REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU-BEKR BELKAID – TLEMCEN

THÈSE LMD

Présentée à :

FACULTE DES SCIENCES – DEPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

DOCTORAT

Spécialité : *Informatique distribuée et réseaux (IDR)*

Par :

Mr MALTI Arslan Nedhir

Sur le thème

Algorithmes d'ordonnancement multi-objectif dans les systèmes distribués

---

Soutenue publiquement le 02/03/2024 à Tlemcen devant le jury composé de :

Mr BENAMAR Abdelkrim	Professeur	Université de Tlemcen	Président
Mr BENMAMMAR Badr	Professeur	Université de Tlemcen	Directeur de thèse
Mr HAKEM Mourad	Maître de Conférences	Université de Franche-Comté	Co-Directeur de thèse
Mr BENDAOUD Fayssal	Maître de Conférences A	ESI de Sidi Bel-Abbes	Examinateur
Mr MERAD BOUDIA Omar Rafik	Maître de Conférences A	Université d'Oran 1	Examinateur
Mme AMRAOUI Asma	Maître de Conférences A	Université de Tlemcen	Examinatrice

Laboratoire de Télécommunication de Tlemcen (LTT)
BP 119, 13000 Tlemcen - Algérie

À mes chers parents, qui ont toujours été là pour moi, et qui m'ont donné un magnifique modèle de labeur et de persévérance,

À mon frère et à ma sœur pour leur soutien et leurs encouragements,

À ma grand-mère et à tous mes proches,

Je dédie ce travail

- Arslan

Remerciements

Je tiens avant tout à exprimer ma gratitude envers ALLAH le tout puissant pour m'avoir doté du courage, de la patience, de la force et de la détermination nécessaire pour mener à bien ce travail.

J'exprime mes sincères appréciations et ma profonde reconnaissance envers mon directeur de thèse, **Pr. Badr BENMAMMAR**, et mon co-directeur, **Dr. Mourad HAKEM**, pour leur esprit scientifique, leur pédagogie, leurs encouragements, leur soutien et leurs précieuses directives continues. Leurs orientations éclairées ont façonné mon parcours doctoral de manière à en faire une expérience à la fois stimulante et gratifiante. Leurs mains secourables ont toujours été là lorsque j'en avais besoin. Les discussions fructueuses que nous avons eues, leur capacité d'analyse et leur enthousiasme m'ont montré que le monde de la recherche pouvait être un univers passionnant. Je leur suis infiniment reconnaissant pour tous les efforts déployés en ma faveur.

Je remercie également les membres du laboratoire LTT pour leur soutien. Ma reconnaissance s'étend également aux membres du comité : **Pr. Abdelkrim BENAMAR** en tant que président du comité, **Dr. Fayssal BENDAOUD** de l'ESI de Sidi Bel-Abbes, **Dr. Omar Rafik MERAD BOUDIA** de l'université d'Oran 1 et **Dr. Asma AMRAOUI** de l'université de Tlemcen en tant qu'examinateurs, qui ont généreusement accepté d'examiner et d'évaluer mon travail. De même, je remercie **Dr. Houcine MATALLAH,** chef du département d'informatique, pour sa disponibilité, sa compréhension et son encouragement. Mes remerciements vont également aux évaluateurs anonymes pour leurs commentaires précieux, leur objectivité et leurs efforts.

Sur un plan plus personnel, je remercie chaleureusement mes parents, mon frère et ma sœur, dont la patience et les encouragements ont été une source d'inspiration essentielle pour la réalisation de cette recherche doctorale.

Je ne saurais oublier de remercier les nombreux enseignants et amis qui m'ont soutenu avec une foi inébranlable, apportant ainsi une énergie et un enthousiasme indispensables à la concrétisation de ce travail. Je leur souhaite à tous un épanouissement continu dans leurs travaux.

Résumé

De nos jours, le Cloud Computing est largement adopté dans divers domaines et son expansion s'accompagne des défis et des contraintes dont les organisations doivent prendre en compte pour en tirer pleinement parti. L'ordonnancement des tâches émerge comme un enjeu clé pour améliorer les performances. Malheureusement, il est bien connu que ce problème est NP-difficile, nécessitant des approches judicieuses. La plupart des recherches antérieures se concentrent sur un seul objectif, et celles qui considèrent plusieurs objectifs utilisent souvent des fonctions de compromis simples, négligeant l'influence mutuelle des paramètres. De plus, le phénomène de stagnation des solutions locales et la convergence prématurée entravent le processus de recherche. Pour surmonter ces limites, cette thèse propose de nouveaux algorithmes d'optimisation basés sur des approches bio-inspirées, notamment le comportement de pollinisation des fleurs, la capacité d'exploration de l'optimiseur du loup gris, les opérateurs de croisement des algorithmes évolutionnaires, et la stratégie employée dans l'algorithme de saut de grenouille. Ces approches sont appliquées à l'ordonnancement des tâches dans les systèmes de Cloud hétérogènes, formulé sous forme d'un problème d'optimisation multi-objectif. Les résultats obtenus à travers une série de tests et de simulations, avec des charges de travail synthétiques et standard, démontrent la supériorité de nos algorithmes en termes de qualité des solutions et de vitesse de convergence par rapport à d'autres techniques récemment proposées dans la littérature.

Mots clés : *algorithmes distribués*, *cloud computing*, *optimisation multi-objectif*, *ordonnancement des tâches*, *méta-heuristiques*, *optimisation combinatoire*.

Abstract

Nowadays, Cloud Computing is widely adopted in various domains and its expansion is accompanied by challenges and constraints that organizations must to be aware of and address to fully harness its power. Task scheduling has emerged as a key concern for enhancing performances. Unfortunately, it is well-known that this problem is NP-hard, requiring judicious approaches. Most previous research focuses on a single objective, and those dealing multiple objectives often use a simple compromise function and do not consider how each of the parameters might influence the others. Additionally, the stagnation phenomenon of local solutions and the premature convergence hinder the research process. To overcome these limitations, this thesis proposes new optimization algorithms based on bio-inspired approaches, including the flower pollination behavior, the exploration capability of the gray wolf optimizer, evolutionary algorithms crossover operators, and the strategy employed in the frog algorithm. These approaches are applied to task scheduling in heterogeneous Cloud systems, formulated as a multi-objective optimization problem. The results obtained through a series of tests and simulations, with synthetic and standard workloads, demonstrate the superiority of our algorithms in terms of solution quality and convergence speed compared to other techniques recently proposed in the literature.

Keywords : *distributed algorithms*, *cloud computing*, *multi-objective optimization*, *task scheduling*, *metaheuristics*, *combinatorial optimization*.

ملخص

في الوقت الحاضر، يتم اعتماد الحوسبة السحابية على نطاق واسع في مجالات مختلفة، ويصاحب توسعها تحديات وقيود يجب على المؤسسات أن تكون على دراية بها ومعالجتها لتسخير قوتها بشكل كامل. لقد برزت جدولة المهام كمشكلة رئيسية لتحسين أداء أنظمة الحوسبة السحابية. لسوء الحظ، فمن المعروف جيدًا أن هذه المشكلة صعبة، وحلها يتطلب أساليب حكيمة. تركز معظم الأبحاث السابقة على هدف واحد، وغالبًا ما يستخدم أولئك الذين يتعاملون مع أهداف متعددة وظيفة تسوية بسيطة دون مراعاة التأثير المتبادل للمعلمات. إضافة إلى ذلك، فإن ظاهرة ركود الحلول المحلية والتقارب المبكر يعيقان عملية البحث. للتغلب على هذه القيود، تقترح هذه الأطروحة خوارزميات تحسين جديدة تعتمد على أساليب مستوحاة من العوامل البيولوجية، بما في ذلك سلوك تلقيح الزهور، وقدرة الاستكشاف لمُحسِنْ الذئب الرمادي، وعمليات التهجين في الخوارزميات التطورية، والاستراتيجية المستخدمة في خوارزمية الضفدع. يتم تطبيق هذه الأساليب على جدولة المهام في الأنظمة السحابية غير المتجانسة، والتي تمت صياغتها كمشكلة تحسين متعددة الأهداف. تُظهر النتائج التي تم الحصول عليها من خلال سلسلة من الاختبارات والمحاكاة، مع أعباء العمل التركيبية والقياسية، تفوق خوارزمياتنا من حيث جودة الحلول وسرعة التقارب مقارنة بالتقنيات المقترحة مؤخرًا في الأدبيات.

الكلمات المفتاحية : الخوارزميات الموزعة ، حوسبة سحابية ، التحسين متعدد الأهداف ، جدولة المهام ، الاستدلال الفوقية ، تحسين تركيبي .

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des acronymes

ABC	Artificial Bee Algorithm
ACO	Ant COlony Algorithm
AEF	Automated Emulation Framework
AG	Algorithmes Génétiques
ALO	AntLion optimization
API	Application Programming Interface
AWS	Amazon Web Services
BA	Bat Algorithm
BFA	Bacteria Foraging Algorithm
BFO	Bacterial Foraging Optimization
BH	Black Hole optimization
BHEFT	Budget Heterogeneous Earliest Finish Time
BSO	Brain Storm Optimization
CPU	Central Processing Unit
CRM	Customer Relationship Management
CRO	Chemical Reaction Optimization
CS	Cuckoo Search
CSO	Cat Swarm Optimization
CSPs	Cloud Service Providers
CWSA	Cloud-based Workflow Scheduling Algorithm
DAG	Graphes Acycliques Dirigés
DCs	Data Centers
DE	Differential Evolution
DVMC	Dynamic Virtual Machine Consolidation
EC2	Elastic Compute Cloud
ECS	Energy-Conscious Scheduling
EEFPA	Exploration-Enhanced Flower Pollination Algorithm
ELHHO	Elite Learning Harris Hawks Optimizer
EOBL	Elite Opposition-Based Learning
ES	Evolution Strategies
FA	Firefly Algorithm
FCFS	First Come First Serve
FIFO	First-in-First-out
FOA	Forest Optimization Algorithms
FPA	Flower Pollination Algorithm
GBO	Gradient-Based Optimizer
GBSA	Galaxy-Based Search Algorithm
GBSO	Global Best Brain Storm Optimization
GCE	Google Compute Engine
GGWO	Genetic Gray Wolves Optimization
GOA	Grasshopper Optimization Algorithm

