fournies de manière automatique par le fournisseur sans nécessiter d'interaction humaine.

- **Large accès au réseau** : Les ressources du réseau sont disponibles via des protocoles standardisés, compatibles avec différentes plateformes comme les smartphones, les ordinateurs portables et les PDA.
- **Ressources partagées** : Les utilisateurs utilisent conjointement les équipements fournis par le prestataire.
- **Elasticité rapide** : En fonction des exigences, les ressources et compétences sont déployables et modifiables automatiquement, à tout moment et pour n'importe quelle quantité.
- **Service mesuré :** On peut contrôler et ajuster toutes les ressources allouées pour mesurer leur usage, en faisant correspondre le degré d'abstraction au service spécifique, tel que la capacité de stockage, le traitement des données, le débit de transfert et les comptes utilisateurs en service.
- **Paiement à l'usage (pay-as-you-use)** : Le montant facturé dépend de l'usage réel, reflétant ainsi l'utilisation effective par l'utilisateur.

1.4 Types de déploiement Cloud computing

On reconnaît trois modes de déploiement des services cloud : privé, public et hybride, ce qui détermine le niveau d'accès des utilisateurs aux fournisseurs de cloud [4].

1.4.1 Cloud public

Le cloud public opère sur la base d'un modèle de Cloud computing classique, dans lequel un fournisseur met à disposition, par le biais d'Internet, des ressources telles que des applications et de l'espace de stockage au grand public, accessible à tous ceux qui souhaitent les utiliser ou les obtenir. Administré par un intervenant externe, il propose une structure souple et accessible [3]. Dans le cadre d'un Cloud public, le fournisseur met à disposition ses services via un réseau accessible pour ses utilisateurs (voir Figure 1.2). Ces solutions, utilisées par divers utilisateurs et acteurs économiques, opèrent dans un environnement partagé [5].

Figure 1.2 : Modèle d'un Cloud public [10].

1.4.2 Cloud privé

Dans un Cloud privé, la gestion des services s'effectue à l'intérieur d'un réseau protégé (voir Figure 1.3), séparé par un pare-feu. On peut mettre en place un cloud privé soit dans un centre de données personnel, soit en souscrivant à un service auprès d'un fournisseur externe, assurant de ce fait une sécurité et un contrôle optimaux [4]. La gestion peut être assurée par l'entreprise ou ses filiales, une configuration que l'on désigne comme « Le Cloud privé Interne ». Cet outil peut être administré soit par un fournisseur externe, soit par l'entreprise en question. Dans ce cas, on le désigne sous le terme « Cloud Privé Externe ». Son accès se fait via des réseaux sécurisés tels que le VPN (Réseau Privé Virtuel) [3].

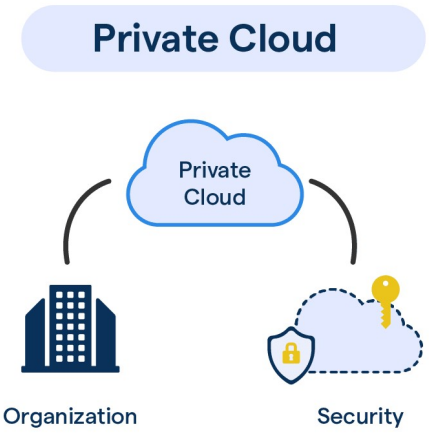


Figure 1.3 : Modèle d'un cloud privé [11].

1.4.3 Cloud hybride

L'adoption d'une stratégie hybride permet aux organisations de tirer parti des bénéfices du Cloud privé et public (voir Figure 1.4). Cela implique l'attribution des applications critiques au Cloud privé et la transition des autres vers le Cloud public, favorisant ainsi une diminution notable des coûts liés à l'infrastructure et à la gestion [3]. Les solutions hybrides allient l'utilisation d'un centre de données privé pour les informations sensibles à celle du Cloud public pour une extensibilité accrue [4].

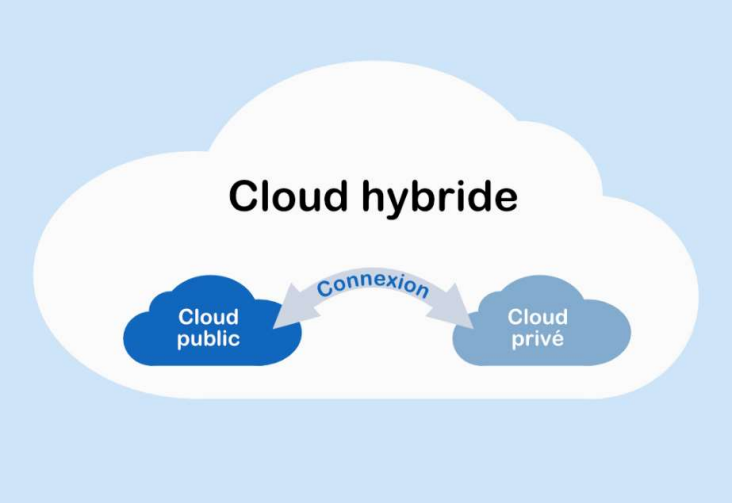


Figure 1.4 : Modèle d'un Cloud hybride [12].

1.5 Services du Cloud computing

Les services de Cloud computing peuvent être classés en trois types principaux :

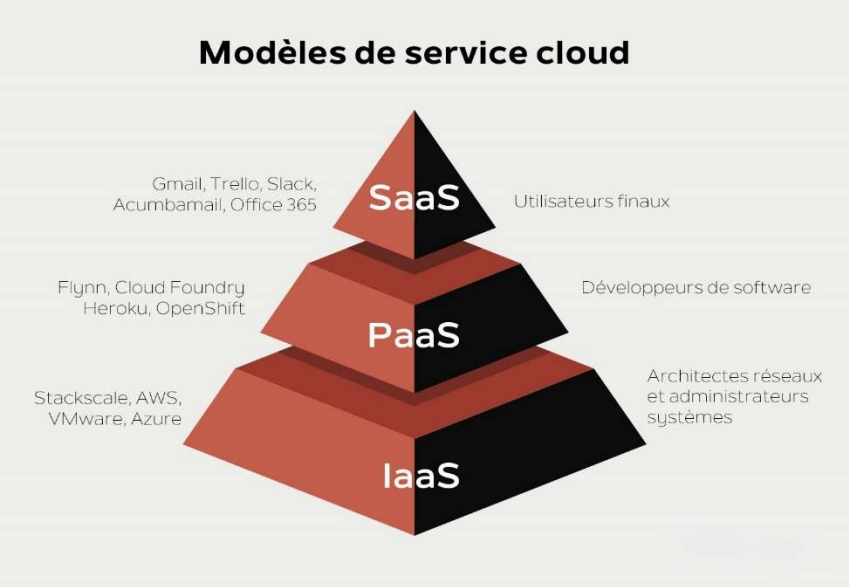


Figure 1.5 : Modèles de service Cloud [13].

1.5.1 Software as a service

Le modèle de distribution logicielle désigné sous l'appellation Software as a Service (SaaS) repose sur l'externalisation de l'hébergement applicatif, assurée par un prestataire tiers, et sur un accès aux services via le réseau Internet (voir Figure 1.6). Dans ce paradigme, les fournisseurs sont responsables non seulement de la mise à disposition des applications, mais également de leur maintenance, des mises à jour continues, ainsi que de l'administration de l'infrastructure technique sous-jacente. L'utilisateur final est ainsi déchargé des contraintes liées à l'installation, à la configuration et à la gestion du logiciel. Un exemple représentatif de ce modèle est Gmail, qui offre une solution de messagerie électronique immédiatement fonctionnelle, assortie d'un espace de stockage en ligne. D'autres services tels que Google, Twitter, Facebook et Flickr s'inscrivent également dans cette logique, en permettant une accessibilité ubiquitaire à leurs fonctionnalités depuis tout terminal connecté à Internet [4].

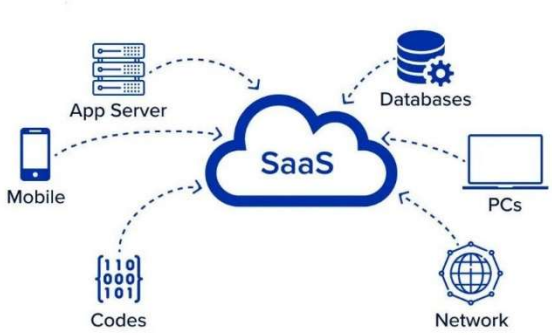


Figure 1.6 : Software as a service (SaaS) [14].

1.5.2 Platform as Services

La plateforme Platform as a Service (PaaS) constitue un environnement intégré permettant aux utilisateurs de concevoir, de gérer et de déployer des applications, tout en déléguant la gestion de l'infrastructure sous-jacente aux fournisseurs de services (voir Figure 1.7). Ce modèle libère les entreprises des tâches techniques liées à la sécurité, à l'administration des systèmes d'exploitation, à la gestion des serveurs ainsi qu'aux procédures de sauvegarde. Un exemple emblématique de ce type de service est AWS Elastic Beanstalk proposé par Amazon, qui permet l'automatisation du déploiement d'applications sur un ensemble d'instances virtuelles, optimisant ainsi les processus de mise en production [3].

Figure 1.7 : Platform as Services (PaaS) [15].

1.5.3 Infrastructure as a services

Dans le modèle Infrastructure as a Service (IaaS), le fournisseur met à disposition des clients des ressources informatiques fondamentales, telles que les serveurs, les capacités de stockage et les équipements réseau (voir Figure 1.8). Ce modèle permet aux utilisateurs de contrôler et de gérer leur propre système d'exploitation, leurs applications, leurs configurations de stockage ainsi que certains composants réseau, tout en étant abstraits de la gestion de l'infrastructure physique sous-jacente. L'IaaS repose sur la virtualisation des fonctions administratives, ce qui permet un gain de temps significatif et une concentration accrue sur des projets à forte valeur ajoutée. Contrairement à l'acquisition de matériel, ce modèle repose sur une facturation à l'usage. Un exemple illustratif est Netflix, qui exploite l'infrastructure d'Amazon Web Services (AWS) pour héberger ses services, tout en assurant de manière autonome la gestion de sa plateforme, la sécurisation des transactions, ainsi que la diffusion de contenu en streaming [3].

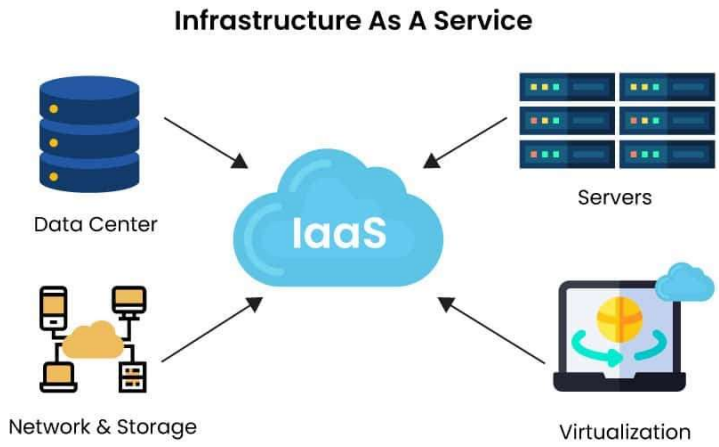


Figure 1.8 : Infrastructure as a services (IaaS) [16].

1.6 Évolution du Cloud computing

L'évolution du Cloud computing s'inscrit dans la continuité des concepts d'informatique utilitaire et de virtualisation. À ses débuts, les services informatiques étaient centralisés autour des ordinateurs centraux (mainframes). Avec l'émergence des ordinateurs personnels, une phase de décentralisation a marqué le paysage informatique. Aujourd'hui, on assiste à un retour vers la centralisation, cette fois à travers les services Cloud. Ce

changement de paradigme s'explique notamment par la croissance exponentielle des besoins en traitement de données, la généralisation de l'accès à Internet et les progrès majeurs réalisés dans les technologies de virtualisation. Ces facteurs ont joué un rôle déterminant dans le développement et l'adoption massive des services Cloud [1] [5].

1.7 Sécurité du Cloud computing

La sécurité dans le Cloud fait référence à l'ensemble des dispositifs techniques, politiques et procédures destinés à garantir la confidentialité, l'intégrité et la disponibilité des données hébergées dans des environnements dématérialisés. Elle concerne à la fois les infrastructures Cloud privées, utilisées pour l'hébergement d'applications sur site, et les solutions de stockage en ligne standardisées, telles que Google Drive ou Box, qui offrent un espace de stockage préconfiguré aux utilisateurs. Dans le paradigme du Cloud computing, la responsabilité principale de la sécurité des données incombe au fournisseur de services, qui doit assurer la mise en œuvre de mesures de protection avancées. Toutefois, afin de bénéficier d'un niveau de sécurité optimal, il est impératif que les clients procèdent à une évaluation rigoureuse des prestataires, en privilégiant ceux qui intègrent la sécurité comme élément central de leur architecture Cloud [6].

1.8 Avantages et inconvénients du Cloud computing

1.8.1 Avantages du Cloud computing

De nombreuses études ont mis en évidence les bénéfices associés à l'adoption du Cloud Computing. Parmi ces avantages, on peut citer [3] :

- **Flexibilité** : Grâce au modèle de mutualisation des ressources (multi-location), le Cloud permet une allocation dynamique et granulaire des ressources en fonction des besoins en temps réel. Cette capacité d'adaptation favorise une meilleure évolutivité des applications et une optimisation de leur performance.
- **Optimisation des coûts** : Le Cloud computing permet de réduire significativement les dépenses liées à l'infrastructure informatique, notamment par la diminution des effectifs dédiés à la gestion technique et l'adoption de modèles tarifaires basés sur la consommation réelle (pay-as-you-go). Ce modèle élimine ainsi le besoin d'investissements initiaux importants.
- **Portabilité** : Le Cloud offre une portabilité des ressources informatiques, rendant les services et applications accessibles depuis n'importe quel emplacement géographique disposant d'un accès réseau, ce qui favorise la mobilité et la continuité des opérations.
- **Simplicité d'utilisation** : Les plateformes Cloud proposent des interfaces conviviales et des services facilement déployables, utilisables de manière transparente via des navigateurs web. Cette simplicité améliore l'expérience utilisateur et réduit le temps de mise en œuvre.

1.8.2 Inconvénients du Cloud computing

Bien que le Cloud computing offre de nombreux avantages, il présente également certaines limitations qu'il convient de prendre en considération [3] :

- **Gestion énergétique** : La planification efficace de l'utilisation des ressources dans le Cloud requiert une stratégie de gestion énergétique rigoureuse. Le fournisseur de

services doit mettre en place des politiques spécifiques afin de minimiser la consommation électrique des infrastructures.

- **Confidentialité et sécurité** : La sécurité des données reste l'un des enjeux majeurs du Cloud computing. Les risques liés à la confidentialité, notamment durant le transfert ou le stockage dans des environnements publics, rendent les systèmes vulnérables aux attaques potentielles, exigeant des mécanismes de protection renforcés.
- **Gestion des ressources** : La gestion des ressources dans un environnement virtualisé peut s'avérer complexe en raison de la multiplicité et de la variabilité des machines virtuelles. Cette complexité peut engendrer des problèmes de performance et de surcharge si elle n'est pas correctement maîtrisée.
- **Dépendance au fournisseur** : Lorsque les besoins spécifiques d'un client ne sont pas couverts par l'offre standard du fournisseur, il peut être difficile d'obtenir des personnalisations. Cela impose aux clients de choisir méticuleusement un fournisseur fiable, en tenant compte de leur capacité à répondre à des exigences particulières.