GoCJ	Google Cloud Jobs
GPU	Graphical Processing Unit
GSA	Gravitational Search Algorithm
GUI	Graphical User Interface
GWO	Gray Wolf Optimization
HDEA	Hybrid Differential Evolution Algorithm
HEFT	Heterogeneous Earliest Finish Time
HHO	Harris Hawks Optimization
HPC2N	High-Performance Computing Center North
HS	Harmony search
HSJF	Heterogeneous Shortest Job First
IA	Intelligence Artificielle
IaaS	Infrastructure as a Service
ICA	Imperialist Colony Algorithm
ICA	Imperialist Competitive Algorithm
IoT	Internet Of Things
KWH	kilowattheure
LAN	Local Area Network
LBMM	Load-Balanced Min-Max
LDIW	Linearly Decreasing Inertia Weight
LPGWO	Local Pollination-based Gray Wolf Optimizer
LPMSA	Local Pollination based Moth Search Algorithm
MA	Memetic Algorithm
MCT	Minimum Completion Time
MHBFA	Multi-Hybrid Bacteria Foraging Algorithm
MI	Million Instructions
MIPS	Millions d'Instructions Par Seconde
MOCSO	Multi-Objective Cat Swarm Optimization
MOGAS	Many-Objective Genetic Algorithm Scheduler
MR	Maximum Resource
MSA	Moth Search Algorithm
MD	Matrice de Décision
MT	Modified Throttle
MVO	MultiVerse optimizer
NIST	National Institute of Standards and Technology
NMD	Normalisation de la Matrice de Décision
NSGA-III	Non-dominated Sorting Genetic Algorithm III
OBL	opposition-Based Learning
OVF	Open Virtualization Format
P2P	Peer-to-Peer
PaaS	Platform as a Service
PEFT	Predict Earliest Finish Time
PHs	Physical Hosts
PR	Proximité Relative
PSO	Particle Swarm Optimization
QoS	Quality of Service
RASA	Resource Aware Scheduling Algorithm
REST	Representational State Transfer
RIS	Resource Information Server

RR	Round Robin
RUR	Taux d'utilisation des ressources
S3	Simple Storage Service
SaaS	Software as a Service
SA	Simulated Annealing
SDE	Shift-Based Density Estimation
SFLA	Shuffled Frog-Leaping Algorithm
SIP	Solution Idéale Positive
SIN	Solution Idéale Négative
SJF	Shortest Job First
SLA	Service-Level Agreement
SLC	Soccer League Competition algorithm
SNIC	Swedish National Computing Infrastructure
SSA	Salp Swarm Algorithm
T-BDMWS	TOPSIS inspired Budget and Deadline aware Multi Workflow Scheduling
TASA	Task Aware Scheduling Heuristic
TI	Technologies de l'Information
TOPSIS	Technique for Order Performance by Similarity to Ideal Solution
TPU	Tensor Processing Unit
TSMGWO	Task Schedule using MultiObjectives Grey Wolf Optimizer
UAV	Unmanned Aerial Vehicle
VMM	Virtual Machine Manager
VMs	Virtual Machines
VNS	Variable Neighbourhood Search
VPN	Virtual Private Network
WAN	Wide Area Network
WOA	Whale Optimization Algorithm
WSM	Weighted Sum Model

Introduction générale

Contexte et motivation

Le Cloud Computing, une révolution technologique, a connu une expansion fulgurante, touchant aussi bien les domaines commerciaux que scientifiques. Il se fonde sur un modèle de prestation de services en Technologies de l'Information (TI) qui permet l'accès aux ressources par le biais d'outils et d'applications basés sur le web, accessibles via Internet. Ce modèle offre une flexibilité inégalée en fournissant un accès à la demande à des pools partagés de ressources système configurables, tels que les réseaux, les serveurs, le stockage et les applications.

Il y a quelques décennies, le Cloud Computing était une technologie peu répandue, mais elle est désormais utilisée dans tous les secteurs. Les particuliers l'utilisent pour gérer leurs tâches quotidiennes, tandis que les petites entreprises, qui ne disposent pas des moyens nécessaires pour créer leur propre centre de données, adoptent le Cloud en optant pour le modèle de service le mieux adapté à leurs besoins spécifiques. Cette démocratisation du Cloud repose en grande partie sur les avantages considérables qu'il offre aux fournisseurs et aux clients. En effet, les services Cloud permettent un déploiement rapide des applications pour les clients, avec une tarification basée sur l'utilisation réelle (pay-as-you-go). Pour les prestataires de services Cloud (CSPs), cette technologie est devenue une source de revenus majeure. Selon [3], le marché mondial du Cloud Computing était estimé à 545,8 milliards de dollars en 2022, avec une prévision de croissance impressionnante à un taux annuel composé de 17,9 %, pour atteindre 1240,9 milliards de dollars d'ici 2027.

Avec cette croissance impressionnante du Cloud, suivie inévitablement par des besoins des clients en constante évolution, de nouveaux défis émergent, susceptibles de compromettre son bon fonctionnement. Parmi ces défis, la sécurité des données, l'équilibrage des charges de travail, l'ordonnancement des ressources et la gestion de la consommation d'énergie sont d'une importance cruciale. Ces difficultés ont le potentiel d'influencer de manière significative l'efficacité et la durabilité de cette technologie, remettant en question sa viabilité à long terme.

Dans le cadre de cette thèse, nous abordons l'un de ces défis majeurs, à savoir l'ordonnancement des tâches. Il s'agit d'un élément essentiel pour garantir l'utilisation efficiente des ressources informatiques dans le Cloud [4]. L'ordonnancement des tâches implique l'orchestration judicieuse des demandes émises par les utilisateurs, exprimées sous forme de tâches (dépendantes ou indépendantes), à l'ensemble des ressources en prenant en compte des contraintes spécifiques dans l'environnement IaaS. L'objectif principal est d'optimiser les performances globales du système tout en respectant ces contraintes. Ainsi, la performance globale et l'utilisation ef-

ficace des ressources dépendent étroitement de la résolution adéquate de ce défi.

Contributions

Le Cloud, grâce à la virtualisation, crée l'illusion de ressources pratiquement illimitées, avec diverses configurations. Toutefois, allouer ou libérer ces ressources en fonction des besoins des applications requiert une attention particulière. Malheureusement, ce problème est connu pour être NP-complet, en raison de la complexité de l'espace des solutions possibles, ce qui implique un temps considérable pour trouver une solution optimale [5]. De ce fait, aucun algorithme de planification en temps polynomial n'est applicable pour optimiser l'ordonnancement des ressources de façon optimale dans les environnements Cloud.

Pour relever ce défi, de nombreuses techniques heuristiques et méta-heuristiques ont été développées, chacune avec ses propres objectifs spécifiques. Au départ, de nombreuses heuristiques ont été proposées, mais la plupart dépendent largement des règles de base pour améliorer la qualité des solutions. En revanche, les techniques méta-heuristiques se sont avérées plus efficaces pour résoudre une variété de problèmes d'optimisation complexes, y compris les problèmes d'ordonnancement, en fournissant des solutions quasi-optimales en un temps raisonnable. Néanmoins, elles présentent certaines limites, telles que la convergence prématurée et des phases de processus d'optimisation qui peuvent stagner dans des optima locaux [6]. Afin de surmonter ces limitations, une approche prometteuse consiste à combiner une ou plusieurs heuristiques et méta-heuristiques, exploitant ainsi leurs forces tout en minimisant leurs faiblesses.

De plus, diverses variantes de ce problème d'optimisation ont fait l'objet d'études, notamment l'optimisation multi-objectif, qui implique la prise en compte simultanée de différents critères antagonistes. Cependant, la plupart de ces travaux se sont focalisés sur un ou deux objectifs d'optimisation, mettant principalement l'accent sur le temps d'achèvement et le coût [7]. Même lorsque certaines études intègrent d'autres paramètres de qualité de service, elles adoptent souvent une simple fonction de compromis entre ces objectifs, sans considérer l'influence mutuelle des paramètres.

Afin de combler ces lacunes, notre thèse a pour objectif de développer et d'évaluer de nouveaux algorithmes d'optimisation multi-objectifs pour l'ordonnancement de charges de travail dans des environnements hétérogènes au niveau IaaS de l'infrastructure du Cloud Computing. Notre approche s'appuie sur une catégorie d'algorithmes métaheuristiques inspirés des mécanismes d'adaptation observés chez les plantes à fleurs et de l'intelligence en essaim pour résoudre le défi de l'ordonnancement des tâches dans les environnements Cloud.

Notre *première contribution* dans cette thèse consiste en un algorithme d'ordonnancement qui traite à la fois les variantes d'optimisation mono-objectif et multi-objectif. Cet algorithme s'inspire du comportement de pollinisation des fleurs et utilise des méthodes d'évaluation multicritères basées sur le principe d'optimalité de Pareto et la technique TOPSIS pour améliorer la qualité des solutions obtenues. L'objectif principal est d'abou-

tir à un ordonnancement le plus satisfaisant en réponse aux besoins et aux préférences des utilisateurs Cloud.

La combinaison de diverses heuristiques ou métaheuristiques pour exploiter leurs points forts et compenser leurs limites représente une avenue de recherche prometteuse. Dans cette perspective, notre *deuxième contribution* consiste en un nouvel algorithme d'optimisation hybride qui combine le comportement de pollinisation des fleurs, les principes d'optimisation issus de l'intelligence des loups gris, et les opérateurs de croisement génétique. L'objectif est de tirer parti des atouts de différentes approches pour améliorer les performances globales du processus d'ordonnancement des tâches, tout en cherchant un compromis entre les avantages du fournisseur de service et les exigences de qualité de service de l'utilisateur.

Dans notre *troisième contribution*, nous améliorons de manière significative notre précédente proposition en intégrant la stratégie du "memeplex", alignée sur le modèle utilisé dans l'algorithme de saut de grenouille. Ces ajustements visent à intensifier l'exploration de l'espace des solutions, offrant ainsi une résolution plus efficace des problèmes d'ordonnancement à optimisation multi-objectifs.

Organisation du manuscrit

Le manuscrit de cette thèse est divisé en cinq chapitres, organisés comme suit :

— Le premier chapitre établit les bases nécessaires à la compréhension générale de notre travail. Nous y présentons le contexte du Cloud Computing ainsi que les concepts fondamentaux. De plus, nous explorons divers défis de recherche, parmi lesquels l'ordonnancement des tâches dans le Cloud occupe une place centrale. Cet enjeu particulier retient notre attention et fait l'objet de cette thèse.

— Le deuxième chapitre définit la problématique que nous abordons, jetant ainsi les fondements pour une meilleure compréhension du problème que nous cherchons à résoudre. Notre objectif est d'optimiser l'affectation des tâches aux ressources du Cloud tout en équilibrant les avantages du fournisseur de services et les exigences en matière de qualité de service de l'utilisateur.

— Le troisième chapitre, pour mieux cerner notre problématique, examine les aspects liés à l'optimisation et explore les diverses approches de résolution existantes dans la littérature. Nous présentons également un état de l'art des algorithmes d'ordonnancement de charges de travail dans le Cloud Computing. Cette analyse nous permet d'identifier les lacunes et les limitations des approches existantes, tout en mettant en lumière les exigences essentielles à considérer pour le développement de nouvelles approches.

— Le quatrième chapitre introduit notre première contribution, qui réside dans l'adaptation d'un algorithme pour l'ordonnancement des tâches dans les environnements Cloud IaaS hétérogènes. Cette adaptation tient compte à la fois des objectifs uniques et multiples. Notre approche s'inspire du comportement bio-inspiré de la pollinisation des

fleurs. L'algorithme repose sur une évaluation multicritère de la fonction de fitness, en combinant le principe de l'optimalité de Pareto avec la technique multicritères TOPSIS. Notre objectif principal est de sélectionner de manière judicieuse la solution la plus pertinente en réponse aux besoins des utilisateurs, tout en tenant compte des compromis nécessaires entre un ensemble d'objectifs contradictoires.

— Le cinquième chapitre comporte deux contributions. La première contribution présente une nouvelle stratégie hybride qui combine le comportement bio-inspiré de la pollinisation des fleurs avec l'optimisation du loup gris, dans le but d'améliorer de manière significative les performances globales du processus d'ordonnancement des tâches dans les environnements hétérogènes du Cloud Computing. Nous utilisons des opérateurs de croisement issus des algorithmes génétiques pour maintenir la diversité des solutions de la population tout au long du processus d'optimisation. La deuxième contribution réutilise la même approche métaheuristique hybride, tout en apportant des mises à jour substantielles. Nous incorporons en plus les caractéristiques clés de l'algorithme de grenouille pour apporter des améliorations et surmonter les faiblesses de l'approche précédemment proposée.

Nous concluons ce manuscrit par une synthèse générale récapitulant l'ensemble des contributions présentées tout au long de cette thèse. De plus, nous jetons un regard vers l'avenir en évoquant des pistes de recherche futures prometteuses.

Concepts fondamentaux des systèmes distribués

1

Sommaire

1.1 Introduction

Dans l'ère actuelle de notre monde interconnecté et numérique, les systèmes distribués jouent un rôle fondamental dans la technologie contemporaine. Face à l'évolution rapide des besoins en traitement de données et en ressources informatiques, les systèmes centralisés traditionnels ont révélé leurs limites en termes de scalabilité et de disponibilité. Ainsi, une nouvelle ère a vu le jour, caractérisée par l'émergence des systèmes distribués, qui se positionnent comme une solution ingénieuse pour une gestion efficace des ressources informatiques.

Les systèmes distribués forment la base de nombreuses technologies qui animent notre vie quotidienne. Leur rôle crucial réside dans la capacité à permettre la collaboration cohérente entre des entités autonomes composées de matériel et de logiciel. Grâce aux échanges de messages et à une coordination synchronisée via des réseaux de communication, ces éléments se concertent pour accomplir des tâches spécifiques et fournir des services de manière efficace. Exploiter pleinement le potentiel des systèmes distribués ouvre la voie à des applications plus performantes, flexibles et résilientes, capables de répondre aux exigences croissantes de notre univers numérique en constante évolution.

Dans ce chapitre, nous allons explorer les différences fondamentales entre les systèmes centralisés et distribués, ainsi que les divers types de systèmes distribués qui ont émergé. Cependant, notre attention se portera tout particulièrement sur le paradigme du Cloud Computing, une avancée majeure ayant révolutionné la manière dont les ressources informatiques sont gérées et utilisées. Nous aborderons l'évolution historique du Cloud Computing, ses caractéristiques, ses modèles de déploiement, les types de services qu'il propose, ainsi que les acteurs clés qui ont contribué à son développement. Enfin, nous examinerons les nombreux avantages qu'offre le Cloud Computing, tout en mettant en lumière les défis de recherche persistants dans cet environnement en perpétuelle évolution.

1.2 Systèmes centralisés vs distribués

Les systèmes distribués et les systèmes centralisés se distinguent fondamentalement par leur architecture et leur mode de fonctionnement, comme illustré dans la figure 1.1. Dans un système centralisé, toutes les ressources, les données et le processus de prise de décision sont concentrés au sein d'un nœud central unique. Cette approche présente l'avantage indéniable d'une gestion simplifiée et d'un contrôle cohérent. Cependant, cette centralisation expose le système à des vulnérabilités notables, telles que des points de défaillance uniques, des goulets d'étranglement et des limitations de scalabilité. Ces limites peuvent entraver la croissance et l'efficacité du système, le rendant moins apte à gérer des charges de travail en constante évolution.

En revanche, les systèmes distribués adoptent une structure décentralisée, où les ressources et les responsabilités se répartissent entre plusieurs nœuds interconnectés. Cette architecture évoluée permet une meilleure to-

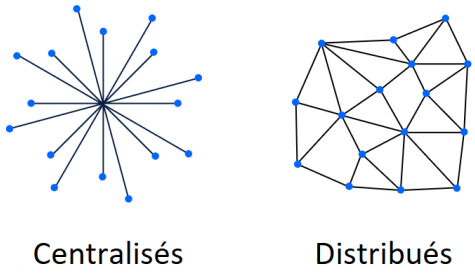


FIGURE 1.1 – *Systèmes centralisés vs distribués*

lérance aux pannes, une répartition optimale des charges de travail et une scalabilité accrue, rendant le système adapté aux environnements dynamiques. Les systèmes distribués tirent parti de leur nature distribuée pour maximiser la résilience face aux défaillances matérielles et logicielles, assurant ainsi une plus grande disponibilité. De plus, en permettant une répartition des tâches et une parallélisation efficace des traitements, les systèmes distribués peuvent améliorer considérablement les performances du système, répondant ainsi aux exigences des applications modernes.

Toutefois, les systèmes distribués ne sont pas sans défis. La coordination et la communication entre les nœuds doivent être soigneusement orchestrées pour garantir la cohérence des données et éviter les conflits. Les protocoles de communication sophistiqués sont nécessaires pour synchroniser les actions entre les nœuds, ce qui peut entraîner des complexités supplémentaires en matière de cohérence des données à travers le système.

1.3 Types de systèmes distribués

Les systèmes distribués peuvent être classés de différentes manières. Dans [8], trois classes essentielles ont été identifiées : les systèmes de calcul distribué, les systèmes d'information distribués et les systèmes pervasifs distribués.

Les systèmes de calcul distribué sont primordiaux pour les tâches informatiques à haute performance. Ils s'articulent autour d'une répartition des ressources sur un réseau distribué pour gérer des calculs complexes. D'autre part, les systèmes d'information distribués se trouvent au sein des organisations confrontées à une multitude d'applications en réseau. Ces organisations doivent relever le défi de l'interopérabilité entre divers systèmes. Pour y répondre, de nombreux middleware ont été conçus pour faciliter l'intégration d'applications dans des systèmes d'information à l'échelle de l'entreprise.

Les systèmes distribués pervasifs se démarquent par leur omniprésence et leur stabilité dans notre environnement. Ces systèmes ont évolué avec l'avènement des dispositifs informatiques embarqués et mobiles, ils opèrent dans des contextes souvent instables. Ces dispositifs, petits, alimentés par batterie, mobiles, et généralement connectés sans fil. L'une de leurs caractéristiques importantes est le manque de contrôle administratif humain, car ils doivent s'adapter automatiquement à leur environnement.

Dans les domaines de l'Internet des Objets (IoT) et de la domotique, ces systèmes jouent un rôle clé.

La classification précédente met l'accent sur les domaines applicatifs plutôt que sur les caractéristiques internes. Pourtant, l'organisation interne et les spécificités des ressources , telles que la répartition géographique et le support de communication, revêtent une importance capitale. Ces facteurs impactent les besoins de performances des applications. Dans la suite, une classification sera présentée, davantage axée sur l'organisation interne et les caractéristiques des ressources.

La grappe d'ordinateur (Cluster) : cette architecture consiste en un ensemble d'ordinateurs connectés par un réseau local (LAN) rapide et fiable. Elle vise à garantir une haute disponibilité et à surmonter les limites d'un seul ordinateur. Les ressources sont quasi-homogènes, avec un même système d'exploitation et des logiciels similaires. Un seul nœud (ordinateur), le nœud maître, exerce le contrôle sur l'ensemble de la grappe. Cette approche est largement utilisée pour les applications nécessitant des performances élevées et une haute disponibilité.

La grille informatique (Grid) : les grilles informatiques représentent une évolution avancée des systèmes distribués. Elles regroupent des ordinateurs hétérogènes répartis sur différents sites, gérés par plusieurs organisations. Ces sites sont interconnectés via un réseau étendu (WAN) et chaque site contient plusieurs ressources informatiques administrées de manière autonome et uniforme. Les grilles informatiques impliquent souvent des collaborations entre différentes organisations ou groupes de recherche, chacun contribuant avec ses ressources informatiques. Cette architecture permet de combiner la puissance de calcul de multiples nœuds pour réaliser des simulations et des analyses qui dépassent largement les capacités des ordinateurs individuels.

Les systèmes pair-à-pair (Peer-to-Peer (P2P)) : parmi les plus répandues, ces structures impliquent que chaque nœud agisse en tant que client et serveur simultanément, instaurant une égalité entre participants. En contraste avec les architectures client-serveur classiques, où un serveur centralisé répond à plusieurs clients, les systèmes P2P exploitent les ressources disponibles sur chaque nœud pour offrir des services de manière décentralisée. Concrètement, dans un réseau P2P, chaque nœud dispose de ses propres capacités de stockage, de traitement et de transmission de données. Lorsqu'un utilisateur demande un service ou partage des fichiers, ce nœud peut contribuer en mettant à disposition ses ressources. Cette approche se distingue par l'absence d'un serveur central unique pour répondre aux demandes. Au contraire, elle capitalise sur la puissance collective des nœuds répartis pour garantir la disponibilité et la résilience du service. Cependant, cette flexibilité accrue s'accompagne d'une légère réduction de la fiabilité et de la stabilité par rapport aux systèmes centralisés. En outre, les systèmes P2P doivent faire face à des défis de sécurité, tels que la vulnérabilité aux attaques de type Sybil, où un attaquant peut créer plusieurs identités pour influencer le réseau. Ils doivent également relever des défis en matière de recherche efficace de ressources distribuées et de gestion de la cohérence des données.

Superordinateur (Supercalculateur) : les superordinateurs représentent l'apogée des systèmes distribués en termes de performances de calcul. Ils sont conçus pour des applications scientifiques et techniques nécessitant d'énormes ressources de calcul, comme la modélisation climatique, la recherche en astrophysique, la simulation moléculaire et les prévisions météorologiques à long terme. Les superordinateurs se distinguent par leur capacité de traitement massivement parallèle, avec des milliers, voire des millions de processeurs travaillant simultanément sur différentes parties d'un problème complexe. En raison de leur complexité et de leur coût élevé, les superordinateurs sont généralement exploités par des institutions de recherche, des organismes gouvernementaux ou de grandes entreprises. Ils jouent un rôle crucial dans les avancées scientifiques et technologiques, permettant de simuler et de résoudre des problèmes qui seraient autrement inaccessibles aux méthodes de calcul traditionnelles.

Les réseaux de capteurs (Sensor Network) : les réseaux de capteurs sont une collection de micro-ressources informatiques (micro-capteur ou système embarqué) reliées le plus souvent par un réseau sans fil. Leur objectif est de collecter, d'échanger et de transmettre des données, généralement environnementales, de manière autonome vers un ou plusieurs points de collecte. Ils œuvrent dans divers domaines comme la surveillance environnementale, la gestion du trafic, l'agriculture intelligente, etc. Leurs caractéristiques principales sont leur capacité d'autonomie et leur déploiement dans des environnements distribués et difficiles d'accès.

Le Cloud : du point de vue systémique, le Cloud Computing est une approche système qui permet le stockage et l'accès aux données via Internet plutôt que par le disque dur d'un ordinateur local. Il utilise la virtualisation pour mutualiser les ressources, favorisant ainsi la flexibilité, la disponibilité et un plan de reprise à moindre coût. Cette architecture est devenue essentielle pour de nombreuses entreprises qui cherchent à optimiser leurs ressources et à répondre à la demande croissante d'accès à distance aux données.

Dans la section suivante, nous explorerons plus de détails les différentes facettes du paradigme du Cloud Computing, en examinons ses types de services, ses caractéristiques distinctives, ses avantages et les enjeux de recherche qui subsistent ce modèle révolutionnaire.