1.9 Rôle de l'optimisation dans le Cloud computing

Avec l'accroissement de la complexité des environnements Cloud, la gestion efficace des ressources devient un enjeu crucial. Les techniques d'optimisation sont largement utilisées pour améliorer les performances globales des systèmes Cloud, notamment à travers les aspects suivants [7] :

- **Ordonnancement des tâches** : L'allocation optimale des tâches aux machines virtuelles (VM) permet de maximiser les performances tout en réduisant les temps de réponse.
- **Répartition de charge (Load Balancing)** : Elle vise à équilibrer la charge de travail entre les serveurs afin d'éviter les surcharges et de garantir un fonctionnement fluide.
- **Efficacité énergétique** : Réduction de la consommation énergétique des centres de données, contribuant ainsi à une infrastructure plus durable.
- **Réduction des coûts** : Diminution des coûts d'exploitation pour les fournisseurs comme pour les utilisateurs, grâce à une utilisation rationnelle des ressources.

1.10 Conclusion

Ce chapitre a présenté un aperçu complet de Cloud computing, détaillant sa définition, son évolution, ses modèles de service et de déploiement, ses caractéristiques principales, ses avantages et ses défis. En tant que pilier de l'infrastructure informatique moderne, le Cloud computing joue un rôle essentiel en permettant une prestation de services dynamique, évolutive et rentable.

Avec la complexité croissante des systèmes Cloud, il y a un besoin grandissant de solutions d'optimisation intelligentes pour gérer les ressources de manière efficace et garantir des performances élevées. Ce besoin souligne l'importance des algorithmes métaheuristiques avancés, qui sera exploré dans les chapitres suivants pour leur potentiel à résoudre des problèmes d'optimisation clés dans l'environnement de Cloud computing.

Chapitre 2 : Les métaheuristiques

2.1 Introduction

Les métaheuristiques sont des algorithmes puissants déployés pour dénicher des solutions satisfaisantes à des problèmes d'optimisation complexes, surtout lorsque les méthodes conventionnelles ne parviennent pas à les traiter avec efficacité. Ces algorithmes sont élaborés pour résoudre des problèmes qui sont ardus voire impossibles à résoudre de manière précise, comme ceux présentant des caractéristiques non linéaires, non différentiables ou combinatoires. À l'opposé des techniques d'optimisation classiques, les métaheuristiques ne requièrent pas une compréhension approfondie du problème ni ne s'articulent autour de formules mathématiques spécifiques, ce qui leur confère une flexibilité et une adaptabilité face à une multitude d'applications dans divers domaines tels que l'ingénierie, l'intelligence artificielle et l'économie. Les métaheuristiques tirent leur inspiration de processus naturels tels que l'évolution ou le comportement animal et exploitent ces concepts pour traquer les solutions optimales. Dans ce chapitre, nous allons découvrir les différentes façons de trouver les meilleures solutions à des problèmes. Ensuite, nous allons nous pencher sur une méthode particulière appelée Squirrel Search Algorithm(SSA) [8].

2.2 Les méthodes de résolution

Deux types de méthodes de résolution sont définis comme suit [17] :

2.2.1 Les méthodes exactes

Les méthodes exactes assurent une solution optimale en analysant toutes les options possibles. Cela comprend la programmation linéaire et entière. Ces méthodes sont efficaces pour les problèmes de petite taille, mais leur rendement se dégrade rapidement à mesure que l'ampleur du problème s'accroît en raison de leur complexité exponentielle. Bien qu'elles garantissent des solutions optimales, elles sont principalement utilisées comme référence pour évaluer les solutions obtenues à partir d’approches heuristiques et métaheuristiques.

2.2.2 Les méthodes approchées

Les méthodes approchées ou approximatives sont des techniques d'optimisation qui cherchent à dénicher rapidement de bonnes solutions sans assurer leur optimalité. Ces techniques conviennent parfaitement aux problèmes vastes et complexes pour lesquels les méthodes exactes sont coûteuses en termes de calcul. Les techniques heuristiques, comme les algorithmes gloutons, prennent des décisions locales pour parvenir à une solution, alors que les métaheuristiques telles que les algorithmes Génétiques (GA) et le Recuit Simulé (SA) mettent en œuvre des stratégies plus évoluées pour explorer l'espace de solutions.

Bien qu'ils ne garantissent pas la solution optimale, ces méthodes sont appréciées pour leur efficacité à traiter des problèmes de grande envergure.

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

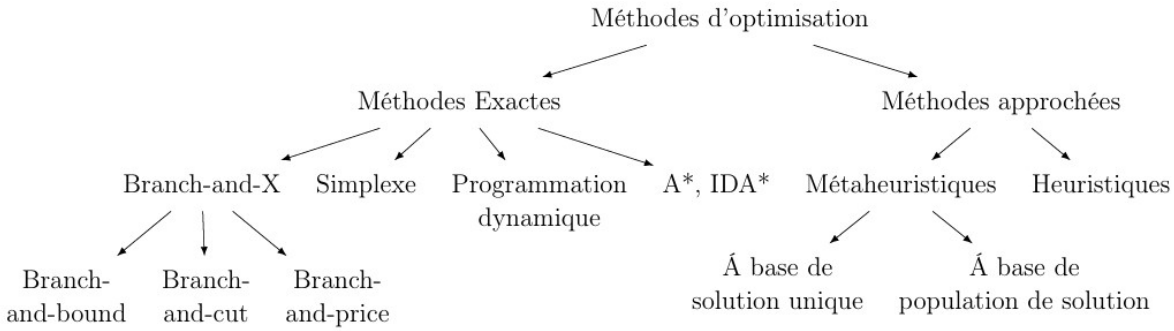


Figure 2.1: Classes des méthodes de résolutions [18].

2.2.2.1 Heuristique

Les méthodes heuristiques sont des stratégies d'optimisation visant à trouver des solutions acceptables et efficaces à des problèmes complexes, grâce à l'utilisation de processus décisionnels simplifiés. Ces approches s'appuient généralement sur l'expérience, les essais-erreurs ou les connaissances spécifiques à un domaine plutôt que sur une recherche exhaustive. Bien qu'ils ne garantissent pas des solutions optimales, ils s'avèrent spécifiquement efficaces pour les problèmes complexes et à grande échelle où les méthodes exactes traditionnelles seraient trop longues ou exigeantes en calcul [19].

L'algorithme glouton, une technique heuristique de prestige, effectue une série de choix optimaux localement dans l'espoir que ces choix mèneront à une solution optimale globale. Malgré leur simplicité et leur efficacité, les algorithmes gloutons peuvent parfois ne pas trouver la solution optimale, en particulier dans le cas de problèmes aux structures complexes.

2.2.2.2 Métaheuristique

Les métaheuristiques sont jugées plus efficaces que les méthodes heuristiques, grâce à leur capacité à traiter efficacement des problèmes d'optimisation plus vastes et plus complexes. Tandis que les méthodes heuristiques s'appuient généralement sur des prises de décision locales pour dénicher des solutions viables, les métaheuristiques mettent en œuvre des stratégies sophistiquées qui équilibrent exploration et exploitation afin d'éviter de se retrouver coincées dans des optima locaux. Ils sont conçus pour offrir des solutions de haute qualité dans un délai raisonnable, même pour les problèmes comportant des espaces de recherche vastes et complexes [20].

Les métaheuristiques s'inspirent de processus naturels, tels que les algorithmes évolutionnaires, l'intelligence en essaim et le recuit simulé. Ces méthodes examinent l'espace des solutions de manière plus systématique et flexible que les heuristiques, en utilisant souvent des processus d'amélioration itératifs qui imitent les systèmes naturels ou physiques. Des exemples courants de métaheuristiques incluent des algorithmes tels que l'Optimisation par Essaim de Particules (PSO) et l'Algorithme Génétique à Tri Non Dominateur II (NSGA-II). On les applique fréquemment à des problèmes d'optimisation combinatoire, non linéaire et multi-objectifs complexes, pour lesquels une solution exacte est impraticable en raison de la taille ou de la structure du problème [21].

2.3 Caractéristiques des métaheuristiques

Les métaheuristiques se caractérisent par [22] :

- Les stratégies d'optimisation sont élaborées pour orienter la recherche de solutions de haute qualité à des problèmes complexes.
- L'objectif principal des métaheuristiques est d'explorer de manière efficace l'espace des solutions afin de trouver des solutions quasi-optimales dans un délai raisonnable.
- Ces méthodes vont de techniques simples de recherche locale à des processus plus élaborés basés sur l'apprentissage.
- Les métaheuristiques sont généralement non déterministes et n'assurent pas des solutions optimales.
- Ils intègrent des dispositifs pour éviter de se coincer dans des optima locaux en explorant efficacement différentes zones de l'espace de solution.
- Les principes fondamentaux des métaheuristiques peuvent être exposés de manière générale et abstraite, sans dépendre d'un problème spécifique.
- Elles sont dirigées par une stratégie de niveau supérieur afin de préserver leur adaptabilité et leur efficacité.

2.4 Quelques algorithme inspirés de phénomènes naturels

2.4.1 algorithmes génétiques

Les algorithmes génétiques (GA) sont des techniques d'optimisation inspirées du processus de l'évolution naturelle. Ces algorithmes exploitent une population de solutions candidates qui évoluent à travers plusieurs générations pour découvrir des solutions de meilleure qualité. Le processus intègre des opérations cruciales comme la sélection, le croisement (recombinaison) et la mutation, qui s'appuient sur les principes de la génétique biologique. GA explore l'espace des solutions en simulant la sélection naturelle, où les individus les plus aptes ont une plus grande chance de se reproduire et de transmettre leur patrimoine génétique à la génération suivante, ce qui permet d'améliorer les solutions au fil du temps. Les algorithmes génétiques sont largement utilisés pour résoudre des problèmes d'optimisation complexes, en particulier dans les cas de problèmes combinatoires et de recherche où les méthodes traditionnelles pourraient ne pas être efficaces [23].

2.4.2 Optimisation par essaims de particules

L'optimisation par essaim de particules (PSO) est un algorithme d'optimisation inspiré de la nature, basé sur le comportement social observé chez les bancs de poissons et les vols d'oiseaux. C'est une méthode basée sur la population où chaque individu, ou particule, symbolise une solution potentielle au problème d'optimisation. Les particules se déplacent dans l'espace de solutions, modifiant leur position en fonction de leur meilleure solution personnelle et de la meilleure solution trouvée par l'ensemble du groupe. Le comportement collectif de l'essaim, dirigé par l'apprentissage individuel et l'interaction sociale. PSO se révèle particulièrement utile pour traiter les problèmes d'optimisation continue, et il a été largement utilisé dans des domaines tels que l'ingénierie, l'apprentissage automatique et l'ordonnancement des taches dans Cloud computing [24].

2.4.3 Algorithme de recherche d'écureuil

2.4.3.1 Inspiration

Algorithme de recherche d'écureuil (SSA) est une technique d'optimisation bio-inspirée, basée sur le comportement adaptatif des écureuils volants du sud (Glaucomys volans) (voir la Figure 2.2). Ces petits mammifères utilisent le vol plané pour couvrir efficacement de vastes distances à la recherche de nourriture, réduisant ainsi leur consommation d'énergie. Leur tactique de recherche de nourriture fluctue en fonction des saisons : durant l'automne, leur régime est largement composé de glands qu'ils accumulent, et ils mettent de côté des noix de caryer pour l'hiver, moment où ces stocks sont exploités pour satisfaire une demande élevée en énergie. Cette gestion géniale des ressources, alliée à leur capacité à se déplacer efficacement, a servi de source d'inspiration pour la modélisation mathématique de SSA. Des études comparatives et des analyses statistiques ont prouvé que le SSA surpasse de nombreux algorithmes traditionnels en matière de précision et de vitesse de convergence. Sa performance a été confirmée dans des applications concrètes, comme l'Expérience de Flux Thermique, attestant de sa solidité et de sa capacité à s'adapter à différents problèmes d'optimisation [26].

Figure 2.2: Southern flying squirrel [25].

2.4.3.2 Description de l'algorithme

- **Initialisation aléatoire**

Supposons qu'il y ait n écureuils volants dans une forêt, chacun étant représenté par un vecteur décrivant sa position. L'ensemble complet des positions pour tous les écureuils volants peut être structuré sous la forme d'une matrice [26] :

$$FS = \begin{bmatrix} FS_{1,1} & FS_{1,2} & \dots & \dots & FS_{1,d} \\ FS_{2,1} & FS_{2,2} & \dots & \dots & FS_{2,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ FS_{n,1} & FS_{n,2} & \dots & \dots & FS_{n,d} \end{bmatrix} \quad (1)$$

Où $FS_{i,j}$ représente la dimension j^{th} de l'écureuil volant numéro i, j.

L'emplacement initial de chaque écureuil volant dans la forêt est attribué en utilisant une distribution uniforme (Eq. (2)).

$$FS_{ij} = FS_L + U(0,1) \times (FS_U - FS_L) \quad (2)$$

FS_L et FS_U définissent respectivement les bornes inférieure et supérieure du $i^{ème}$ écureuil volant dans la $j^{ème}$ dimension.

$U(0,1)$ est un nombre aléatoire uniformément distribué dans l'intervalle $[0,1]$.

- **Evaluation de fitness**

La valeur de fitness de l'emplacement pour chaque écureuil volant est calculée en insérant les valeurs des variables de décision dans une fonction de fitness définie par l'utilisateur, et les valeurs correspondantes sont ensuite stockées dans le tableau suivant [26].

$$f = \begin{bmatrix} f_1([FS_{1,1}, FS_{1,2}, \dots, FS_{1,d}]) \\ f_2([FS_{2,1}, FS_{2,2}, \dots, FS_{2,d}]) \\ \vdots \\ \vdots \\ f_n([FS_{n,1}, FS_{n,2}, \dots, FS_{1,d}]) \end{bmatrix} \quad (3)$$

La valeur de fitness de l'emplacement de chaque écureuil volant reflète la qualité de la source alimentaire qu'il a trouvée, c'est-à-dire : une source optimale (noyer de Hickory), une source normale (chêne à glands) ou aucune source alimentaire (lorsque l'écureuil volant se trouve sur un arbre ordinaire), ce qui détermine également sa probabilité de survie

- **Tri, déclaration et sélection aléatoire d'objets**

Après avoir stocké les valeurs de fitness de l'emplacement de chaque écureuil volant, le tableau est trié par ordre croissant. L'écureuil volant ayant la valeur de fitness minimale est considéré comme étant sur un noyer de Hickory. Les trois écureuils suivants les mieux classés sont supposés être sur des chênes à glands, et ils sont censés se diriger vers le noyer de Hickory.

Les écureuils volants restants sont supposés se trouver sur des arbres ordinaires. Ensuite, par sélection aléatoire, certains écureuils sont considérés comme se dirigeant vers le noyer de Hickory, en supposant qu'ils ont satisfait leurs besoins énergétiques quotidiens. Les écureuils restants se dirigeront vers les chênes à glands afin de répondre à leurs besoins énergétiques journaliers.

Ce comportement de recherche de nourriture chez l'écureuil volant est toujours influencé par la présence de prédateurs. Ce comportement naturel est modélisé en utilisant un mécanisme de mise à jour des emplacements, basé sur une probabilité de présence de prédateur (P_{dp}) [26].

- **Générer de nouveaux emplacements**

Comme déjà mentionné, trois scénarios potentiels se présentent lors du comportement de recherche dynamique des écureuils volants. Dans chaque scénario, on suppose que sans la présence d'un prédateur, l'écureuil volant se déplace librement et efficacement à travers la forêt à la recherche de sa nourriture favorite. Toutefois, en présence d'un prédateur, l'écureuil devient prudent et doit effectuer un petit parcours aléatoire pour trouver un endroit proche où se cacher.

On peut modéliser mathématiquement ce comportement de recherche dynamique comme suit [27] :

Cas 1 :

Les écureuils volants situés sur les chênes à glands (FS_{at}) pourraient se déplacer vers les hickorys. Dans cette situation, on peut calculer les nouvelles positions des écureuils de la manière suivante :

$$FS_{at}^{t+1} = \begin{cases} FS_{at}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{at}^{t}),\ R_1 \geq P_{dp} \\ Random\ location \quad , \quad sinon \end{cases} \tag{4}$$

Où *dg* est la distance de vol plané aléatoire, *R1* est un nombre aléatoire dans l'intervalle [0,1], FS_{ht} est la position de l'écureuil volant qui a atteint le noyer de Hickory, et *t* désigne l'itération actuelle.

L'équilibre entre exploration et exploitation est atteint grâce à la constante de vol plané *Gc* dans le modèle mathématique. La valeur de cette constante influence de manière significative les performances de l'algorithme proposé. Dans le présent travail, la valeur de *Gc* est fixée à 1,9.

Cas 2 :

Les écureuils volants se trouvant sur des arbres ordinaires (FSnt) peuvent se déplacer vers des chênes à glands afin de satisfaire leurs besoins énergétiques quotidiens. Dans ce cas, la nouvelle position des écureuils peut être déterminée comme suit :

$$FS_{nt}^{t+1} = \begin{cases} FS_{nt}^{t} + d_g \times G_c \times (FS_{at}^{t} - FS_{nt}^{t}),\ R_2 \geq P_{dp} \\ Random\ location \quad , \quad sinon \end{cases} \tag{5}$$

Où R2 est un nombre aléatoire dans l'intervalle [0,1].

Cas 3 :

Certains écureuils qui se trouvent sur des arbres ordinaires et ont déjà consommé des glands peuvent se déplacer vers un arbre à noix de hickory afin d'y stocker des noix de hickory, qui pourront être consommées en période de rareté alimentaire. Dans ce cas, la nouvelle position des écureuils peut être obtenue comme suit :

$$FS_{nt}^{t+1} = \begin{cases} FS_{nt}^{t} + d_g \times G_c \times (FS_{ht}^{t} - FS_{nt}^{t}),\ R_3 \geq P_{dp} \\ Random\ location \quad , \quad sinon \end{cases} \tag{6}$$

Où R3 est un nombre aléatoire c dans l'intervalle [0,1]. Dans toutes les situations étudiées, la probabilité de présence du prédateur (P_{dp}) est fixée à 0,1 pour cette étude.

La Figure 2 montre le modèle conceptuel du vol plané utilisé par les écureuils volants pour leur déplacement efficace lors de la recherche de nourriture pendant la nuit. Un écureuil volant plane en modifiant les forces de portance et de traînée.

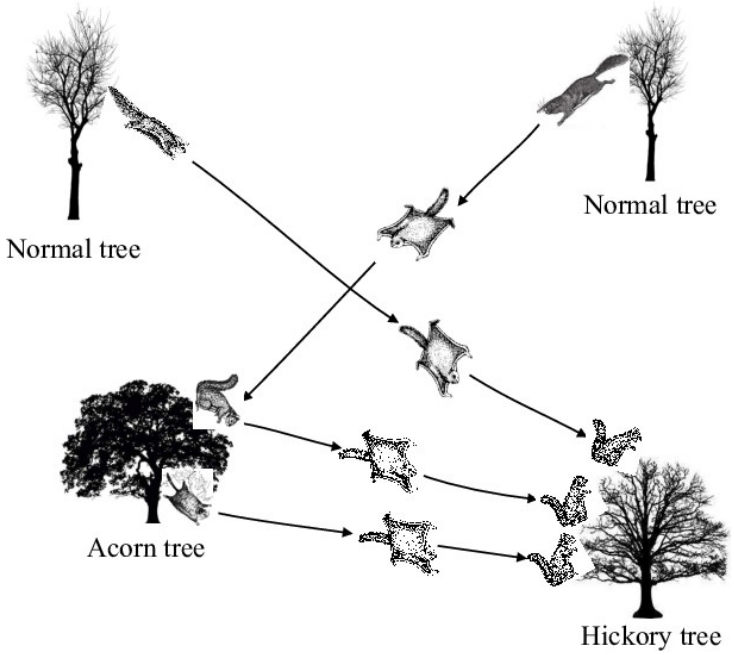


Figure 2.3: Modèle conceptuel du déplacement d'un écureuil volant d'un arbre à un autre en utilisant le vol plané [31].

- **Aérodynamique du glissement**

Le mécanisme de vol plané des écureuils volants est décrit par un plané en équilibre, dans lequel la somme des forces de portance (L) et de traînée (D) produit une force résultante (R) dont la magnitude est égale et opposée à la direction du poids de l'écureuil volant (Mg). Ainsi, R fournit une trajectoire de vol plané linéaire (voir Figure 2.4) à l'écureuil volant, à une vitesse constante.

Dans ce travail, un modèle approximatif du comportement de vol plané (Figure 2.5) est utilisé dans la conception de l'algorithme d'optimisation.

Un écureuil volant qui plane à vitesse constante descend toujours selon un angle ϕ par rapport à l'horizontale, et le rapport portance/traînée ou rapport de finesse est défini comme suit :

$$L/D = 1/tan\theta \quad (7)$$

Portance (L) : force qui permet à un objet de rester en l'air.
Traînée (D) : force de résistance de l'air qui s'oppose au mouvement.

Les écureuils volants peuvent prolonger la distance de leur plané en réduisant l'angle de descente (ϕ), ce qui améliore le rapport portance-traînée. Ici, la portance résulte de la déviation descendante de l'air passant au-dessus des ailes et est définie comme suit :

$$L = 1/(2\rho C_L)V^2S \quad (8)$$

Où représente $\rho(= 1.204kgm^{-3})$ est la densité de l'air, C_L est désigné comme le coefficient de portance, $V(= 5.25ms^{-1})$ c'est la vitesse et $S(= 154cm^2)$ représente la surface corporelle. La traînée de frottement est définie comme suit [20] :

$$\mathrm{D} = 1/2pV^2SC_D \quad (9)$$

Où C_D représente le coefficient de traînée de frottement. À faible vitesse, ce composant de traînée est très important, tandis qu'à grande vitesse, il devient moins significatif. Par conséquent, Eq. (7) détermine l'angle de de plané en régime permanent :

$$\phi = \arctan(D/L) \quad (10)$$

On calcule la distance de plané approximative (d_g) comme suit :

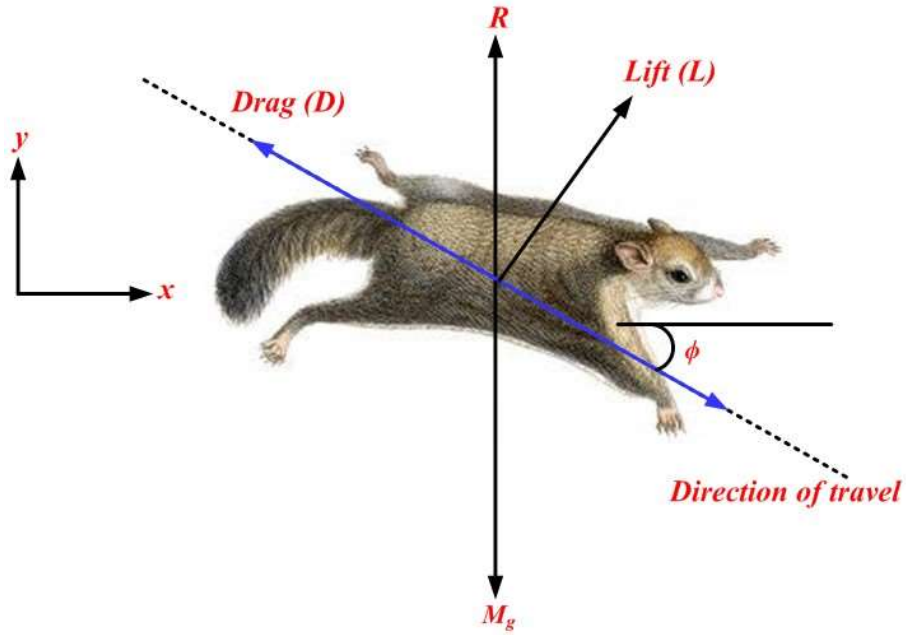
$$d_g = \left(\frac{h_g}{\tan\phi}\right). \quad (11)$$

Où $h_g (= 8\text{m})$ est la perte de hauteur survenue après le vol plané. Toutes les valeurs paramétriques nécessaires pour le calcul de dg, y compris C_L et C_D, sont extraites des données réelles. Ainsi, un écureuil volant peut modifier la distance de son vol en changeant simplement le rapport portance-traînée, conformément à l'emplacement d'atterrissage souhaité.

Dans ce travail, les simulations sont effectuées en intégrant des variations aléatoires dans C_L, avec une plage de $0.675 \leq C_L \leq 1.5$, tandis que C_D est considéré comme constant à 0.60.

La valeur de dg est assez élevée et peut provoquer d'importantes perturbations dans Eq. (4), Eq. (5) et Eq. (6), ce qui pourrait nuire à la performance de l'algorithme.

d_g est divisé par une valeur non nulle appropriée appelée facteur d'échelle $(\text{sf}) = 18$.



(a) Flying squirrel gliding at equilibrium

Figure 2.4:: Un écureuil volant en équilibre [29].

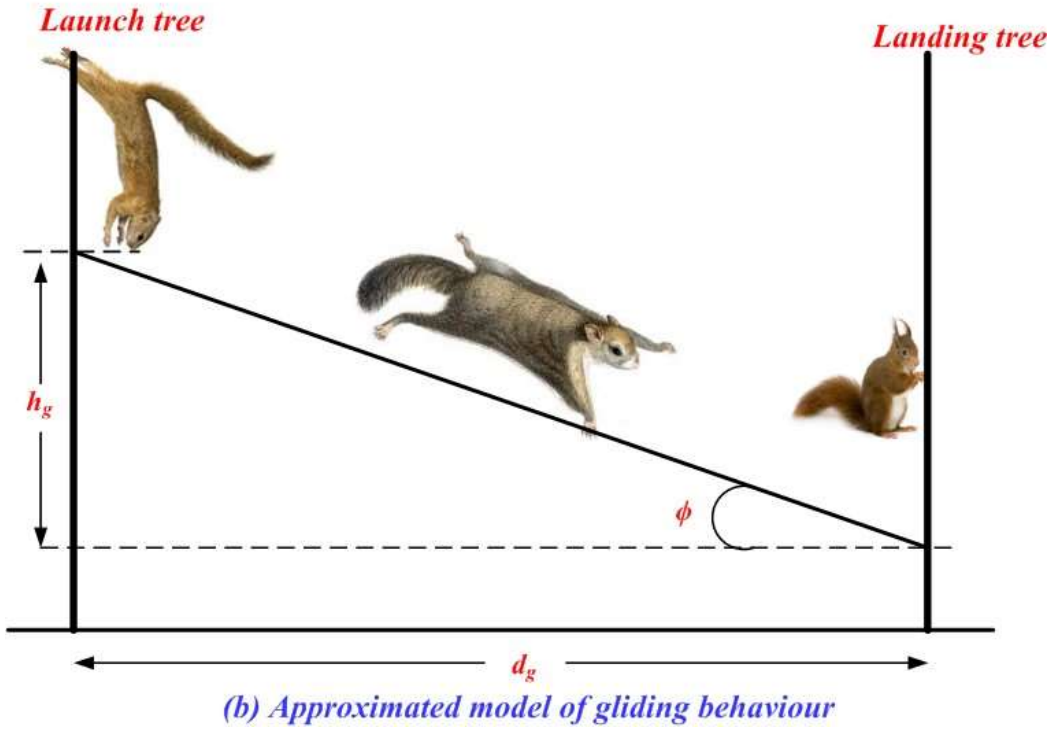


Figure 2.5: Modèle approximatif du comportement de vol [29].

- **Condition de surveillance périodique**

Les changements saisonniers affectent de manière significative l'activité de recherche de nourriture des écureuils volants. Ils subissent une perte de chaleur importante à basse température, car ils possèdent une température corporelle élevée et une petite taille, ce qui rend la recherche de nourriture coûteuse en énergie et risquée en raison de la présence de prédateurs actifs. Les conditions climatiques les obligent à être moins actifs en hiver qu'en automne.

Ainsi, le déplacement des écureuils volants est affecté par les variations météorologiques, et l'intégration de ce comportement peut offrir une approche plus réaliste pour l'optimisation. Par conséquent, une condition de surveillance saisonnière est introduite dans l'algorithme SSA, ce qui empêche l'algorithme proposé de rester coincé dans des solutions optimales locales.

Les étapes suivantes sont impliquées dans la modélisation du comportement [28] :

a. D'abord, calculez la constante saisonnière (Sc) en utilisant l'équation (12).

$$S_c^t = \sqrt{\sum_{k=1}^{d} (FS_{at,k}^t - FS_{ht,k})^2} \qquad (12)$$

b. Vérifiez la condition de surveillance saisonnière, c'est-à-dire St < Smin où Smin est la valeur minimale de la constante périodique qui est calculée comme suit :

$$S_{min} = \frac{10E^{-6}}{(365)^{t/(tm/2.5)}} \qquad (13)$$

Où t et t_m représentent respectivement les valeurs d'itération actuelles et maximales. La capacité de la méthode proposée à explorer et exploiter est influencée par la valeur S_{min}. Des valeurs plus élevées de S_{min} favorisent l'exploration, tandis que des valeurs plus faibles améliorent la capacité d'exploitation de l'algorithme. Pour qu'un métaheuristique soit efficace, il faut trouver un équilibre approprié entre ces deux phases. Cependant, cet équilibre est maintenu par un taux de plané constant Gc (Eq. (4), Eq. (5) et Eq. (6)), mais il peut être amélioré en adaptant de manière dynamique la valeur de S_{min} au cours des itérations.

c. Si une condition de surveillance périodique est vérifiée (c'est-à-dire, la saison d'hiver est terminée), déplacez aléatoirement les écureuils volants qui n'ont pas pu explorer la forêt à la recherche de nourriture optimale pour l'hiver.

- **Déplacement aléatoire à la fin de la saison d'hiver**

Comme discuté précédemment, la fin de la saison hivernale rend les écureuils volants actifs en raison du faible coût de recherche de nourriture. Les écureuils volants qui n'ont pas pu explorer la forêt pour trouver une source de nourriture optimale en hiver et qui ont malgré tout survécu peuvent chercher de la nourriture dans de nouvelles directions. L'intégration de ce comportement dans la modélisation peut améliorer la capacité d'exploration de l'algorithme proposé.