1.4 Paradigme du Cloud Computing

1.4.1 Historique

Le Cloud Computing, un concept dont les origines remontent aux premières décennies de l'informatique moderne, s'ancre dans une époque où les ordinateurs occupaient des salles entières et les ressources informatiques étaient rares et coûteuses. Les années 1950 et 1960 ont été marquées par l'avènement des systèmes informatiques centralisés basés sur le "time-sharing", une des premières manifestations du partage de ressources entre utilisateurs via des terminaux. Cependant, les utilisateurs devaient réser-

ver du temps de calcul spécifique et n'avaient accès qu'aux performances de l'ordinateur central pendant cette période.

Dans les décennies qui ont suivi, la virtualisation est devenue un élément essentiel du développement informatique. La division des ordinateurs en multiples machines virtuelles a permis une consolidation et un partage plus efficace des ressources. Néanmoins, cette technologie restait onéreuse et était principalement réservée aux grandes entreprises et institutions de recherche [9]. Dans cette même période, les premiers réseaux d'ordinateurs interconnectés ont émergé, dont ARPANET, précurseur d'Internet. Cette avancée a jeté les bases du Cloud Computing en rendant possible le partage et l'accès à distance aux ressources informatiques. Cependant, le concept n'a pas gagné une large reconnaissance avant les années 1990, lorsque la croissance d'Internet a ouvert de nouvelles perspectives pour les services d'hébergement Web.

Le véritable essor du Cloud Computing a débuté dans les années 1990, parallèlement à la popularisation d'Internet. L'émergence du World Wide Web a permis aux entreprises de proposer des services en ligne, tels que le stockage de données et les applications, aux utilisateurs via les navigateurs web. Ce moment marquant a jeté les bases du modèle de "service en ligne" et a ainsi posé les fondements du Cloud Computing moderne.

Vers la fin des années 1990 et au début des années 2000, plusieurs entreprises ont joué un rôle de pionnier dans le développement des services de Cloud Computing. En 1999, Salesforce fut précurseur en proposant l'une des premières applications de gestion de la relation client (CRM) basées sur le Cloud. De même, en 2002, Amazon Web Services (AWS) a lancé ses premiers services de stockage et de calcul sur le Cloud.

À cette époque, les data centers souffraient de sous-utilisation, car les entreprises réservaient une partie de leurs capacités pour faire face à des pics d'utilisation occasionnels, comme les périodes de fêtes. Pour remédier à cela, Amazon prit l'initiative de proposer ses serveurs en location à la demande, accompagnés de ses outils S3 (Simple Storage Service) et EC2 (Elastic Compute Cloud), offrant respectivement des services de stockage de données et de calcul. Cette démarche novatrice a permis à Amazon de devenir un pionnier du Cloud Computing moderne à grande échelle en 2006. En proposant des services de calcul et de stockage à la demande, AWS a révolutionné la manière dont les entreprises abordaient l'informatique, en leur permettant de louer des ressources plutôt que de les posséder physiquement, ouvrant ainsi la voie à une infrastructure de Cloud public [10].

Au cours des années 2010, le Cloud Computing a connu une croissance exponentielle, portée par d'importants acteurs de l'industrie tels que Microsoft, IBM, Google et Sun, qui ont investi massivement dans le développement d'infrastructures Cloud et de services associés. De nos jours, le terme "Cloud Computing" est devenu omniprésent dans le paysage technologique, propulsant une multitude d'applications et de services utilisés à l'échelle mondiale. Son évolution continue est alimentée par les progrès technologiques et les besoins croissants en matière de stockage et d'accès aux données, promettant ainsi de façonner l'avenir de l'informatique dans les années à venir.

1.4.2 Définition

Le terme « Cloud » signifiant « nuage » et « Computing » signifiant « informatique », le Cloud Computing représente l'informatique en nuage, également désignée comme l'informatique dématérialisée dans une traduction anglaise-française littérale. Plusieurs définitions du Cloud Computing ont été proposées par diverses communautés académiques et industrielles [11, 12, 13, 14, 15]. Cependant, nous retiendrons la définition du NIST, qui décrit le Cloud Computing comme un modèle offrant un accès réseau omniprésent, pratique et à la demande à un ensemble partagé de ressources informatiques configurables (comme les réseaux, les serveurs, le stockage, les applications et les services). Ce modèle permet une provision rapide et un retrait de ces ressources avec un effort minimal de gestion ou d'interaction avec le fournisseur de services. Ce modèle en nuage se compose de cinq caractéristiques fondamentales, de trois modèles de service et de quatre modèles de déploiement [16].

1.4.3 Caractéristiques du Cloud Computing

Le Cloud Computing, selon les définitions précédentes, englobe cinq caractéristiques essentielles largement adoptées par les fournisseurs de services. Nous allons détailler chacune de ces caractéristiques ci-dessous [16] :

— **Libre-service à la demande :** Le Cloud Computing offre aux utilisateurs un accès convivial et automatisé pour allouer et configurer les services et ressources informatiques selon leurs besoins spécifiques. L'intervention humaine du fournisseur de services n'est plus nécessaire, permettant ainsi aux consommateurs de gagner en autonomie et en flexibilité.

— **Large accès au réseau :** Les services Cloud sont accessibles à distance via des mécanismes standards, garantissant leur disponibilité sur diverses plates-formes client telles que smartphones, tablettes, ordinateurs portables et stations de travail. Cette accessibilité favorise une utilisation aisée et transparente des ressources à travers le réseau.

— **Ressources partagées :** les ressources informatiques du fournisseur sont mises en commun à travers un modèle multi-locataire basé sur une technologie de virtualisation. Ce modèle multi-locataire permet de servir simultanément plusieurs consommateurs tout en affectant et réaffectant dynamiquement les ressources physiques et virtuelles en fonction de leurs demandes spécifiques. Cette approche assure une utilisation efficace des ressources et une indépendance partielle vis-à-vis de leur localisation exacte.

— **Flexibilité rapide :** Une des caractéristiques les plus marquantes du Cloud Computing réside dans sa capacité à fournir une flexibilité rapide. Les infrastructures Cloud peuvent être redimensionnées dynamiquement, permettant aux utilisateurs d'adapter leurs ressources informatiques en fonction de leurs besoins évolutifs. Cette élasticité donne l'illusion d'une disponibilité illimitée de ressources à tout moment.

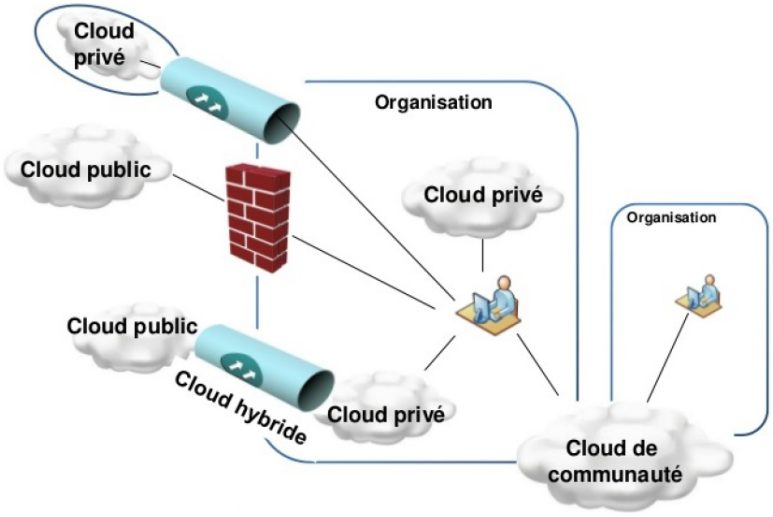


FIGURE 1.2 – *Modèles de services du Cloud Computing*

— **Service mesuré :** Les fournisseurs de services Cloud fournissent des métriques d'utilisation détaillées, permettant aux utilisateurs de comprendre et de contrôler leurs coûts d'utilisation. Grâce à un modèle de facturation à l'utilisation (pay as you go), les clients sont facturés précisément pour la durée exacte des ressources informatiques utilisées. Cette transparence renforce la confiance entre le fournisseur et l'utilisateur.

Le Cloud Computing est souvent comparé à d'autres services d'infrastructure courants, tels que l'approvisionnement en eau ou en électricité. De la même manière que les utilisateurs n'ont pas besoin de comprendre les mécanismes internes d'une centrale électrique pour bénéficier de l'électricité, le Cloud Computing offre une approche similaire. Les utilisateurs peuvent utiliser les ressources informatiques (espace de stockage, puissance de calcul) sans se soucier de la composition matérielle, de la même manière qu'ils utilisent l'électricité sans connaître les détails du réseau électrique.

1.4.4 Modèles de déploiements du Cloud Computing

Les modèles de déploiement du Cloud Computing se distinguent par la manière dont les ressources physiques sont utilisées par les parties prenantes, qu'elles soient chez l'utilisateur ou chez un fournisseur, partagées ou non, destinées aux entreprises ou à d'autres types d'utilisateurs. Dans la figure 1.2, nous illustrons les quatre modèles ou typologies de déploiement offerts par le Cloud, chacun présentant des caractéristiques distinctes adaptées aux diverses exigences de sécurité, de confidentialité et de contrôle des entreprises.

Cloud privé

Le Cloud privé met l'ensemble de ses ressources exclusivement à la disposition d'une seule entreprise ou organisation. Il peut être adminis-

tré soit en interne par l'entreprise elle-même (Cloud privé interne), soit par un prestataire tiers (Cloud privé externe). Les ressources d'un Cloud privé sont généralement localisées dans les locaux de l'entreprise ou chez un fournisseur de services, avec un accès sécurisé via un réseau privé virtuel (VPN). Cette approche garantit la sûreté et la sécurité des données et d'autres informations sensibles de l'entreprise, car seules les parties autorisées ont accès au Cloud, et aucune donnée sensible n'est partagée avec d'autres organisations.

Cloud communautaire

Le Cloud communautaire fonctionne de manière similaire au Cloud privé, mais il permet à plusieurs clients de partager des services et des instances matérielles dédiées. La composition des utilisateurs n'est pas aléatoire, mais est ciblée, réunissant plusieurs clients, généralement issus du même secteur d'activité ou ayant des intérêts similaires. La gestion du Cloud communautaire peut être assurée par une ou plusieurs organisations de la communauté, par un tiers, ou par une combinaison de ces acteurs. L'objectif est de réaliser des économies par rapport à la mise en place de plusieurs Clouds privés séparés.

Cloud public

Le Cloud public repose sur le modèle standard du Cloud Computing, où le fournisseur de services rend ses services et ressources accessibles à tout utilisateur disposant d'une connexion Internet. Ces services sont ouverts au grand public, qu'il s'agisse d'individus ou d'entreprises, et sont accessibles depuis Internet. Les services de Cloud public peuvent être gratuits ou facturés en fonction de leur utilisation.