On suppose que seuls les écureuils n'ayant pas réussi à trouver la source de nourriture (noix de hickory) et ayant survécu, se déplaceront dans différentes directions afin de trouver une meilleure source de nourriture. Le déplacement de ces écureuils volants est modélisé à travers l'équation suivante [28] :

$$FS_{nt}^{new} = FS_L + Lévy(n) \cdot (FS_U - FS_L)\,. \tag{14}$$

Où la distribution de Lévy est utilisée pour favoriser une exploration efficace et optimale de l'espace de recherche. Lévy flight est un outil mathématique puissant utilisé par les chercheurs pour améliorer la capacité d'exploration globale de diverses métaheuristiques. Les déplacements en vol de Lévy aident à découvrir de nouvelles solutions candidates éloignées de la meilleure solution actuelle.

C'est une marche aléatoire où la longueur des pas est tirée d'une distribution de Lévy. Cette distribution est fréquemment décrite par la formule de loi de puissance $L(s) \sim |s|^{-1-\beta}$ où $0 < \beta \leq 2$. La distribution de Lévy est formulée mathématiquement comme suit :

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}}\, exp\left[-\frac{\gamma}{2(s-\mu)}\right] \frac{1}{(s-\mu)^{3/2}} & 0 < \mu < s < \infty \\ 0 & sinon \end{cases} \tag{15}$$

Où $\mu, \gamma > 0$. γ est le paramètre d'échelle et μ est le paramètre de décalage. Le vol de Lévy est calculé comme suit :

$$Lévy(x) = \frac{0.01 \times r_a \times \sigma}{|r_b|^{1/\beta}} \tag{16}$$

Où r_a et r_b sont deux nombres aléatoires distribués normalement dans l'intervalle [0,1], β est une constante estimée à 1.5 dans le travail actuel, et σ est calculé en tant que :

$$\sigma = \left[\frac{\Gamma(1+\beta) \times \sin\left(\left(\pi\beta\right)/2\right)}{\Gamma\left(\left(1+\beta\right)/2\right) \times \beta \times 2^{\left[\left(\beta-1\right)/2\right]}} \right]^{1/\beta} \quad (17)$$

tel que $\Gamma(x) = (x-1)!$.

- **Critère d'arrêt**

Le critère de convergence souvent utilisé est la tolérance fonctionnelle, où une valeur seuil, admissible mais petite, est définie entre les deux derniers résultats consécutifs. Il arrive parfois que le temps d'exécution maximal soit également employé comme critère d'arrêt. Dans l'étude actuelle, le nombre maximal d'itérations est considéré comme critère d'arrêt [28]. Le pseudocode de SSA est inclus dans la Figure 2.6 [30].

Set $Iter_{max}$, NP, n, P_{dp}, sf, G_c, FS_U and FS_L
Randomly initialize the flying squirrels locations
$$FS_{i,j} = FS_L + \text{rand}() * (FS_U - FS_L),\ i = 1, 2, \ldots, NP,\ j = 1, 2, \ldots, n$$
Calculate fitness value
$$f_i = f_i(FS_{i,1}, FS_{i,2}, \ldots, FS_{i,n}),\ i = 1, 2, \ldots, NP$$
while $Iter < Iter_{max}$
$$[sorted\_f, sorte\_index] = sort(f)$$
$$FS_{ht} = FS(sorte\_index(1))$$
$$FS_{at}(1:3) = FS(sorte\_index(2:4))$$
$$FS_{nt}(1:NP-4) = FS(sorte\_index(5:NP))$$
Generate new locations
for $t = 1$: $n1$ ($n1$ = total number of squirrels on acorn trees)
if $R_1 \geq P_{dp}$
$$FS_{at}^{new} = FS_{at}^{old} + d_g G_c (FS_{ht}^{old} - FS_{at}^{old})$$
else
$$FS_{at}^{new} = random\ location$$
end
end
for t =1: $n2$ ($n2$ = total number of squirrels on normal trees moving towards acorn trees)
if $R_2 \geq P_{dp}$
$$FS_{nt}^{new} = FS_{nt}^{old} + d_g G_c (FS_{at}^{old} - FS_{nt}^{old})$$
else
$$FS_{nt}^{new} = random\ location$$
end
end
for $t = 1$: $n3$ ($n3$ = total number of squirrels on normal trees moving towards hickory trees)
if $R_3 \geq P_{dp}$
$$FS_{nt}^{new} = FS_{nt}^{old} + d_g G_c (FS_{ht}^{old} - FS_{nt}^{old})$$
else
$$FS_{nt}^{new} = random\ location$$
end
end
$$S_c^t = \sqrt{\sum_{k=1}^{n} (FS_{at,k}^t - FS_{ht,k})^2},\ S_{cmin} = \frac{10E-6}{365^{Iter/(Iter_{\max})/2.5}}$$
if $S_c^t < S_{cmin}$
$$FS_{nt}^{new} = FS_L + \text{Lévy(n)} \times (FS_U - FS_L)$$
end
Calculate fitness value of new locations
$$f_i = f_i(FS_{i,1}^{new}, FS_{i,2}^{new}, \ldots, FS_{i,n}^{new}),\ i = 1, 2, \ldots, NP$$
$$Iter = Iter + 1$$
end

Figure 2.6: : pseudo code de l'algorithme SSA [30].

2.5 Conclusion

Dans ce chapitre, nous avons abordé la manière de résoudre des problèmes en utilisant des méthodes exactes et approximatives, en mettant l'accent sur les métaheuristiques. Le chapitre s'est intéressé en particulier à l'algorithme NSGA-II et PSO et nous avons approfondi la métaheuristique de l'algorithme de recherche en écureuil, en présentant ses caractéristiques et son pseudocode.

Dans le chapitre suivant, nous considérerons l'application de l'algorithme SSA pour l'ordonnancement de tâches multi-objectif dans le cadre du Cloud computing.

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

3.1 Introduction

Ce chapitre est dédié à l'implémentation de l'algorithme de recherche d'écureuil pour l'ordonnancement des tâches dans un environnement de Cloud computing. Ses performances sont comparées à celles des algorithmes NSGA-II et PSO, en se basant sur trois critères principaux : le makespan, le coût et la consommation énergétique.

De plus, le chapitre présente les outils de simulation et de développement utilisés lors de ce projet, tels que le langage de programmation Java, l'environnement de développement NetBeans, la bibliothèque JFreeChart pour visualiser les données et le simulateur CloudSim pour modéliser l'infrastructure du Cloud.

Par ailleurs, une description détaillée de l'interface utilisateur est présentée afin d'optimiser l'interaction entre l'utilisateur et le système. Enfin, une analyse approfondie des résultats obtenus est effectuée.

3.2 Environnement de développement et de simulation

3.2.1 Java

Java est un langage de programmation orienté objet, créé par Sun Microsystems en 1995 et géré aujourd'hui par Oracle. Il repose sur le principe « écrire une fois, exécuter partout » grâce à la compilation en bytecode indépendant de la plateforme, exécuté par la machine virtuelle Java (JVM), assurant ainsi une compatibilité interplateforme. Sa syntaxe, proche de C++, favorise la robustesse, la sécurité et la gestion automatique de la mémoire via le ramasse-miettes (garbage collector). Java supporte le multithreading et dispose d'une riche bibliothèque standard, ce qui le rend adapté à de nombreuses applications, des mobiles (notamment Android) aux serveurs web et systèmes embarqués. Le kit de développement (JDK) facilite la création de logiciels, tandis que l'environnement d'exécution (JRE) permet leur exécution. Sa portabilité et sa communauté active en font un des langages les plus populaires au monde [32].

3.2.2 Netbeans

NetBeans est un Environnement de Développement Intégré (IDE) gratuit et open-source, principalement dédié à la programmation en Java, mais qui offre aussi le support pour PHP, C/C++, HTML5 et JavaScript. Sous l'égide de la Fondation Apache Software, il propose des outils complets pour la programmation, le débogage, les tests et le déploiement d'applications sur diverses plateformes. Parmi les fonctionnalités notables, on peut citer la mise en évidence de la syntaxe, la complétion du code, le réfactorisation et un constructeur d'interface utilisateur par glisser-déposer pour Java Swing. Il s'intègre parfaitement avec des systèmes de contrôle de version tels que Git et SVN, des outils de construction comme Maven et Ant, ainsi que des serveurs tels que Tomcat et GlassFish [33].

3.2.3 JFreeChart

JFreeChart est une bibliothèque Java open-source et gratuite destinée à la création d'une grande diversité de graphiques de qualité professionnelle. Il prend en charge une multitude de types de graphiques tels que les graphiques à secteurs, les graphiques à barres (y compris horizontaux, verticaux, empilés et avec effet 3D), les graphiques linéaires… etc.

Parmi les fonctionnalités clés, on retrouve le zoom interactif, la gestion des événements, les infobulles et l'accès aux données via des interfaces définies. JFreeChart a la capacité

d'exporter les graphiques dans divers formats tels que JPEG, PNG, SVG et PDF en utilisant n'importe quelle implémentation Graphics2D [34].

3.2.4 CloudSim

CloudSim est un Framework open source qui facilite la simulation de l'infrastructure et des services de Cloud computing. C'est une création de l'organisation CLOUDS Lab de l'université de Melbourne, en Australie, et est entièrement codé en Java. Elle est utilisée pour modéliser et simuler un environnement de Cloud computing afin d'examiner des hypothèses avant la création de logiciels, en reproduisant des tests et des résultats [28].

CloudSim Core, le moteur de simulation principal, offre des interfaces pour la gestion des ressources comme les machines virtuelles (VM), la mémoire et la capacité de transfert de données des centres de données en virtualisation.

La gestion de la création et de l'exécution des entités importantes comme les VM, les Cloudlets et les hôtes est assurée par la couche CloudSim. Elle est aussi responsable de l'exécution réseau, de la fourniture des ressources et de leur gestion.

Le cadre de contrôle est supervisé par l'utilisateur, donnant ainsi au développeur la possibilité de définir les besoins matériels en fonction du scénario.

La couche CloudSim est composée de plusieurs classes, voici quelques-unes des classes les plus couramment utilisées durant la simulation [35] :

- **Centre de données (Data Center) :**

Infrastructure virtualisée centralisée qui regroupe des ressources physiques (serveurs, stockage, réseaux) et logicielles (hyperviseurs, orchestrateurs). Il sert de base matérielle pour le Cloud computing, garantissant une mutualisation dynamique des ressources grâce à des algorithmes d'attribution (processeur, mémoire vive, bande passante) et des stratégies de gestion énergétique. Les Hôtes (hôtes physiques) accueillent des machines virtuelles (VM) en suivant des schémas de provisionnement élastique, intégrant des systèmes de résilience et de disponibilité élevée.

- **DatacenterBroker :**

Le DatacenterBroker joue le rôle d'un intermédiaire intelligent entre les utilisateurs et les fournisseurs de services Cloud. Il étudie les besoins des utilisateurs (tels que la vitesse ou le prix) et discute avec les centres de données pour attribuer les ressources requises aux machines virtuelles. Grâce à l'utilisation de techniques d'optimisation, il garantit que les missions (Cloudlets) sont réalisées de manière efficace, tout en maintenant la qualité de service (QoS).

- **Host :**

Cette classe contient des informations essentielles, notamment sur la quantité de mémoire et d'espace de stockage, le type de processeur (pour illustrer une machine multi-cœur), ainsi que sur les règles d'allocation pour le partage de la puissance de traitement entre les machines virtuelles. Elle aborde également les directives concernant l'attribution de la mémoire et de la bande passante aux machines virtuelles.

- **VM :**

Une VM (Virtuel Machine) représente une simulation logicielle d'un ordinateur, qui opère sur un hôte matériel. Elle possède des ressources virtuelles (processeur, mémoire, stockage) qui sont attribuées de manière dynamique selon les exigences. Les machines virtuelles offrent une exécution flexible et isolée des tâches, avec une tarification basée sur

leur utilisation (heures, consommation de processeur). Elles peuvent être transitoires (effacées après utilisation) ou durables (conservées).

- **Cloudlet :**

Cette classe symbolise toute opération effectuée sur une machine virtuelle, telle qu'un traitement, un accès à la mémoire, etc. Elle renferme des paramètres qui décrivent les spécificités d'une tâche, tels que la durée, la taille, le nombre de millions d'instructions (mi). En plus, elle propose des méthodes pour déterminer le délai d'exécution, l'état, le coût et l'historique d'une tâche.

La figure 3.1 montre la structure logicielle de CloudSim [36] :

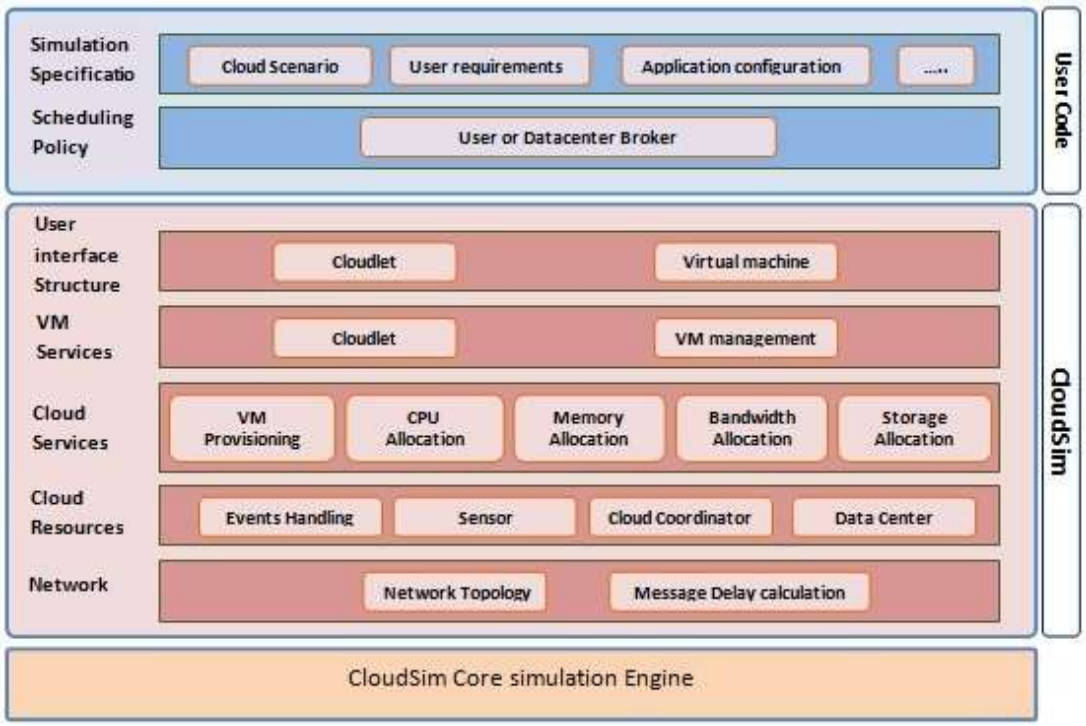


Figure 3.1 : Structure de CloudSim [36] .

3.3 Critères d'évaluation

3.3.1 Makespan

Dans le contexte du cloud computing, le makespan fait référence au temps nécessaire pour compléter un ensemble de tâches, calculé en fonction de la machine virtuelle (VM) ayant la plus longue durée d'exécution. Cette mesure essentielle illustre l'efficacité de la répartition des ressources : un temps de production élevé montre une surcharge de travail, alors qu'un temps de production optimisé assure une distribution uniforme des VMs.

La formule suivante (Eq. (18)) exprime cette idée en mesurant le temps total d'exécution par VM et en déterminant la valeur maximale [37].

$$Makespan = max_{j\in[1,m]} \left(\sum_{i=1}^{n} \frac{L_i}{M_j} \times \delta_{I,j}\right) \quad (18)$$

Où :

- L_i : Longueur (MI) de la tâche i*i* (Cloudlet.getCloudletLength())
- M_j : Capacité de calcul (MIPS) de la VM j*j* (Vm.getMips())
- $\delta_{I,j} = 1$ Si la tâche i*i* est assignée à la VM j*j*, sinon 00
- m : Nombre total de VMs,
- n : Nombre total de tâches

3.3.2 Coût

Dans le Cloud, le coût opérationnel est influencé à la fois par la durée d'exécution des tâches et par les caractéristiques énergétiques des processeurs employés. Chaque genre de processeur (comme le Turion MT34 ou le Pentium M) est lié à un coût unitaire e_c qui reflète son efficacité énergétique. Cette mesure évalue l'impact économique des répartitions de ressources, en punissant les machines virtuelles trop grandes ou consommantes d'énergie [37].

$$Coùt = \sum_{i=1}^{n} \frac{L_i}{M_j} \times e_c^{(j(i))} \quad (19)$$

3.3.3 Énergie

L'une des principales stratégies d'économie d'énergie à l'échelle du système que nous recommandons est le DVFS (Dynamic Voltage and Frequency Scaling), une approche d'adaptation dynamique de la tension et de la fréquence. La méthode DVFS est capable de modifier l'utilisation d'énergie et le temps d'exécution en ajustant la fréquence d'alimentation et la tension du processeur.

Dans ce travail, le modèle énergétique employé s'appuie sur les principes de consommation d'énergie des circuits à semi-conducteurs à oxyde métallique complémentaires (CMOS). Dans la technologie CMOS, les deux principales sources de consommation d'énergie sont la puissance statique, utilisée lorsque le circuit est en état stable, et la puissance dynamique, qui est dissipée lors des opérations de commutation du circuit. Environ 90% de la dissipation totale de puissance processeur est régulée par la consommation dynamique d'énergie. On utilise l'équation (Eq .(20)), qui considère les facteurs majeurs influant sur la consommation d'énergie pendant l'exécution des tâches, pour déterminer la puissance dynamique ($P_{dynamic}$). Cette méthode de modélisation permet de conserver la faisabilité computationnelle pour nos simulations tout en saisissant fidèlement les caractéristiques énergétiques des processeurs contemporains [39].

$$P_{dynamic} = AC_{ef}v^2f \quad (20)$$

Où :

A : le facteur d'activité défini comme le nombre moyen de transition par cycle d'horloge.

C_{ef}: la capacité effective.

v : la tension d'alimentation.

f : la fréquence d'horloge.

Étant donné que la consommation d'énergie est principalement influencée par la tension d'alimentation, comme le démontre (Eq. (20)), diminuer la tension est la méthode la plus

efficace pour réduire l'utilisation énergétique. On définit l'énergie E_c utilisée pour exécuter les tâches de workflow de la manière suivante :

$$E_c = \sum_{i=1}^{n} \gamma \, v_i^2 \, f_i \, wr_i^* \quad (21)$$

Tel que :

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

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

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

Ce dernier passe en mode veille lorsque le processeur n'est pas en activité. La fréquence relative et la tension d'alimentation sont à leurs niveaux les plus bas. Ainsi, nous définissons l'énergie consommée par les processeurs E_i pendant leurs phases d'inactivité de la manière suivante :

$$E_i = \sum_{j=1}^{m} \sum_{idle_{jk} \epsilon \, IDLE_k} \gamma \, v_{minj}^2 \, f_{minj} \, L_{jk} \quad (22)$$

Où :

$IDLE_j$ = Ensemble de toutes les périodes d'inactivité pour la machine p_j
$vmin_j$ = Tension d'alimentation minimale pour la machine p_j
$fmin_j$ = Fréquence de fonctionnement minimale pour la machine p_j
L_{jk} = Durée de la k-ième période d'inactivité ($idle_{jk}$) sur la machine p_j

L' (Eq. (23)) fournit l'expression de la consommation d'énergie totale (E_total) liée à l'achèvement du flux de travail dans un environnement cloud :

$$E_{total} = E_c + E_i \quad (23)$$

3.4 La méthode de la somme pondérée

La méthode de la somme pondérée (WSM) est une technique d'aide à la décision multicritères à la fois simple et largement utilisée, tant dans les contextes académiques que pratiques. Sa facilité de mise en œuvre la rend particulièrement adaptée à une utilisation courante pour les tâches d'optimisation [40].

- **Conditions 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 :

$$f(x) = \Sigma_{i=1}^{n} wi.fi \quad (24)$$

- Où w_i représente le poids attribué au critère f_i.

- **Contraintes de poids :**
  - **Non-négativité :** Les poids doivent respecter la condition $0 \leq w_i \leq 1$.
  - **Somme à un :** L'addition de tous les poids doit être égale à 1, c'est-à-dire que $\sum_{i=1}^{n} wi = 1$. (25)

- **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 :**

$$X_{\text{normalisé}} = \frac{x - x_{\min}}{x_{\max} - \min} \quad (26)$$

- **Critères de Maximisation :**

$$X_{\text{normalisé}} = \frac{x_{max} - x}{x_{max} - x_{\min}} \quad (27)$$

- **Fonction d'optimisation multi-objectifs :**

La fonction objectif agrégée combinant des critères normalisés est :

$$F = w_1 * Makespan + w_2 * Cout + w_3 * energie \quad (28)$$

Les priorités de l'utilisateur sont exprimées par le vecteur de poids w = {0.6, 0.2, 0.2}. Dans cette configuration, la minimisation du makespan est considérée comme la priorité principale (60%), tandis que l'optimisation de l'énergie et la réduction des coûts revêtent une importance égale (20% chacun).

3.5 IHM développée

Nous avons mis au point notre propre interface homme-machine (IHM) afin de faciliter l'accès et le contrôle de la simulation, car CloudSim ne fonctionne qu'en mode console et ne dispose pas d'une interface graphique.

Les interfaces ci-dessous montrent les étapes pour configurer la simulation.

Interface principale : l'interface principale est l'écran initial que vous découvrez lors de l'ouverture de notre application.

La Figure 3.2 présente cette interface, affichant l'ensemble des données relatives à notre projet de fin d'études.

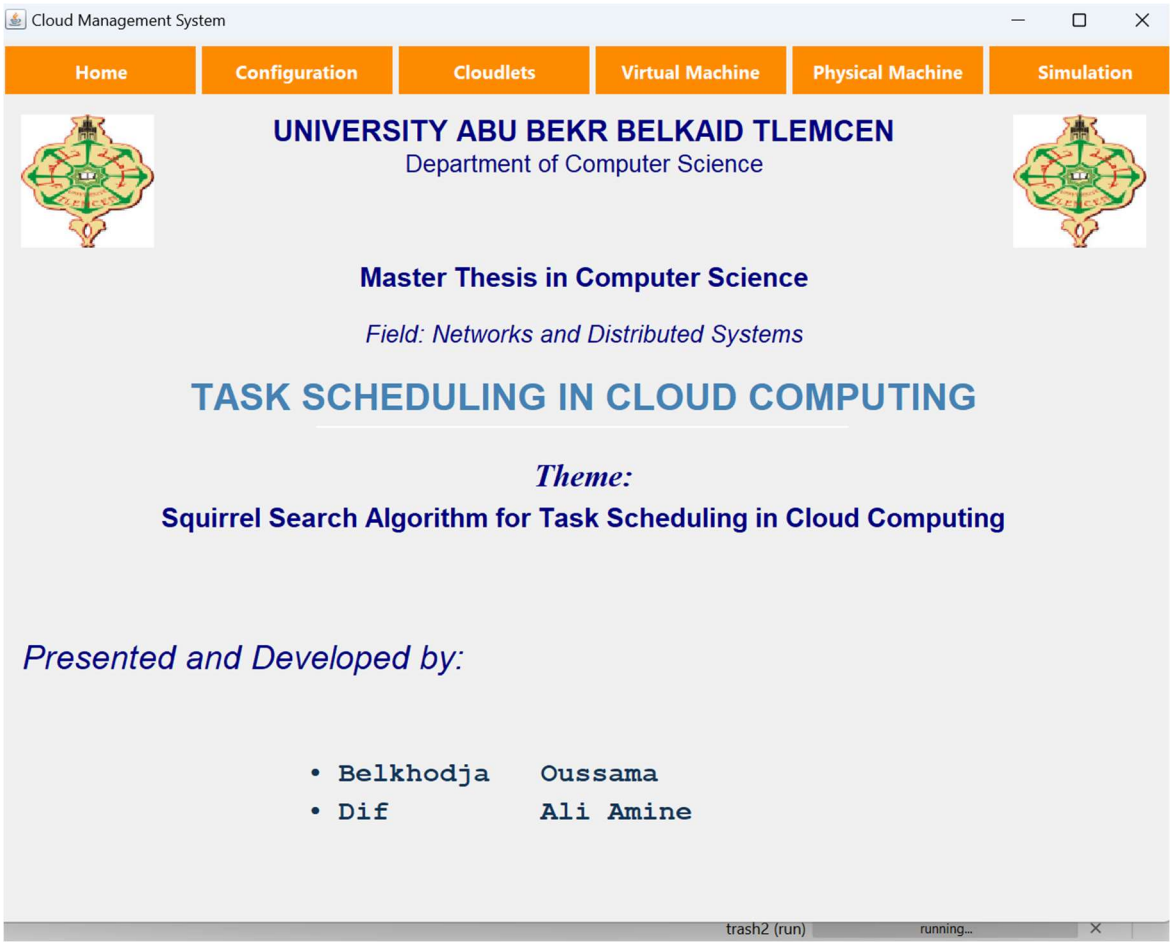


Figure 3.2 : Interface principale .

Choix de la configuration : L'interface propose à l'utilisateur deux modes de configuration : la configuration par défaut et la configuration personnalisée, comme le montre la Figure 3.3.

Ces options offrent à l'utilisateur un contrôle total sur la configuration de la simulation, lui permettant de choisir soit une configuration standard prédéfinie, soit d'ajuster les paramètres de simulation en fonction des besoins et des objectifs spécifiques. Notre application offre les deux modes de configuration.

Par défaut, l'option initiale conduit directement à l'interface de simulation, où l'utilisateur peut choisir le nombre de cloudlets et de machines virtuelles (VM), comme indiqué dans la Figure 3.3.

La seconde option offre une configuration personnalisée, donnant la possibilité d'accéder à et de modifier les paramètres liés aux cloudlets, aux machines virtuelles et aux machines physiques.

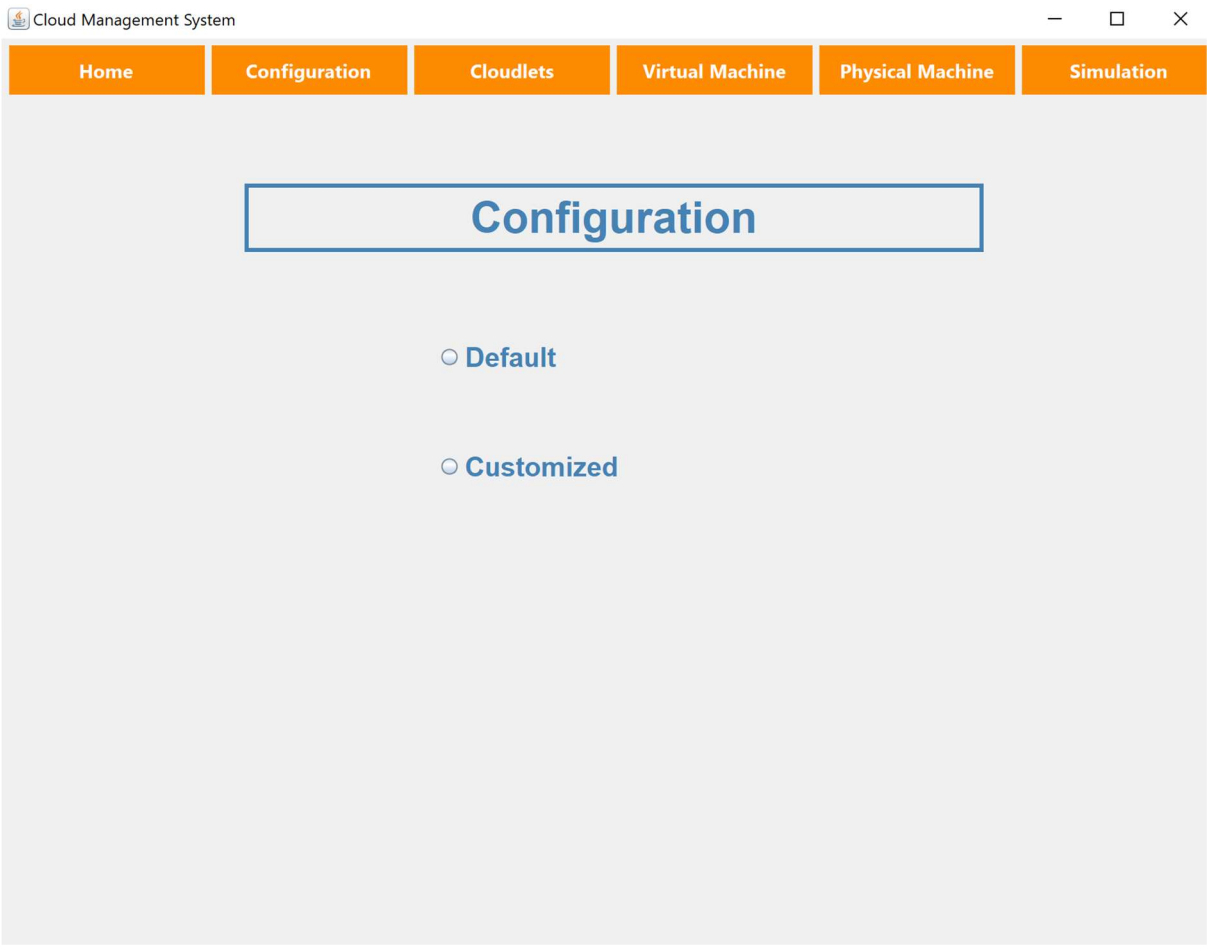


Figure 3.3: Choix de la configuration.

Lancement des résultats de simulation : l'interface illustrée à la Figure 3.4 s'affiche une fois que l'utilisateur a sélectionné la configuration par défaut. L'utilisateur peut déterminer le nombre de tâches via cette interface. On peut ensuite cliquer sur le bouton « Start Simulation » pour visualiser les résultats de la simulation.