Cloud hybride

Le Cloud hybride est un modèle qui combine plusieurs autres modèles. Comme son nom l'indique, il s'agit d'un mélange de Cloud privé et public, lié par une technologie normalisée ou propriétaire qui permet la portabilité des données et des applications. Ce modèle est généralement utilisé lorsque les entreprises ont besoin d'une infrastructure sécurisée pour leur stockage, mais souhaitent également effectuer d'autres tâches sur des infrastructures Cloud publiques ou communautaires. Pour les entreprises aux besoins complexes, le Cloud hybride offre une solution combinant le meilleur des deux mondes. Par exemple, une entreprise peut faire appel au Cloud public de manière ponctuelle, notamment lors de pics d'activité ou pour des charges de travail temporaires, tout en utilisant ses ressources internes le reste du temps.

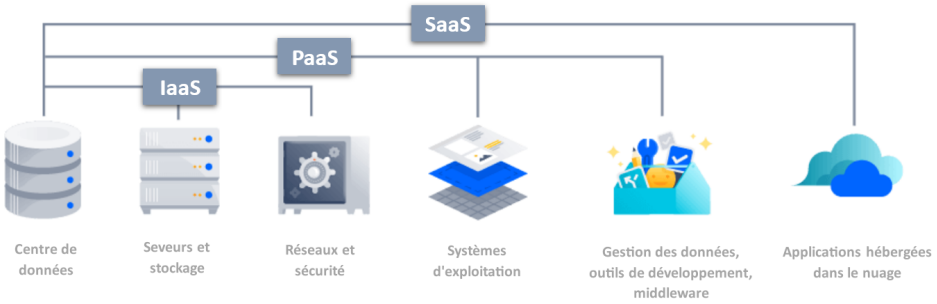


FIGURE 1.3 – *Types de service Cloud Computing*

1.4.5 Types de services du Cloud

Le Cloud Computing est un paradigme orienté service, offrant une gamme variée de services regroupés en trois catégories en fonction de leur niveau d'abstraction. Ces modèles fournissent des solutions adaptées aux besoins des utilisateurs, allant des applications prêtes à l'emploi aux environnements de développement personnalisés, en passant par l'accès direct aux ressources informatiques sous-jacentes, comme illustré dans la figure 1.3. Dans ce qui suite, nous détaillerons ces trois modèles de services fondamentaux.

Infrastructure as a Service (IaaS)

Le modèle IaaS est le plus basique des services d'informatique en nuage. Utilisé comme base pour d'autres modèles et également offert comme produit indépendant, l'IaaS externalise l'infrastructure matérielle du service informatique auprès d'un fournisseur tiers. Ce modèle donne aux utilisateurs un accès via une interface virtuelle à des ressources fondamentales telles que le traitement, le stockage, le réseau, et bien plus encore, le tout hébergé sur des serveurs externes. Cependant, les utilisateurs conservent la responsabilité de gérer les systèmes d'exploitation, le middleware, ainsi que les applications. Parmi les fournisseurs populaires d'IaaS, on retrouve Amazon Elastic Compute Cloud, Microsoft Azure, IBM SoftLayer et Google Compute Engine (GCE).

Platform as a Service (PaaS)

Le modèle PaaS offre un niveau d'abstraction supérieur à l'IaaS et s'adresse principalement aux développeurs d'applications. Il fournit une plateforme permettant aux développeurs de créer, tester, déployer et gérer leurs applications dans un environnement sans se soucier de la configuration et de la gestion de l'infrastructure sous-jacente du Cloud, tels que le réseau, les serveurs et les systèmes d'exploitation. Dans ce modèle, le fournisseur prend en charge le système d'exploitation et les outils infrastructurels, tandis que le client peut contrôler ses propres applications et ajouter des outils personnalisés. Les principaux fournisseurs de PaaS incluent SalesForce , Google (Google App Engine), Microsoft (Windows Azure) et SAP (Systems, Applications and Products for data processing), ainsi que d'autres solutions telles qu'Elastic Beanstalk.

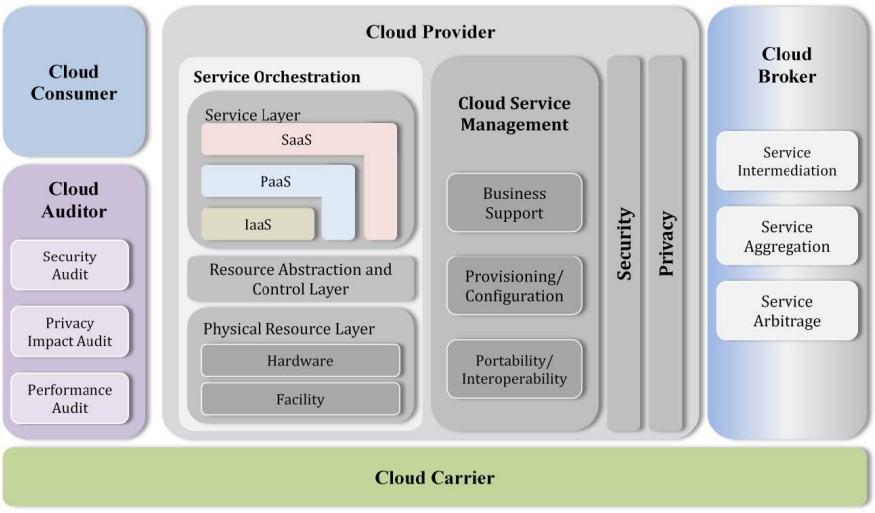


FIGURE 1.4 – *Modèle de référence conceptuel du Cloud NIST [1]*

Software as a Service (SaaS)

Le modèle SaaS se réfère à tout service Cloud permettant aux clients d'accéder à des applications logicielles via Internet, sans nécessiter d'installation, de mise à jour ou de maintenance du logiciel. Les applications et l'infrastructure du Cloud sont gérées par le fournisseur de SaaS de manière transparente pour les utilisateurs. Ces derniers peuvent y accéder à tout moment depuis n'importe quel dispositif connecté, tels que des ordinateurs portables, des tablettes ou des smartphones. Les applications peuvent être utilisées directement via l'interface disponible ou en utilisant des API (Application Programming Interface) fournies (souvent réalisées grâce aux Web Services ou à l'architecture REST (Representational State Transfer)). Parmi les principaux fournisseurs de SaaS, on retrouve Salesforce (logiciels CRM), G-suite et Google (Gmail, Google Apps), ainsi qu'une variété d'autres solutions disponibles gratuitement ou sur abonnement.

1.4.6 Acteurs du Cloud Computing

Selon l'architecture de référence du Cloud Computing décrite par le NIST [1], cinq acteurs principaux interviennent, comme illustré dans la figure 1.4. Chaque acteur représente une entité (une personne ou une organisation) qui participe à une transaction ou à un processus et/ou exécute des tâches dans le Cloud.

— **Consommateur (Cloud Consumer) :** Le consommateur joue un rôle essentiel dans l'écosystème du Cloud Computing. Il peut être une personne ou une organisation ayant une relation commerciale avec un fournisseur d'informatique Cloud et utilisant ses services. Les consommateurs ont la liberté de choisir le fournisseur qui correspond le mieux à leurs besoins. Ils consultent le catalogue de services du fournisseur, demandent les service appropriés, établissent des contrats de service et utilisent les service fournis. Les consommateurs sont également responsables de la gestion des paiements pour les services utilisés. Pour

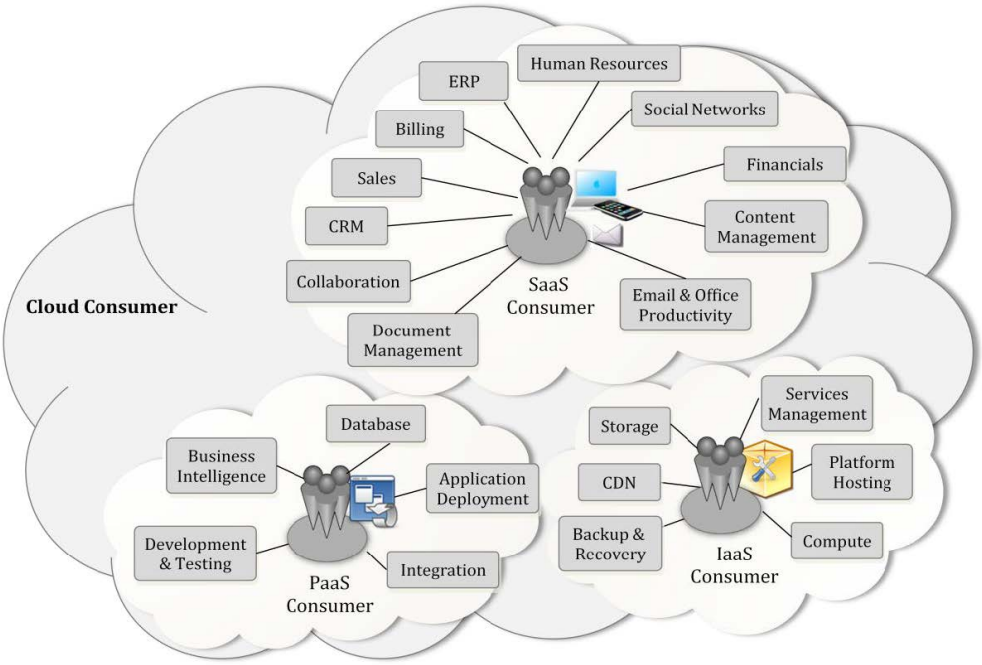


FIGURE 1.5 – *Exemples de services pour un consommateur de Cloud [1]*

faciliter leur prise de décision, les fournisseurs doivent énoncer clairement les accords de niveau de service (SLA) qui spécifient les exigences de performance technique attendues, les mesures prises en cas de défaillance, ainsi que les limitations et obligations que les consommateurs doivent accepter [17].
En fonction des services demandés, les activités et les scénarios d'utilisation peuvent varier d'un consommateur à l'autre. La Figure 1.5 présente quelques exemples de services Cloud disponibles pour un consommateur.

— **Fournisseur (Cloud Provider) :** Au cœur du Cloud Computing, nous trouvons les fournisseurs de services. Ils sont responsables de mettre à disposition les services pour les parties intéressées, qu'il s'agisse d'individus, d'organisations ou d'entités. Ces fournisseurs gèrent l'infrastructure informatique nécessaire pour offrir les services. Ils exécutent le logiciel Cloud qui assure la fourniture des services aux consommateurs via un accès réseau.
— **Auditor (Cloud Auditor) :** Pour garantir la transparence et la confiance dans le Cloud Computing, les auditeurs de services Cloud jouent un rôle crucial. Ils effectuent des examens indépendants des contrôles des services Cloud afin d'exprimer des opinions objectives. Ces audits vérifient la conformité aux normes en examinant des preuves objectives concernant la sécurité, à l'impact sur la vie privée, aux performances, etc. Les auditeurs contribuent à renforcer la sécurité et la qualité des services Cloud en évaluant les pratiques mises en place par les fournisseurs.
— **Courtier (Cloud Broker) :** Avec la croissance continue du Cloud Computing, l'intégration des services peut s'avérer trop complexe à gérer pour les consommateurs. Les courtiers Cloud jouent un rôle clé dans ce contexte. Ils agissent comme des entités intermédiaires aidant les consommateurs à gérer l'utilisation, la performance et l'approvision-

nement des services Cloud. Au lieu de contacter directement un fournisseur, les consommateurs peuvent solliciter un courtier Cloud pour obtenir des services Cloud. Ces courtiers Cloud facilitent les relations entre les fournisseurs et les consommateurs de Cloud, simplifiant ainsi l'accès aux services.