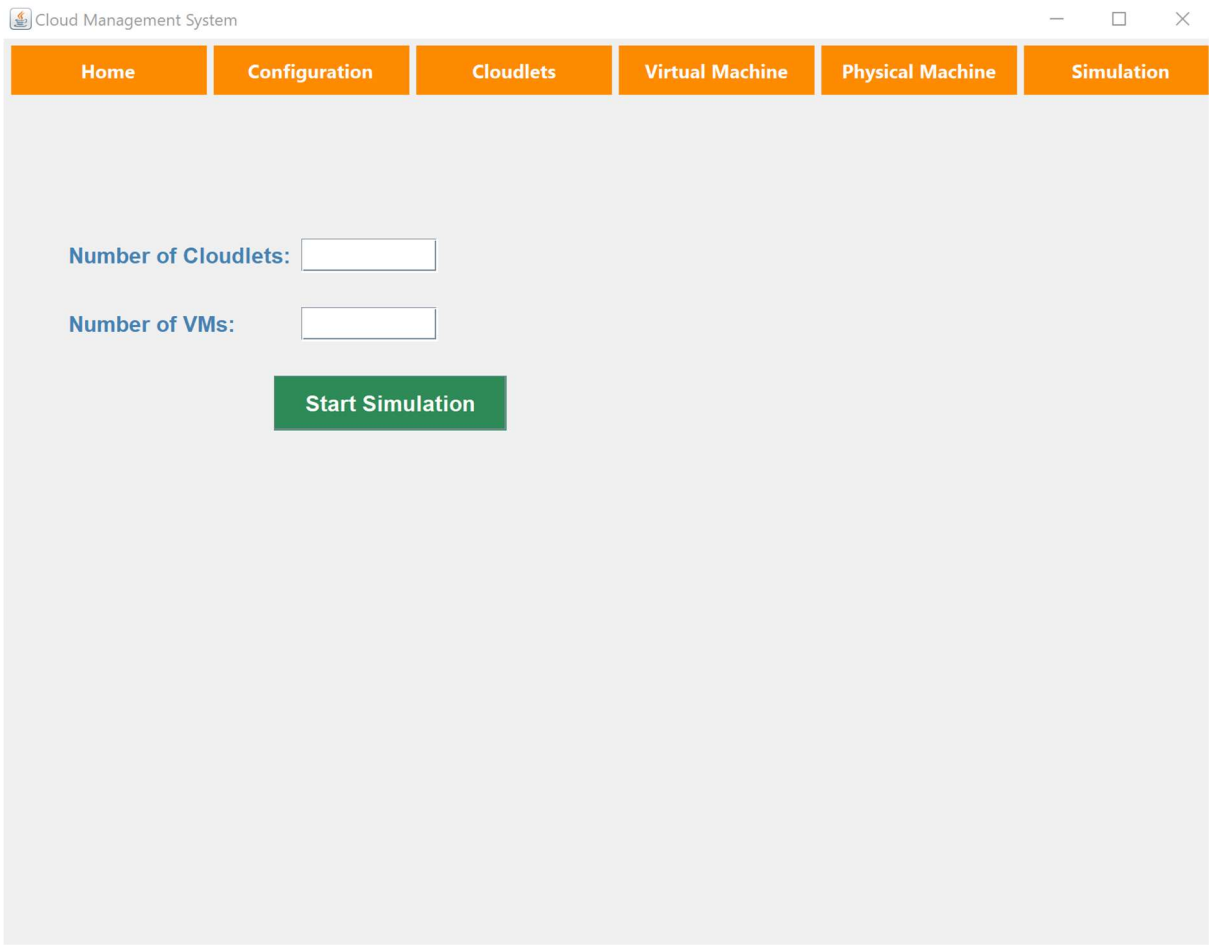


Figure 3.4 : Lancement de la simulation.

Configuration des Cloudlets : dans cette section qui illustrée à la Figure 3.5 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.

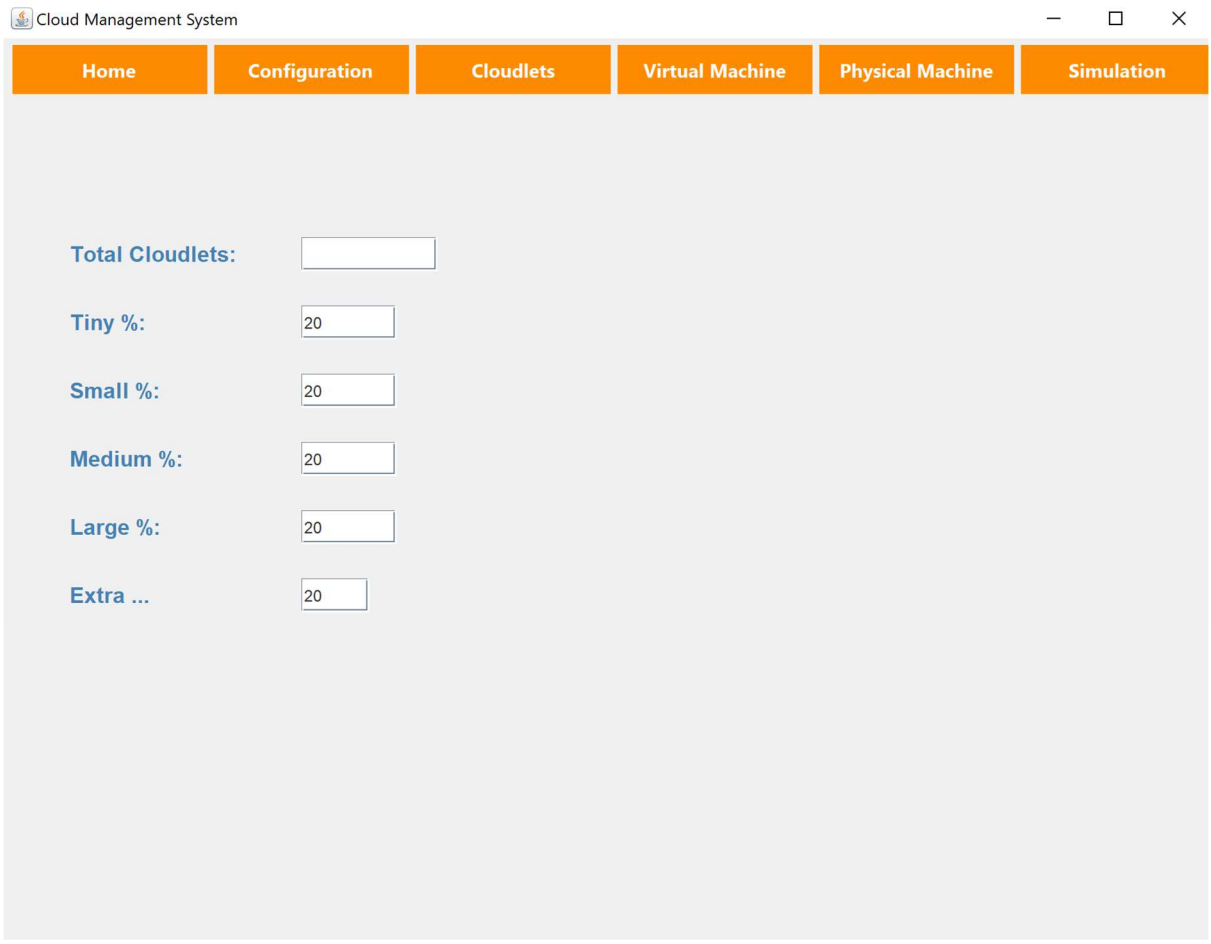


Figure 3.5 : Configuration des Cloudlets.

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 3.6, chaque machine virtuelle a une vitesse de traitement (MIPS) maximum et minimum, un coût d'exploitation, un taux d'échec, une capacité de mémoire vive, une bande passante et une capacité de stockage spécifiques.

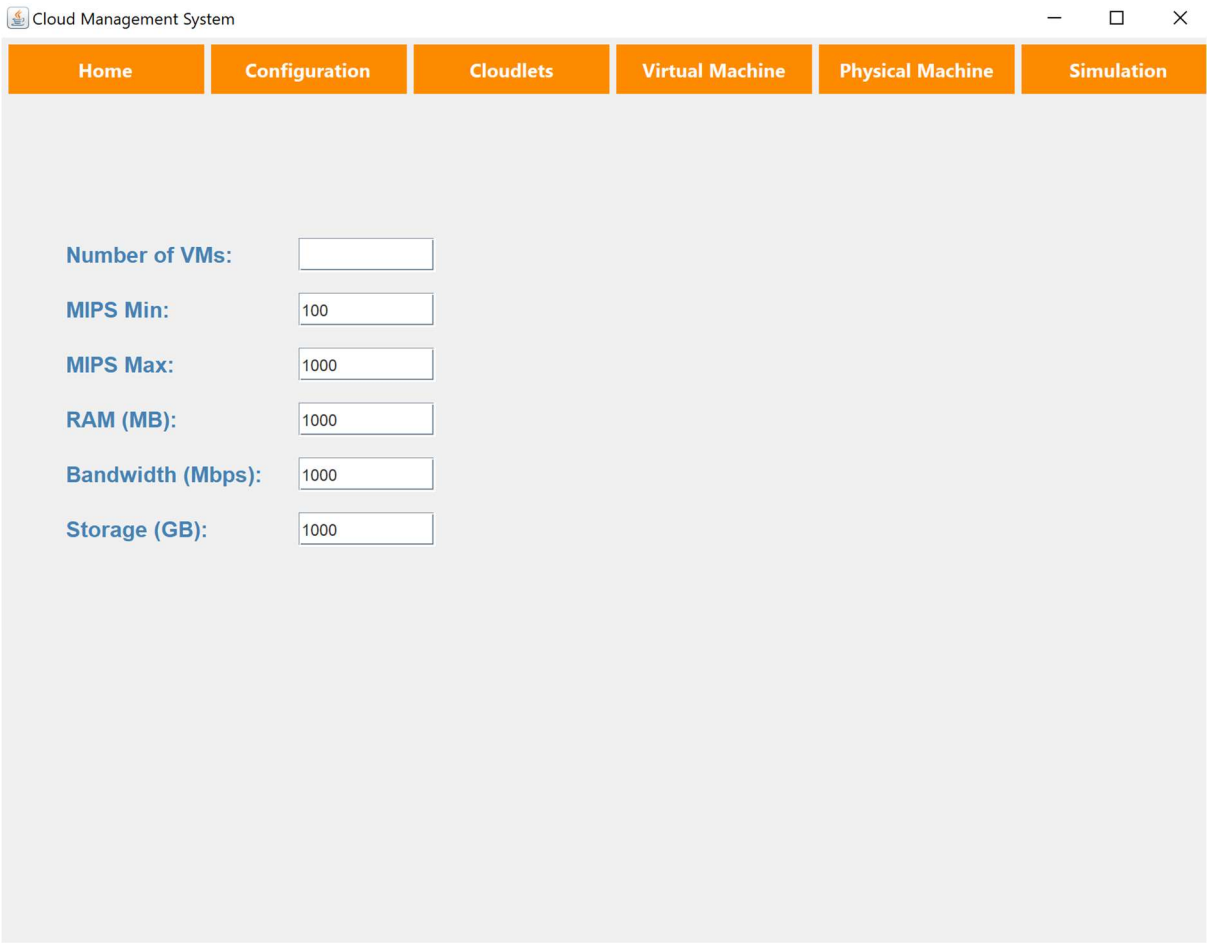


Figure 3.6 : configuration des machines virtuelles.

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 3.7, cela implique de déterminer le nombre d'hôtes par centre, le nombre d'éléments de traitement (PEs) par hôte, la vitesse de traitement de chaque PE (en MIPS), la quantité de RAM, la bande passante disponible et la capacité de stockage.

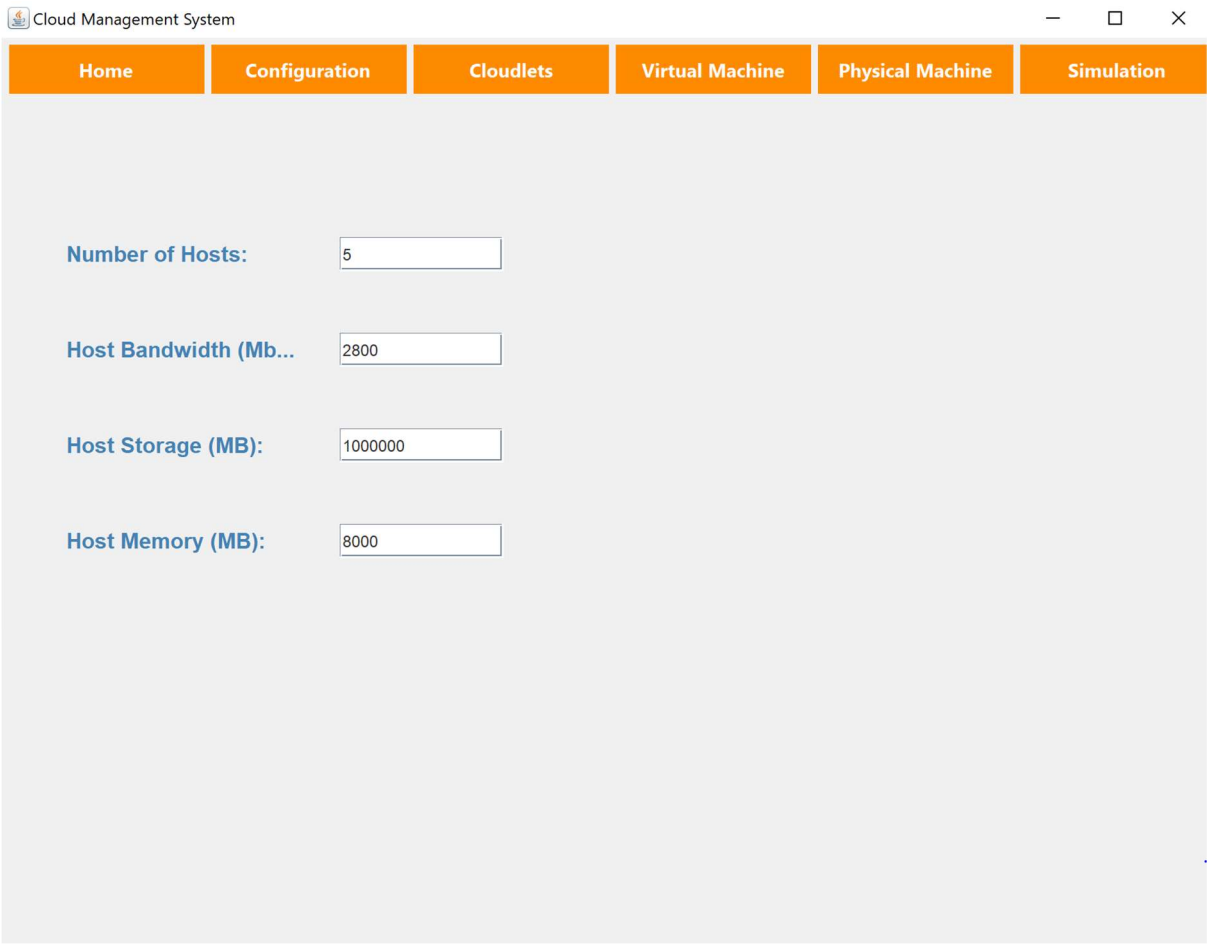


Figure 3.7 : Configuration des machines physiques.

Simulation : Cette interface permet à l'utilisateur de personnaliser la configuration en spécifiant le nombre de tâches et de machines virtuelles souhaité.

Le lancement de la simulation se fait en cliquant sur le bouton «Start Simulation » dans l'interface montrée dans la Figure 3.8. Au cours de la simulation, une approche d'ordonnancement multi-objectifs a été utilisée, se basant sur l'optimisation du makespan, du coût et de la consommation énergétique.

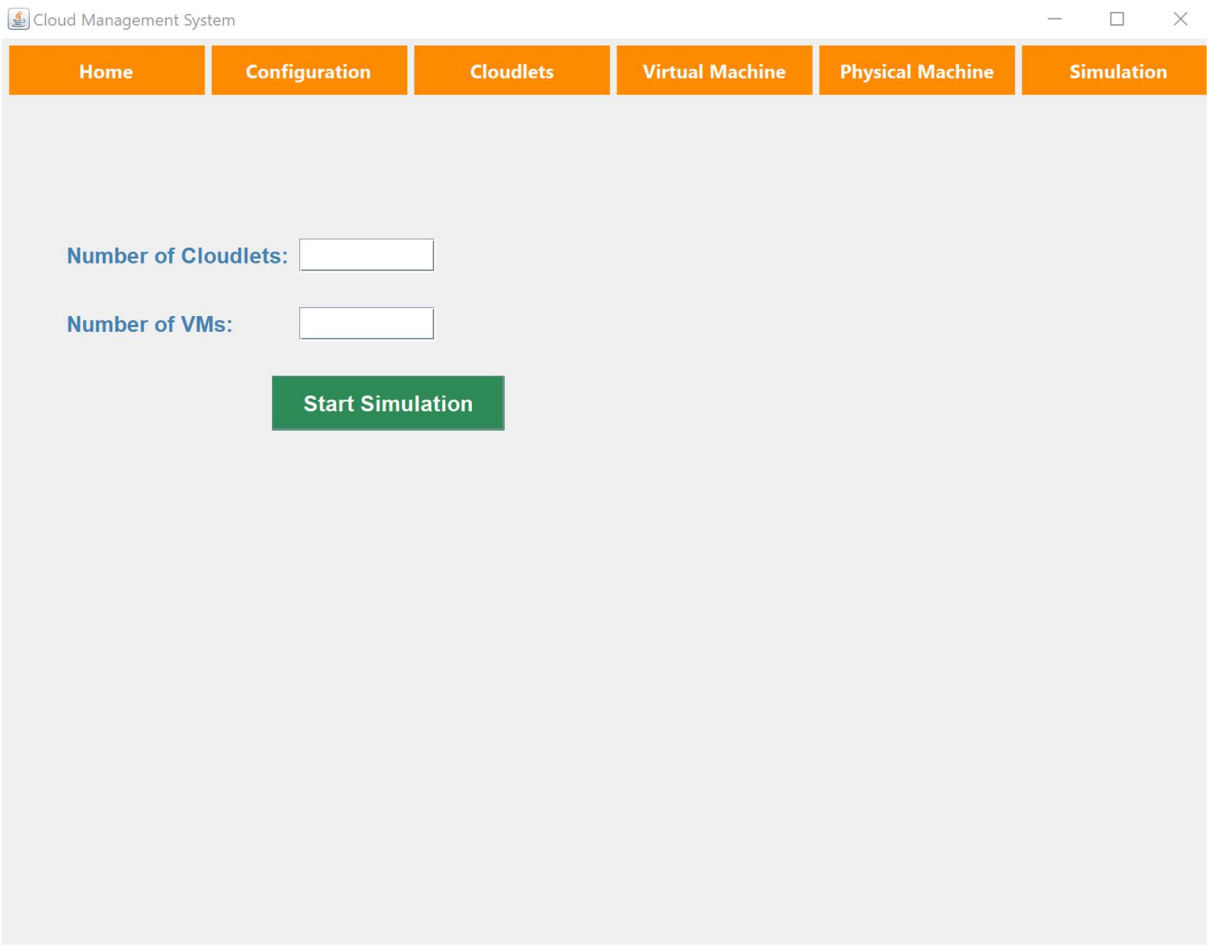


Figure 3.8 : Lancement de simulation.

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 Broker afin de lancer la simulation et afficher les résultats.

3.6 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 SSA pour l'ordonnancement des tâches dans le Cloud computing.

Les simulations ont été effectuées dans un environnement hétérogène, comme précisé dans les tableaux 3.1 et 3.2, qui affichent les valeurs attribuées à chaque paramètre.

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

Tableau 3.1 : Les paramètres de simulation CloudSim [41].

Pour les Cloudlets, nous avons choisis pour une variation de la longueur conformément au tableau 3.2.

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

Tableau 3.2 : Les paramètres de longueur des Cloudlets [41].

Trois processeurs ont été sélectionnés pour cette étude : (Turion MT-34, OMAP et PentiumM).

Le tableau 3.3 montre comment les trois processeurs ajustent leur vitesse, tension et fréquence en fonction des différents niveaux.

La vitesse du processeur diminue également à mesure que la tension et la fréquence se réduisent. Cela permet d'utiliser moins d'énergie. Finalement, une valeur « ec » est assignée à chaque processeur, indiquant la quantité d'énergie qu'il consomme.

VSL	P1: TURION MT-34			P2: OMAP			P3: PENTIUM M		
	Volt (V)	Freq. (Ghz)	R.Speed (%)	Volt. (V)	Freq. (Ghz)	R.Speed (%)	Volt. (V)	Freq. (Ghz)	R.Speed (%)
1	1,2	1,8	100	1,35	0,6	100	1,48	1,4	100
2	1,15	1,6	88,88	1,27	0,55	91,66	1,44	1,2	96,76
3	1,1	1,4	77,77	1,2	0,5	83,33	1,31	1	88,14
4	1,05	1,2	66,66	1	0,25	41,66	1,18	0,8	79,51
5	1	1	55,555	0,9	0,13	20,83	0,96	0,6	64,42
6	0,9	0,8	44,44						
Prix	*ec*= 1,14			*ec*= 0,57			*ec*= 0,76		

Tableau 3.3 : Les paramètres des processeurs [38].

Les machines virtuelles (VMs) sont réparties équitablement en trois groupes afin de gérer l'attribution des VM. D'après le tableau 3.3,un tiers des machines virtuelles utilise le processeur Turion MT-34, un deuxième tiers utilise l'OMAP et le dernier tiers utilise le Pentium M. Un niveau de configuration (allant de 1 à 6 pour Turion MT-34 et de 1 à 5 pour OMAP et Pentium M) est sélectionné aléatoirement pour chaque processeur. D'après le tableau, cette sélection aléatoire détermine précisément les paramètres de tension (V), de fréquence (GHz) et de vitesse relative (%) pour chaque machine virtuelle. Par exemple, le premier groupe pourrait utiliser le niveau 3 du TurionMT-34 (1.1 V, 1.4 GHz, vitesse de 77.77%), tandis qu'un autre groupe identique pourrait opter pour le niveau 5 (1 V, 1 GHz, vitesse de 55.55%). Cette méthode introduit une variabilité contrôlée dans la consommation d'énergie et la performance, tout en assurant une répartition équilibrée des ressources.

3.6.1 Comparaison de résultats en termes de makespan

La figure 3.9 illustre la comparaison de makespan entre SSA et les diverses méthodes examinées pour 25 Vms.

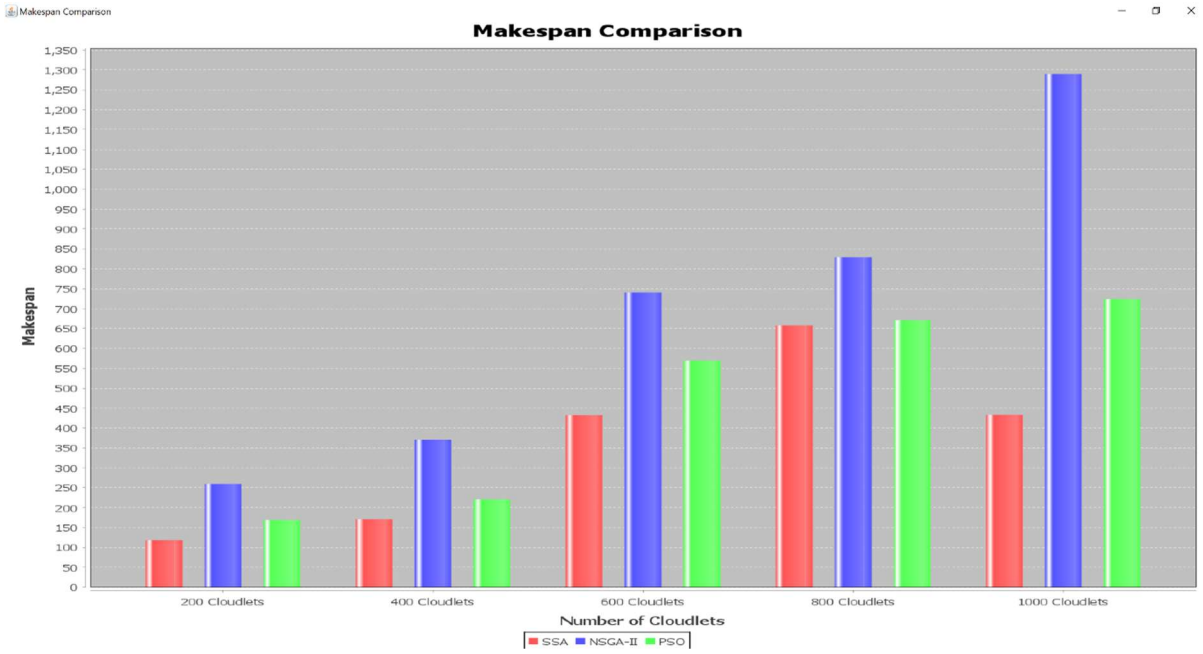


Figure 3.9 : Comparaison en termes de makespan pour 25 Vms.

Le tableau 3.4 montre le détail des valeurs.

Nombre de cloudlets	200	400	600	800	1000
SSA	**117,788**	**170,542**	**432,061**	**657,944**	**432,669**
NSGA-II	259,608	371,061	740,909	829,289	1289,686
PSO	169,008	220,963	569,191	671,028	723,551

Tableau 3.4 : résultats de comparaison en termes de makespan pour 25 Vms.

Résultat : L'examen des données illustrées dans la figure 3.9 et le tableau 3.4 révèle que la technique SSA se distingue clairement par sa capacité à minimiser le makespan, sans tenir compte du nombre de Cloudlets. Avec SSA, l'augmentation du makespan est modérée en comparaison au NSGA-II qui montre des chiffres nettement plus importants. En ce qui concerne la méthode PSO, elle se situe entre les deux, affichant de meilleures performances que le NSGA-II tout en demeurant généralement moins performante que le SSA. En règle générale, la technique SSA démontre une performance stable et efficiente en vue de réduire le temps total d'exécution.

3.6.2 Comparaison de résultats en termes de coût

La figure 3.10 illustre la comparaison de termes de coût entre SSA et les diverses méthodes examinées pour 25 Vms.

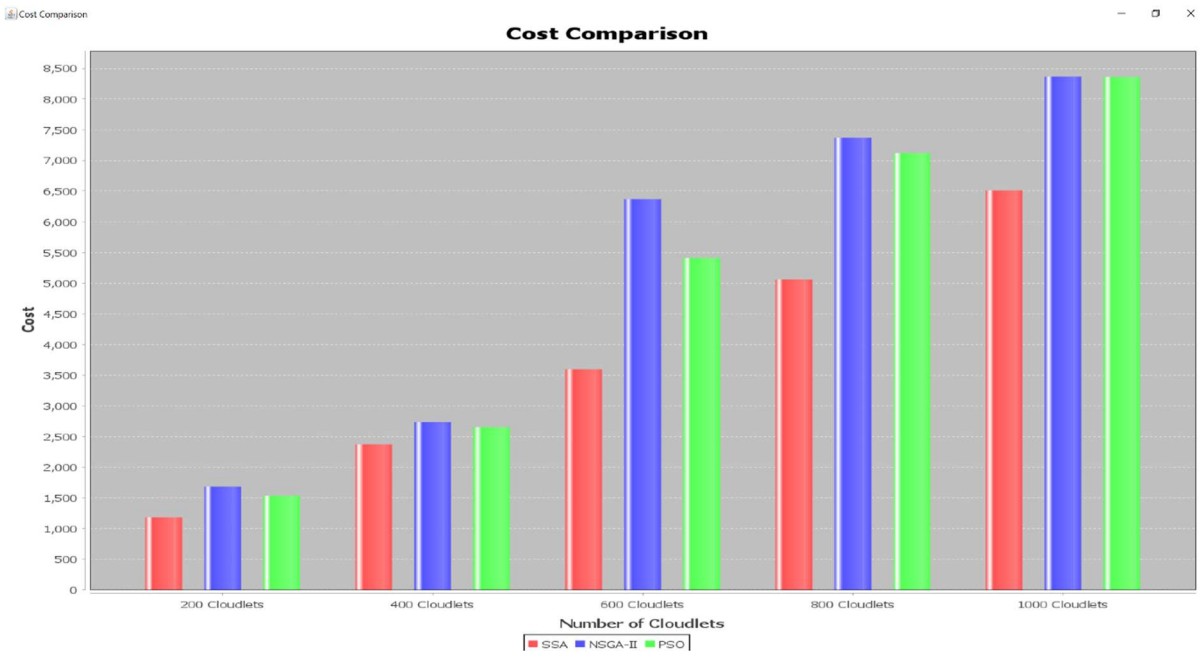


Figure 3.10 : Comparaison en termes de coût pour 25 Vms.

Le tableau 3.5 montre le détail des valeurs.

Nombre de cloudlets	200	400	600	800	1000
SSA	**1181,897**	**2373,421**	**3593,302**	**5060,276**	**6507,881**
NSGA-II	1686,293	2738,621	6371,131	7368,662	8364,449
PSO	1523,658	2656,067	5412,502	7116,563	8358,395

Tableau 3.5 : résultats de comparaison en termes de coût pour 25 Vms.

Résultat : Sur la base des informations présentées dans la figure 3.10 et dans le tableau 3.5, nous notons également que la technique SSA se démarque par ses résultats en matière de diminution des coûts. Pour tous les scénarios examinés, elle produit les dépenses les plus faibles, attestant de sa rentabilité économique. Par contre, les méthodes NSGA-II et PSO montrent des coûts plus importants, leurs valeurs étant très similaires l'une à l'autre. Même si ces deux approches montrent une certaine efficacité dans certaines situations, elles demeurent généralement moins rentables que SSA d'un point de vue financier.