— **Carrier (Cloud Carrier) :** Les transporteurs Cloud agissent comme des intermédiaire assurant la connectivité et le transport des services entre les consommateurs et les fournisseurs Cloud. Ils offrent aux consommateurs un accès aux services Cloud via des réseaux, des télécommunications et d'autres dispositifs d'accès. Ils établissent également des accords de niveau de service avec les fournisseurs pour garantir une consommation cohérente des services Cloud, conformément aux obligations contractuelles envers les consommateurs.

1.4.7 Avantages du Cloud Computing

Le Cloud Computing offre une panoplie de bénéfices incontestables à ses utilisateurs finaux. Voici quelques-uns de ses avantages :

— **Rentabilité :** le Cloud Computing est probablement la méthode la plus rentable à utiliser. Contrairement aux logiciels traditionnels installés localement, il évite aux entreprises les coûts onéreux associés aux licences d'applications et à l'infrastructure matérielle. En optant pour le Cloud, les organisations économisent considérablement en n'ayant à débourser que des frais de propriété. De plus, l'utilisation de terminaux simples et peu coûteux pour accéder au Cloud dispense les équipes de tout investissement dans des ressources matérielles dédiées. Les fournisseurs de services Cloud jouent un rôle déterminant en transformant les ressources matérielles et informatiques en une marchandise hautement compétitive, encourageant ainsi une concurrence qui se traduit par des coûts toujours plus avantageux.

— **Collaboration accrue :** l'un des atouts majeurs du Cloud Computing réside dans l'amélioration de la collaboration. Cette technologie permet à tous les employés, où qu'ils se trouvent, de synchroniser leurs actions, de travailler simultanément sur des documents et des applications partagées, et de suivre en temps réel les mises à jour critiques de leurs collègues.

— **Évolutivité accrue :** la flexibilité du Cloud Computing offre aux organisations la possibilité de redimensionner leurs ressources à la demande. Cette évolutivité élimine les préoccupations liées au manque de ressources ou à l'excès de capacité, car les applications Cloud peuvent ajuster automatiquement leurs ressources d'infrastructure en réponse aux pics de trafic. Cette fonctionnalité garantit une performance optimale en toute circonstance, sans nécessiter d'interventions manuelles fastidieuses.

— **Accès facile à l'information :** grâce au Cloud Computing, l'accès à l'information devient aisé, peu importe l'emplacement de l'utilisateur, pourvu qu'il dispose d'une connexion Internet. Cette particularité transcende les contraintes géographiques, permettant aux utilisateurs de rester connectés à leurs données et à leurs applications essentielles, où qu'ils soient.

— **Intégration automatique des logiciels :** dans le Cloud Computing, l'intégration logicielle se fait généralement de manière automatique, évitant ainsi des efforts supplémentaires pour personnaliser et intégrer les applications. Les fournisseurs de services Cloud déchargent également leurs clients de la responsabilité de la maintenance, offrant une personnalisation facile. Ainsi, les organisations peuvent sélectionner uniquement les services et applications logicielles qui correspondent le mieux à leurs besoins, sans se soucier de la gestion de l'infrastructure.
— **Meilleures performances :** le Cloud Computing fournit les ressources informatiques les plus récentes et les plus performantes. Les utilisateurs ont accès aux dernières ressources matérielles, y compris les processeurs multicœurs, les GPU (Graphical Processing Unit) et les TPU (Tensor Processing Unit), adaptés aux tâches de traitement intensif. Les fournisseurs Cloud restent constamment à la pointe de la technologie, offrant ainsi une expérience utilisateur hautement performante.
— **Sauvegarde et restauration :** un autre avantage essentiel du Cloud Computing réside dans le stockage des données, ce qui facilite grandement la sauvegarde et la restauration. Les fournisseurs de services Cloud sont compétents pour gérer efficacement la récupération des informations en cas de besoin. Le clonage des machines et les captures instantanées offrent des solutions de sauvegarde fiables et des options de restauration simples en cas de problèmes.

1.4.8 Challenges de recherche en environnement de Cloud Computing

La technologie du Cloud Computing a révolutionné la manière d'accéder aux ressources informatiques et de les utiliser. Sa rentabilité, son évolutivité et sa disponibilité à la demande en ont fait une solution attrayante pour les entreprises comme pour les particuliers. Cependant, il convient de rester objectif en reconnaissant également certains défis potentiels auxquels les organisations doivent être conscientes et qu'elles doivent relever pour tirer pleinement parti de sa puissance. Dans ce qui suit, nous mettons en évidence les principales préoccupations et certains des problèmes de recherche dans le domaine de l'informatique en nuage, ouvrant la voie à des solutions potentielles et à des avancées futures dans ce domaine.

Sécurité et confidentialité

L'informatique dématérialisée a engendré des préoccupations majeures concernant la sécurité et la confidentialité des données. Lorsque les utilisateurs confient leurs informations sensibles à des fournisseurs de services tiers, cela soulève des inquiétudes concernant l'intégrité des données, la protection contre les accès non autorisés et la confidentialité. Pour répondre à ces défis, il est impératif de mettre en place un cryptage robuste, des contrôles d'accès stricts et des mécanismes d'audit efficaces. De plus, les transferts de données entre les utilisateurs et les Datacenters des fournisseurs de services représentent des points vulnérables pour les attaques des pirates. Ainsi, il est essentiel de garantir la sécurité des données et leur confidentialité à la fois sur le réseau et dans les Datacenters où elles sont stockées [18].

Gestion de l'énergie

Il est indéniable que l'un des principaux problèmes concernant les grands centres de données Cloud aujourd'hui est leur consommation d'énergie colossale [19, 20]. Des quantités substantielles d'énergie sont nécessaires pour faire fonctionner les systèmes de refroidissement et alimenter les équipements de calcul, de stockage et de mise en réseau. Malheureusement, la consommation d'énergie qui en résulte se traduit par de fortes émissions de CO_2 qui ont un impact considérable sur la biosphère [21]. Selon un rapport du Natural Resource Defense Council [22], les centres de données aux États-Unis consomment environ 91 milliards de kWh (kilowattheure) d'électricité, et les projections estiment que cette consommation atteindra 200 milliards de kWh d'ici 2030. Cette hausse de la consommation d'électricité fait également grimper les coûts des entreprises, ce qui nuit à la productivité [22]. Par conséquent, il est indispensable et urgent de déployer des efforts de recherche considérables pour concevoir des centres de données efficaces sur le plan énergétique, explorer les sources d'énergie renouvelables et mettre en œuvre des pratiques durables pour préserver notre écosystème, mais aussi pour garantir une ère numérique durable.

Interopérabilité et portabilité

À mesure que les écosystèmes de Cloud Computing se diversifient, il devient vital d'assurer une synchronisation, de portabilité et d'Interopérabilité transparente des services entre les différents fournisseurs de Cloud. L'interopérabilité des services est la capacité des utilisateurs du Cloud à utiliser leurs données et applications entre plusieurs CSP avec la même interface de gestion, tandis que la portabilité de service est la capacité de porter leurs applications d'un CSP à un autre quel que soit leur choix en termes de système d'exploitation, format de stockage ou même API. Pour y parvenir, des protocoles et des API normalisés pour la gestion distribuée des ressources doivent être définis [23, 24]. Bien que certaines normes et API soient déjà adoptées, tel que l'Open Virtualization Format (OVF) [25] pour le déploiement des Machines Virtuelles (VMs) sur des plates-formes hétérogènes ; l'API Libcloud [26] qui fait abstraction de l'hétérogénéité entre les Clouds. Néanmoins, il faudra du temps pour que ces normes soient prises en charge par les fournisseurs de Cloud public [27].

Gouvernance des données et conformité

Stocker et traiter des données dans différentes juridictions complique la garantie de la conformité avec les réglementations et les politiques de gouvernance des données. Des directives claires sont nécessaires pour le traitement et les transferts de données transfrontaliers afin de maintenir des pratiques légales et éthiques dans le Cloud Computing.

Problèmes de Migration

La migration dans le Cloud Computing présente deux problèmes majeurs qui nécessitent une approche optimisée. Le premier problème concerne la transition des données entre fournisseurs de Cloud. Souvent, les données échangées ne possèdent pas de structure standard, car la plupart des fournisseurs de services de Cloud Computing utilisent des applications propriétaires non interopérables. Cette incompatibilité peut entraîner des pertes de données ou des erreurs lors de la migration.

Le deuxième problème majeur est la migration des machines virtuelles pour équilibrer la charge de travail entre les Datacenters et optimiser l'utilisation des ressources. Lorsque certains serveurs sont surutilisés, la migration de leurs charges de travail vers des serveurs sous-utilisés est nécessaire pour éviter les goulets d'étranglement et garantir des performances optimales. Cependant, cette migration doit être réalisée de manière efficiente, car elle peut entraîner des besoins en ressources supplémentaires, tels que l'énergie, la bande passante du réseau et les ressources informatiques. De plus, le processus de migration peut impacter les applications fonctionnant dans les machines virtuelles migrantes, nécessitant une planification minutieuse pour minimiser les interruptions de service.

La littérature propose divers schémas de migration de machines virtuelles, mais l'équilibrage optimal de la charge de travail reste un défi. Il est essentiel de développer des algorithmes avancés capables d'identifier rapidement les points chauds, qui sont des régions où les demandes de ressources sont concentrées. En ayant une détection proactive des points chauds, les migrations pourront être initiées de manière préventive pour répondre efficacement aux changements brusques de la demande de charge de travail.

Consolidation de serveurs

La consolidation des serveurs reste un sujet actuel dans la communauté informatique. Elle englobe toutes les stratégies et technologies capables de réduire le nombre ou la répartition géographique des serveurs. Cette approche vise à optimiser l'efficacité de l'utilisation des ressources tout en réduisant au minimum la consommation d'énergie des infrastructures de Cloud Computing. Cette question peut être abordée sous différents angles. Par exemple, l'un des mécanismes fondamentaux est la consolidation dynamique des machines virtuelles (DVMC). L'approche DVMC implique la réaffectation de machines virtuelles d'un hôte à un autre, réduisant ainsi le nombre d'hôtes actifs dans le centre de données en mettant les hôtes inactifs en mode d'économie d'énergie pour conserver l'énergie [28, 29].

Ordonnacement

L'ordonnancement est un aspect critique qui a un impact significatif sur les performances des environnements Cloud. Ce défi d'ordonnancement englobe deux différents niveaux : l'ordonnancement au niveau de l'application et l'ordonnancement au niveau de l'infrastructure. Le premier im-

plique l'ordonnancement des tâches des applications des utilisateurs (des charges de travail sur les machines) vers les services IaaS, tandis que le second concerne l'affectation des machines virtuelles aux infrastructures physiques (machines physiques) de l'écosystème Cloud.

Ces deux types d'ordonnancement jouent un rôle crucial dans l'optimisation des performances et de l'utilisation des ressources dans les environnements Cloud. Cette thèse se concentre sur le premier niveau d'ordonnancement, où les applications des utilisateurs sont souvent exprimées sous la forme de tâches dépendante et/ou indépendante, et leur exécution dans de tels environnements nécessite des stratégies d'ordonnancement efficaces qui tiennent compte des caractéristiques algorithmiques et architecturales.