3.6.3 Comparaison de résultats en termes d'énergie

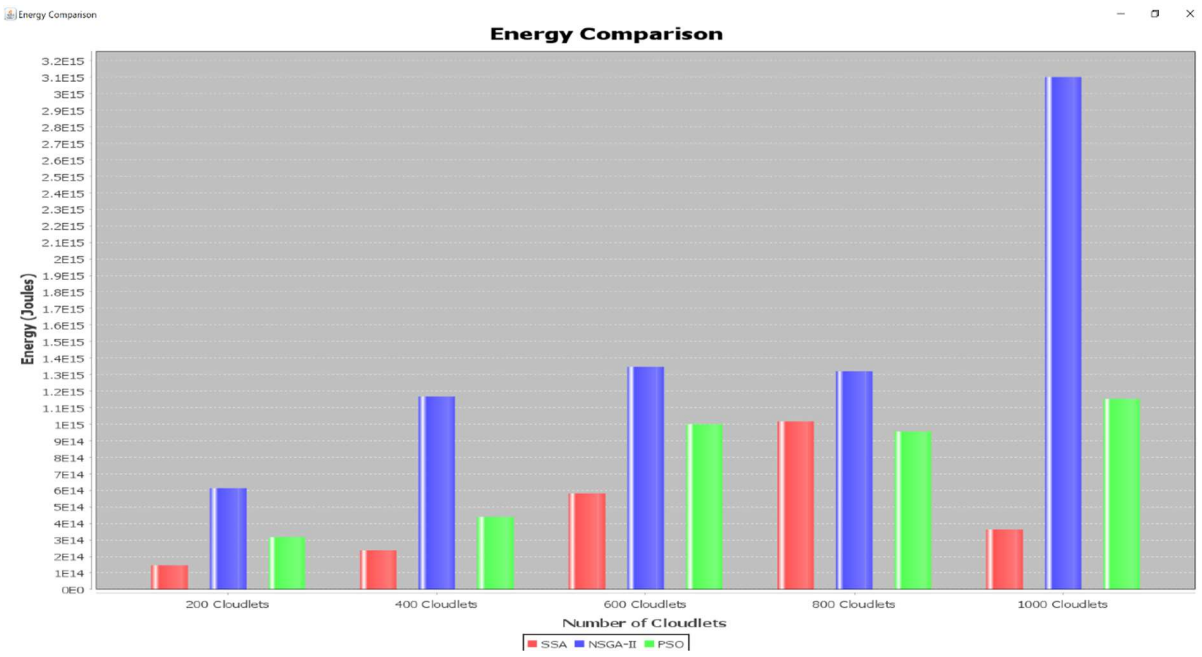


Figure 3. 11 : Comparaison en termes d'énergie pour 25 Vms.

La figure 3.11 illustre la comparaison d'énergie entre SSA et les diverses méthodes examinées pour 25 Vms.

Le tableau 3.6 montre le détail des valeurs en petajoule.

Nombre de cloudlets	200	400	600	800	1000
SSA	**0,1478**	**0,2366**	**0,5834**	1,0187	**0,3632**
NSGA-II	0,6148	1,1672	1,3491	1,3217	3,1023
PSO	0,3191	0,4390	1,0001	**0,9569**	1,1548

Tableau 3.6 : résultats de comparaison en termes d'énergie (PJ) pour 25 Vms.

Résultat : L'analyse de la figure 3.11 et du tableau 3.6 révèle que la technique SSA se distingue généralement par sa meilleure performance en matière de consommation d'énergie. Elle présente les valeurs les plus basses pour pratiquement toutes les situations examinées, en particulier pour 200, 400, 600 et 1000 cloudlets. L'exception à cette règle est le cas des 800 cloudlets, où la méthode PSO affiche une consommation légèrement inférieure à celle de SSA. Cependant, ce bon résultat isolé ne suffit pas à changer la direction générale. La méthode NSGA-II, de son côté, affiche toujours les consommations les plus importantes. Donc, même si une petite fluctuation à 800 cloudlets est observée, la technique SSA conserve généralement sa position en tant que la plus efficace du point de vue énergétique

3.7 Conclusion

Dans ce chapitre, nous avons présenté l'implémentation de notre approche basée sur l'algorithme de recherche d'écureuil (SSA) pour l'ordonnancement des tâches dans le Cloud computing. Nous avons utilisé l'environnement de simulation CloudSim en y intégrant notre algorithme à l'aide du langage Java, de l'IDE, NetBeans et de la bibliothèque JFreeChart pour la visualisation des résultats.

Nous avons développé une interface graphique permettant de configurer et de lancer les simulations selon différents paramètres, facilitant ainsi l'analyse des performances.

Les résultats obtenus à travers plusieurs scénarios ont montré que notre approche basée sur SSA surpasse les métaheuristiques de référence NSGA-II et PSO, notamment en ce qui concerne le makespan, le coût et la consommation d'énergie. Cela confirme l'efficacité de notre contribution dans l'amélioration de la gestion des ressources dans un environnement cloud.

Conclusion générale

Notre étude de recherche s'est penchée sur l'utilisation de la métaheuristique Squirrel Search Algorithm (SSA) pour résoudre un problème complexe d'optimisation de l'ordonnancement des tâches dans des environnements Cloud virtualisés. Confrontée aux défis de la gestion multi-objectifs — notamment la réduction du makespan, des coûts et de la consommation énergétique — notre approche s'est appuyée sur le simulateur CloudSim, reconnu pour sa flexibilité et sa capacité à modéliser de manière réaliste les infrastructures Cloud.

Les résultats expérimentaux, évalués selon les métriques du makespan, du coût et de la consommation énergétique, démontrent la pertinence de SSA dans ce contexte. Une analyse comparative avec deux algorithmes établis, NSGA-II (optimisation multi-objectif génétique) et PSO (Particle Swarm Optimization), révèle une supériorité significative de notre proposition.

Ces conclusions mettent en évidence le potentiel des métaheuristiques bio-inspirées pour relever les défis d'efficacité dans le Cloud computing. Elles ouvrent également des perspectives pour des travaux futurs, notamment en matière d'équilibrage de charge, de consolidation des ressources et d'amélioration de la fiabilité.

Bibliographie

[1] MICHAEL, Armbrust. A view of cloud computing. Magazine Communications of the ACM CACM Homepage archive, 2010, vol. 53, no 4, p. 50-58.

[2] MOHAMMEDI Fatima Zohra. Harris Hawks Optimization Algorithm (HHO) pour l'ordonnancement des tâches dans le Cloud computing. Mémoire de Master en Informatique. Université Abou Bakr Belkaid Tlemcen. 2023-2024.

[3] NADIR, Hamdani, SALIMA, Kerroum, et NACÉRA, Ouallouche. Etude et comparaison des failles de sécurité d'OpenStack et OpenNebula. 2019. Thèse de doctorat. Université Mouloud Mammeri.

[4] Pailhès Bertrand, Heslot Armand. Introduction au could computing. Introduction au Cloud Computing : risques et enjeux pour la vie privée. Disponible sur: https://connect.ed-diamond.com/MISC/misc-060/introduction-au-cloud-computing-risques-et-enjeux-pour-la-vie-privee. Consulté le : 10/4/2025.

[5] RAJKUMAR BUYYA, James Broberg et GOSCINSKI, Andrzej. Cloud computing principles and paradigms. 2019.

[6] EL ALLOUSSI, Hassan, FETJAH, Laila, et SEKKAKI, Abderrahim. L'état de l'art de la sécurité dans le Cloud Computing. INTIS 2012, 2012, p. 3.

[7] YANG, Xin-She, TALATAHARI, Siamak, et ALAVI, Amir Hossein (ed.). Metaheuristics in water, geotechnical and transport engineering. Newnes, 2012.

[8] Talbi, E. G. (2009). Metaheuristics: from design to implementation. John Wiley & Sons. TALBI, El-Ghazali. Metaheuristics: from design to implementation. John Wiley & Sons, 2009.

[9] Cloud Computing: Qu'est-ce qui vous retient? Disponible sur : https://www.tresfacile.net/cloud-computing-quest-ce-qui-vous-retient/

Consulté le : 4/3/2025.

[10] Maintenance performed by a trusted provider Disponible sur : https://www.netvault.net.au/cloud/public-cloud/ . Consulté le : 4/3/2025.

[11] Private Cloud: What is it and How Does it Work? Disponible sur : https://botpenguin.com/glossary/private-cloud . Consulté le : 5/3/2025

[12] Quelles différences entre Cloud Hybride et Multi-cloud ? Disponible sur : https://carrieres.groupeozitem.com/blog/quelles-differences-entre-cloud-hybride-et-multi-cloud . Consulté le : 5/3/2025.

[13] Modèles de service cloud principaux : SaaS, PaaS et IaaS Disponible sur : https://www.stackscale.fr/blog/modeles-service-cloud/ Consulté le : 5/3/2025

[14] Software as a Service (SaaS) Disponible sur : https://www.ringcentral.com/gb/en/blog/definitions/software-as-a-service-saas/ Consulté le : 6/3/2025.

[15] Key Factors to Consider While Selecting the Right PaaS Provider Disponible sur : https://63sats.com/blog/platform-as-a-service-in-cloud-computing/ Consulté le : 6/3/2025.

[16] What is IaaS in Cloud Computing? Disponible sur : https://intellipaat.com/blog/iaas-in-cloud-computing/ Consulté le : 6/3/2025.

[17] Redjem, Rabeh. (2013). Decision support for planning the operations and the human resources in Home Health Care services. Rapport de recherche. Université Jean Monnet - Saint-Etienne, 2013. France.

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

[19] KOKASH, Natallia. An introduction to heuristic algorithms. Department of Informatics and Telecommunications, 2005, vol. 1, p. 1-7.

[20] GENDREAU, Michel, POTVIN, Jean-Yves, et al. (ed.). Handbook of metaheuristics. New York : Springer, 2010.

[21] BEHESHTI, Zahra et SHAMSUDDIN, Siti Mariyam Hj. A review of population-based meta-heuristic algorithms. Int. j. adv. soft comput. appl, 2013, vol. 5, no 1, p. 1-35.

[22] ALMUFTI, Saman M., SHABAN, A. Ahmad, ALI, Z. Arif, et al. Overview of metaheuristic algorithms. Polaris Global Journal of Scholarly Research and Trends, 2023, vol. 2, no 2, p. 10-32.

[23] HOLLAND, John H. Adaptation in natural and artificial systems. University of Michigan Press google schola, 1975, vol. 2, p. 29-41.

[24] KENNEDY, James et EBERHART, Russell. Particle swarm optimization. In : Proceedings of ICNN'95-international conference on neural networks. ieee, 1995. p. 1942-1948.

[25] Joe Millitzer. Night Flyers: The secret life of Missouri's flying squirrels. Disponible sur https://www.yahoo.com/lifestyle/night-flyers-secret-life-missouri-210738776.html. Consulté le : 5/5/2025.

[26] DHAINI, Mahdi et MANSOUR, Nashat. Squirrel search algorithm for portfolio optimization. Expert Systems with Applications, 2021, vol. 178, p. 114968.

[27] ZHANG, Xuncai, ZHAO, Kai, WANG, Lingfei, et al. An improved squirrel search algorithm with reproductive behavior. IEEE Access, 2020, vol. 8, p. 101118-101132.

[28] HU, Hongping, ZHANG, Linmei, BAI, Yanping, et al. A hybrid algorithm based on squirrel search algorithm and invasive weed optimization for optimization. IEEE Access, 2019, vol. 7, p. 105652-105668.

[29] Behavior of Squirrel Search Algorithm (SSA) using Meta-heuristics method. Disponible sur: https://transpireonline.blog/2019/09/18/behavior-of-squirrel-search-algorithm-ssa-using-meta-heuristics-method/. Consulté le : 9/5/2025.

[30] WANG, Yanjiao et DU, Tianlin. A multi-objective improved squirrel search algorithm based on decomposition with external population and adaptive weight vectors adjustment. Physica A: Statistical Mechanics and its Applications, 2020, vol. 542, p. 123526.

[31] SINGH, Nagendra, GUPTA, Krishna Kumar, JAIN, Sanjay K., et al. A flying squirrel search optimization for MPPT under partial shaded photovoltaic system. IEEE Journal of Emerging and Selected Topics in Power Electronics, 2020, vol. 9, no 4, p. 4963-4978.

[32] FARRELL, Joyce. Java™ Programming. Cengage Learning, Inc., 2019.

[33] WIELENGA, Geertjan. Beginning netbeans ide: For java developers. Apress, 2015.

[34] GILBERT, David. The jfreechart class library. Developer Guide. Object Refinery, 2002, vol. 7.

[35] GOYAL, Tarun, SINGH, Ajit, et AGRAWAL, Aakanksha. Cloudsim: simulator for cloud computing infrastructure and modeling. Procedia Engineering, 2012, vol. 38, p. 3566-3572.

[36] BENADDA, Meriem. Le Cloud Computing pour une Gestion Intelligente du Trafic Urbain: Proposition d'un service de prise en charge et de coopération pour la gestion de personnes victimes de malaise pendant la conduite. 2019. Thèse de doctorat. Université Oran 1-Ahmed Ben Bella.

[37] DURILLO, Juan J., NAE, Vlad, et PRODAN, Radu. Multi-objective workflow scheduling: An analysis of the energy efficiency and makespan tradeoff. In : 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. IEEE, 2013. p. 203-210.

[38] YASSA, Sonia. Allocation optimale multicontraintes des workflows aux ressources d'un environnement cloud computing. 2014. Thèse de doctorat. Université de Cergy Pontoise.

[39] MARLER, R. Timothy et ARORA, Jasbir S. The weighted sum method for multi-objective optimization: new insights. Structural and multidisciplinary optimization, 2010, vol. 41, p. 853-862.

[40] YANG, Xin-She. Nature-inspired optimization algorithms. Academic Press, 2020.

[41] ALSADIE, Deafallah. TSMGWO: Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers. IEEE Access, 2021, vol. 9, p. 37707-37725.

Résumé

Le Cloud computing est une avancée technologique qui offre un accès facile et sûr aux informations et services stockés sur des serveurs éloignés, assurant ainsi un service de haute qualité et une disponibilité continue. Cette étude porte sur l'optimisation multi-objectifs dans le Cloud computing en utilisant l'algorithme de recherche des écureuils, avec pour objectif de réduire le makespan, le coût et la consommation d'énergie. La problématique abordée concerne l'ordonnancement des tâches sur les différentes machines virtuelles. Notre méthode a démontré de meilleures performances en comparaison avec les deux méthodes approchées (NSGA-II et PSO). Nous avons utilisé CloudSim comme simulateur et nous avons créé l'interface utilisateur en Java afin de rendre les interactions avec le simulateur plus faciles.

Mots clés : Cloud computing, CloudSim, SSA, NSGA-II, PSO.

Abstract

Cloud computing is a technological advancement that provides easy and secure access to information and services stored on remote servers, thereby ensuring high-quality service and continuous availability. This study focuses on multi-objective optimization in Cloud computing using the Squirrel Search Algorithm, aiming to minimize makespan, cost, and energy consumption. The addressed issue involves task scheduling across various virtual machines. Our approach demonstrated better performance compared to two existing methods (NSGA-II and PSO). We used CloudSim as the simulator and developed the user interface in Java to facilitate interactions with the simulator.

Mots clés : Cloud computing, CloudSim, SSA, NSGA-II, PSO.

ملخص

الحوسبة السحابية هي تقدم تكنولوجي يوفر وصولًا سهلاً وآمنًا إلى المعلومات والخدمات المخزنة على خوادم بعيدة، مما يضمن جودة خدمة عالية وتوفراً مستمراً. تركز هذه الدراسة على تحسين الأهداف المتعددة في بيئة الحوسبة السحابية من خلال استخدام خوارزمية البحث لدى السناجب (SSA) بهدف تقليل زمن التنفيذ، التكاليف، واستهلاك الطاقة. وتتمحور المسألة المدروسة حول جدولة المهام على مختلف الآلات الافتراضية. لقد أظهرت طريقتنا أداءً أفضل مقارنةً بالطريقتين المقاربتين (NSGA-II و PSO) استخدمنا CloudSim كمحاكي، وقمنا بتطوير واجهة المستخدم بلغة Java لتسهيل التفاعل مع المحاكي.

الكلمات الرئيسية : الحوسبة السحاب, CloudSim, SSA, NSGA-II, PSO