1.5 Conclusion

Dans ce chapitre, notre exploration approfondie des systèmes distribués et du paradigme du Cloud Computing confirme leur rôle incontournable et fondamental dans la société moderne. Les systèmes distribués offrent une approche hautement flexible et évolutive, répondant de manière optimale aux besoins croissants en traitement de données et en ressources informatiques. Particulièrement, le Cloud Computing émerge comme une solution révolutionnaire, offrant aux utilisateurs un accès à la demande à un pool de ressources, avec une scalabilité et une rentabilité inégalées. Cette innovation attire de nombreuses organisations vers cette approche prometteuse. Cependant, malgré les nombreux avantages qu'il offre, le Cloud Computing présente également des défis majeurs à relever. L'ordonnancement de tâches se positionne au cœur de ces défis, jouant un rôle critique dans la gestion optimale des ressources Cloud. Une gestion inefficace de l'ordonnancement peut entraîner une sous-utilisation ou une surcharge des ressources, impactant directement les performances du système. Par conséquent, l'optimisation des performances et l'utilisation efficiente des ressources dépendent étroitement de la résolution adéquate de ce challenge.

Dans le cadre de notre travail, nous nous intéressons, dans le deuxième chapitre, au problème de l'ordonnancement des tâches et à l'allocation des ressources dans les systèmes de Cloud Computing.

Ordonnancement de tâches dans le Cloud Computing

2

Sommaire

2.1 Introduction

Les problèmes d'ordonnancement, qui font parties des problèmes d'optimisation combinatoire [30], se retrouvent dans tous les domaines de la vie et exercent une influence substantielle sur la valeur de production et la capacité des systèmes. Des exemples incluent l'ordonnancement d'ateliers de production (workshops) [31], l'ordonnancement de véhicules [32, 33, 34], l'ordonnancement d'Unmanned Aerial Vehicles (UAV) [35, 36, 37], l'ordonnancement dans l'environnement du Cloud [38] et bien d'autres tâches de planification. À mesure que l'échelle d'ordonnancement s'élargit progressivement, les algorithmes d'ordonnancement [39] se perfectionnent continuellement, passant des algorithmes statiques aux algorithmes dynamiques, des algorithmes à objectif unique aux algorithmes à objectifs multiples [40]. Parmi eux, le Cloud Computing, en tant qu'élément fondamental de la puissance de calcul haute performance au XXIe siècle, revêt une importance fondamentale dans l'étude de ses problème d'ordonnancement [41].

Dans cette perspective, l'ordonnancement au sein des services Cloud peut être catégorisé selon deux niveaux majeurs : l'ordonnancement systémique et l'ordonnancement au niveau de l'utilisateur. L'ordonnancement systémique englobe la gestion des ressources au sein des centres de données, tandis que l'ordonnancement au niveau de l'utilisateur gère les problèmes entre le prestataire de services Cloud et le consommateur. Ce dernier niveau prend en compte des facteurs économiques tels que la minimisation des coûts basée sur les préférences élastiques des utilisateurs [42]. Un aspect notable de l'ordonnancement au niveau de l'utilisateur est l'approche orientée marché, où les utilisateurs s'appuient sur les fournisseurs de Cloud pour répondre à leurs besoins de calcul. Cette approche nécessite une prise en compte rigoureuse de la Qualité de Service (QoS) par les fournisseurs, avec des engagements spécifiques négociés dans des accords de niveau de service. Une gestion orientée marché s'impose ainsi pour équilibrer l'offre et la demande de ressources Cloud, incitant économiquement à la fois les utilisateurs et les prestataires, tout en améliorant les mécanismes d'allocation des ressources informatique basés sur la QoS [43].

Ce chapitre est consacré à la classification des problèmes d'ordonnancement dans les services Cloud, en se focalisant spécifiquement au niveau de l'utilisateur. Ces services sont élaborés dans le but d'exécuter les différentes requêtes émises par les utilisateurs. Ces requêtes, généralement formulées sous la forme de tâches, trouvent leur application dans un vaste éventail d'activités scientifiques et commerciales. En analysant la nature de ces problèmes, les types de tâches impliquées, les principaux critères d'ordonnancement, les contraintes et les différents types d'ordonnancement, En outre, nous examinerons les outils de simulation qui fournissent des perspectives précieuses pour évaluer et améliorer les stratégies d'ordonnancement. En fin, une compréhension approfondie de la formulation du problème d'ordonnancement constitue les bases d'une meilleure exploitation des capacités du Cloud Computing.

2.2 Taxonomie de problème d'ordonnancement de tâches dans le Cloud Computing

Dans le contexte du Cloud Computing, l'ordonnancement des tâches est crucial, avec des répercussions diverses en fonction des impératifs de qualité de service. Cette diversité d'objectifs a conduit à une profusion de recherches dans ce domaine. Dans cette section, nous présentons une taxonomie rigoureuse qui vise à offrir une perspective claire et approfondie sur ce problème.

La taxonomie que nous présentons, illustrée dans la Figure 2.1, découle d'une synthèse des travaux de recherche majeurs dans la littérature [43, 44]. Elle se décompose en six catégories fondamentales, chacune abordant un aspect spécifique : la nature du problème d'ordonnancement, le type des tâches, les objectifs principaux de l'ordonnancement, les schémas de mise en correspondance des tâches et des ressources, les contraintes d'ordonnancement ainsi que les outils de simulation.

2.2.1 Nature du problème d'ordonnancement

La complexité de l'optimisation génère des compromis inévitables entre des objectifs concurrents. Dans le domaine spécifique de l'ordonnancement des tâches dans le Cloud, deux catégories d'objectifs existent : l'optimisation à objectif unique et l'optimisation à objectifs multiples.

L'optimisation à objectif unique repose sur une relation d'ordre total entre les solutions réalisables, ce qui facilite la détermination de la meilleure solution. Cela simplifie le processus de décision, car les solutions peuvent être évaluées directement et comparativement. Cette simplicité découle de l'évaluation d'un unique objectif, guidant ainsi l'optimisation dans cet espace unidimensionnel.

Cependant, dans les problèmes d'optimisation multi-objectifs, cette comparaison directe devient impossible. En raison de la nature conflictuelle des objectifs, il n'existe généralement pas de solution unique qui serait à la fois optimale pour chaque objectif. En effet, lorsqu'il y a une multitude d'objectifs en jeu, il est souvent difficile, voire impossible, d'améliorer un objectif sans sacrifier au moins partiellement les autres. Ainsi, l'optimisation multi-objectif exige une approche différente : ils se servent fréquemment de la méthode de domination de Pareto. Cette méthode établit un modèle de comparaison entre les solutions, fondée sur la notion de domination. En utilisant cette technique, les problèmes multi-objectifs peuvent générer un ensemble de solutions optimales de Pareto, reflètent la diversité des compromis entre les objectifs conflictuels. Néanmoins, en vue d'une évaluation de performance concrète, un choix final doit être fait parmi ces solutions optimales, en fonction des préférences et des besoins spécifiques de l'utilisateur.

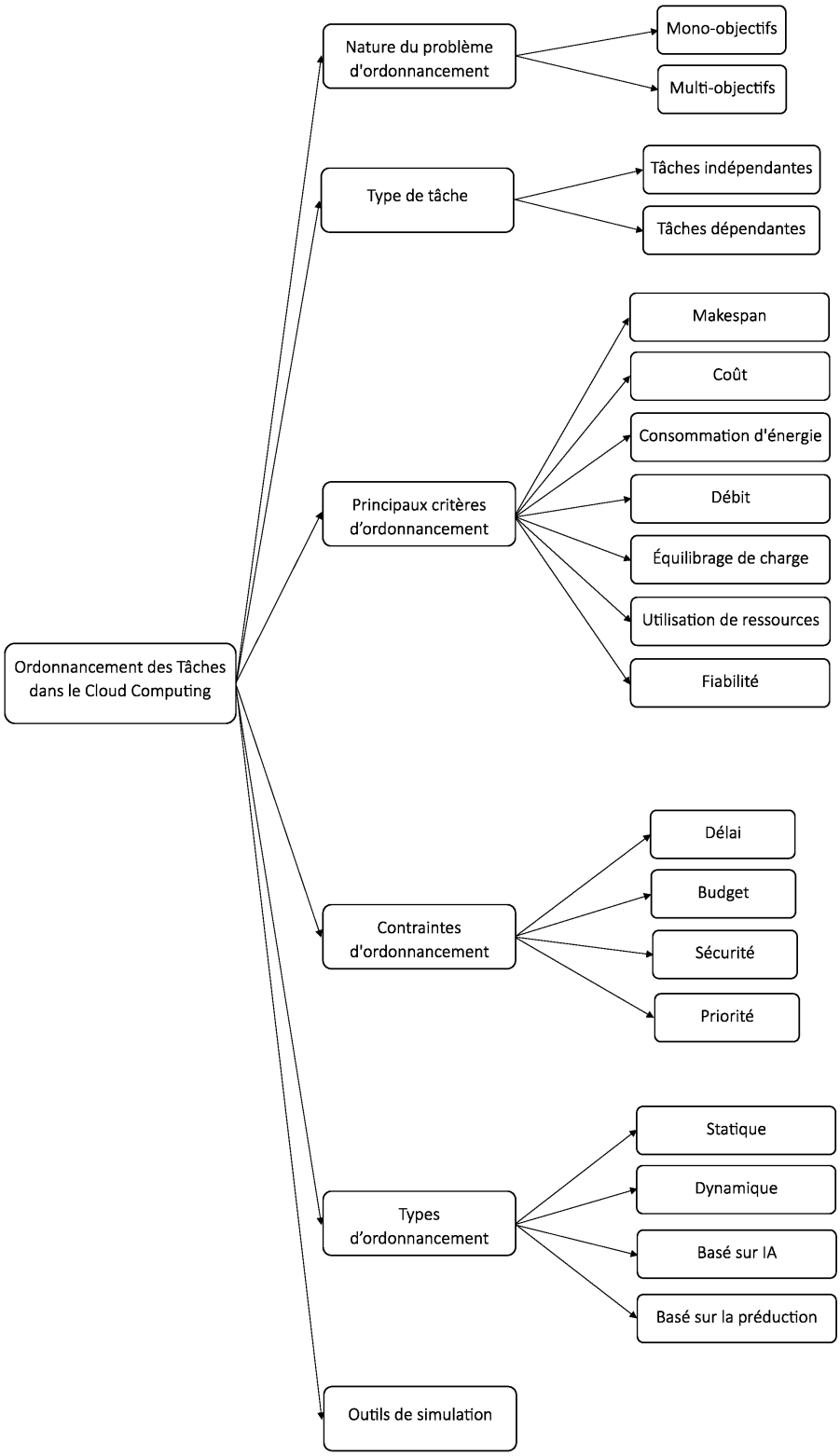


FIGURE 2.1 – *Taxonomie des approches d'ordonnancement des tâches dans le Cloud Computing*

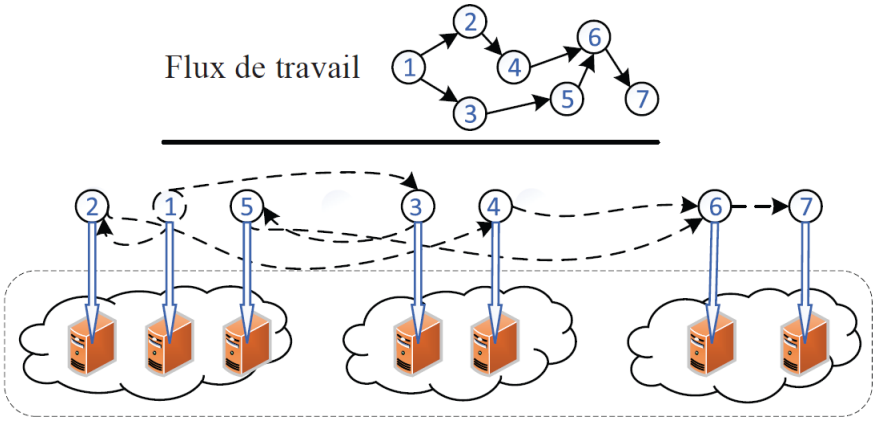


FIGURE 2.2 – *Exemple d'ordonnancement simple d'un flux de travail*

2.2.2 Type de tâches

En fonction de la relation de dépendance entre les tâches, le problème défini précédemment se divise classiquement en deux catégories principales : l'ordonnancement des tâches indépendantes et l'ordonnancement du flux de travail, également appelé ordonnancement dépendant.

L'ordonnancement des tâches indépendantes, comme l'indique son nom, traite des tâches autonomes sans liens de dépendance entre elles. L'objectif principal ici est d'allouer efficacement ces tâches aux machines virtuelles appropriées, tout en satisfaisant les besoins spécifiques de chaque tâche et en optimisant les bénéfices pour les fournisseurs.

En revanche, l'ordonnancement du flux de travail introduit une complexité supplémentaire. Les tâches impliquées dans un flux de travail peuvent être interconnectées de manière complexe ou simple, ce qui rend crucial l'ordre d'exécution. Pour modéliser de telles situations, on utilise généralement des Graphes Acycliques Dirigés (DAG), qui capturent les dépendances entre les tâches, de sorte qu'une tâche ne peut commencer à être exécutée que lorsque sa tâche parente a terminé la sienne. Un exemple simple de cette structure est illustré dans la Figure 2.2, où les nœuds représentent les tâches et les arêtes dirigées indiquent les relations de dépendance entre elles, tout en mettant en évidence les volumes de données à transférer entre deux tâches spécifiques.

2.2.3 Principaux critères d'ordonnancement

Afin d'offrir aux utilisateurs une expérience optimale des services qu'ils ont loués auprès des prestataires de services, tout en maximisant les avantages pour ces derniers, il devient impératif de concevoir un schéma d'ordonnancement distribuant efficacement les tâches sur des ressources appropriées, en se basant sur des critères spécifiques d'ordonnancement. Dans cette optique, nous aborderons ci-dessous les objectifs de planification les plus couramment rencontrés dans le contexte du problème d'ordonnancement, qu'il s'agisse de tâches dépendantes ou indépendantes. Il convient de noter que nous nous focaliserons exclusivement sur la com-

préhension des objectifs impliqués dans l'ordonnancement des tâches dans un environnement Cloud, sans pour autant entrer dans les détails des méthodes de calcul spécifiques que nous explorerons de manière approfondie dans les prochains chapitres.

Makespan : L'optimisation du Makespan est fondamental dans le domaine de l'ordonnancement des tâches. Il se réfère au temps total requis pour achever l'ensemble des tâches soumises par l'utilisateur, en tenant compte à la fois du moment où ces tâches ont été initialement soumises et du moment où elles sont effectivement finalisées [45]. L'objectif sous-jacent est de minimiser cette valeur, traduisant ainsi le désir de l'utilisateur de voir ses tâches accomplies dans un temps le plus rapidement possible. Par conséquent, une valeur de Makespan réduite est souhaitée, signe d'une exécution efficiente. Dans la littérature, la majorité des algorithmes d'ordonnancement proposés s'attèlent à l'optimisation de cette metrique [45, 4].

Coût : Le calcul du coût associé à une application repose sur deux facteurs essentiels, intrinsèquement liés aux ressources cardinales de l'infrastructure Cloud : le coût de calcul et le coût afférent aux transferts de données et à leur stockage. Une allocation substantielle de ressources réduit le temps total d'exécution, mais cela peut engendrer des dépenses accrues et une sous-utilisation potentielle des ressources.

Consommation d'énergie : L'effet de la quantité CPU (Central Processing Unit) et de ressources mobilisées s'exprime directement à travers la consommation énergétique inhérente à l'exécution des tâches. Lorsque les unités CPU ne sont pas utilisées de manière appropriée, le fonctionnement à long terme de l'infrastructure Cloud engendre une consommation énergétique notable, ce qui, à son tour, détériore les performances en raison de l'accumulation de chaleur [44]. Bien que des mesures soient prises, telles que l'acquisition d'un nombre important d'équipements de refroidissement pour maintenir la température sous contrôle, l'empreinte énergétique demeure significative [46]. Dans cette perspective, la réduction optimale de l'indicateur de consommation d'énergie se révèle primordiale tout au long du processus d'ordonnancement.

Débit : Cette mesure fait référence à la quantité de tâches achevées pendant un intervalle de temps donné. Une cadence élevée de réalisation de tâches par unité de temps traduit une capacité d'exécution robuste des ressources Cloud, d'où l'objectif d'atteindre une valeur maximale de cette mesure.

Équilibrage de charge : Au cœur de l'infrastructure fondamentale du Cloud, les éléments clés de traitement sont les machines virtuelles, constituant la base opérationnelle. Lors de la phase d'ordonnancement, le scénario peut engendrer l'attribution simultanée de multiples tâches aux machines virtuelles, ce qui engendre une disparité dans la répartition des charges sur les machines virtuelles. Les prestataires de services Cloud espèrent que les tâches peuvent être réparties sur chaque ressource de la manière la plus raisonnable et la plus homogène possible. Pour cela, le processus d'ordonnancement doit se montrer compétent dans la réparti-

tion équilibrée de la charge parmi les ressources disponibles, prévenant ainsi la surcharge de ces dernières. Un équilibrage adéquat prévient la surcharge des ressources et favorise l'efficacité de l'exécution des tâches, optimisant la durabilité globale des ressources et l'utilisation du Cloud computing.

Utilisation de ressources : L'utilisation des ressources désigne l'exploitation des capacités informatiques durant l'exécution des tâches. L'accroissement de cette utilisation revêt un intérêt conséquent pour le fournisseur de services. Dans l'optique d'optimiser leurs gains, ces fournisseurs mettent en location des ressources limitées aux utilisateurs, visant à en tirer un rendement maximal de manière à ce qu'elles soient pleinement exploitées.

Fiabilité : Les défaillances susceptibles pendant l'exécution des tâches peuvent survenir pour diverses raisons, parmi lesquelles l'indisponibilité des ressources, la défaillance matérielles, les contraintes du réseau ou autres. À cet égard, l'élaboration du processus d'ordonnancement doit systématiquement anticiper et intégrer ces éventualités, en vue d'assurer des exécutions fiables en dépit de situations concurrentielles et de défaillances. Cette assurance se réfère à la fiabilité, traduite par la probabilité d'accomplir les tâches sans interruption jusqu'à leur terme. Cet objectif revêt également une importance capitale, réduisant substantiellement les risques d'échecs dans l'exécution des taches assignés. À cette fin, il est possible de calculer le taux d'échec de manière à ce que la cartographie soit réalisée pour maximiser la fiabilité et réduire les taux d'échec [47]. Une ressource alternative, la réplication et checkpointing sont également des solutions permettant de gérer ces défaillances.

2.2.4 Contraintes d'ordonnancement

Les problèmes d'ordonnancement sont souvent soumis à des contraintes émanant des exigences des utilisateurs, englobant des éléments tels que les délais d'achèvement des tâches, les limites budgétaires, les impératifs de sécurité et les priorités spécifiques pour l'exécution des tâches. Les contraintes associées à l'ordonnancement dans l'environnement du Cloud sont énoncées ci-dessous.

Délai : Pour les applications sensibles à l'aspect temporel, l'achèvement dans un délai précis est impératif. L'ordonnancement sous contrainte temporelle répond à cette exigence en orchestrant l'affectation des ressources de manière à produire des résultats avant l'échéance. Cependant, il faut également tenir compte des aspects financiers, en impliquant une planification qui prend en compte les coûts. Dans le contexte des applications axées sur le facteur temps, une planification robuste avec des délais respectés se révèle essentielle pour renforcer la fiabilité du programme.

Budget : Le budget représente la limite que l'utilisateur impose à l'utilisation des ressources du prestataire de services Cloud. Les décisions d'ordonnancement s'ajustent aux contraintes budgétaires en vue de minimiser la durée globale d'exécution des tâches, tout en garantissant leur achèvement dans les limites financières prévues [48, 49].

Sécurité : En raison de la nature hétérogène des ressources informatiques au sein du Cloud, la sécurité émerge comme un enjeu majeur. Cela suppose que les tâches ne doivent pas être sujettes à des fuites ou des pertes de données privées dues à des attaques extérieures ou des défaillances internes pendant l'exécution.

Priorité : Lorsque les tâches présentent des niveaux d'importance distincts ou des dépendances séquentielles, elles sont affectées à des ordres d'exécution différents dès le début. Les tâches à haute priorité sont exécutées en premier, établissant ainsi une hiérarchie d'exécution.

2.2.5 Types d'ordonnancement

La recherche dans le domaine de l'ordonnancement des tâches a engendré une variété d'algorithmes, chacun adapté à des impératifs spécifiques de qualité de service et visant à minimiser les risques de non-conformité aux accords de niveau de service. Dans les sous-sections suivantes, nous exposons les différents types d'algorithmes existants ainsi que leurs caractéristiques distinctives.

Ordonnancement statique

Les algorithmes d'ordonnancement statique nécessitent des informations préalables sur les tâches pour prendre une décision avant le début de leur exécution. Ils démontrent leur efficacité lorsque la charge de travail demeure stable et que les fluctuations du système sont rares. Cependant, dans le contexte du Cloud Computing où l'environnement est sujet à des fluctuations instantanées de charge, les algorithmes statiques ne constituent pas un choix approprié. Parmi les exemples d'algorithmes statiques, on retrouve le "First-in-First-out" (FIFO), le "Round Robin" (RR), et le "Shortest Job First" (SJF), etc [50].

Ordonnancement dynamique

L'ordonnancement dynamique intervient pendant l'exécution des tâches et ne requiert pas une connaissance complète des propriétés des tâche. Cette approche se révèle pertinente pour gérer les fluctuations de la demande des utilisateurs du Cloud, en particulier lorsque l'optimisation de l'utilisation des ressources est une priorité plus importante que la réduction du temps d'exécution [51].

Ordonnancement basé sur l'Intelligence Artificielle (IA)

Le recours à l'intelligence artificielle pour l'ordonnancement introduit une dimension hautement technique. Il permet la mise en place de techniques intelligentes, fonctionnant et réagissant comme des entités humaines, pour l'allocation et l'ordonnancement des ressources. Cette approche englobe une gamme variée de méthodologies incluant les systèmes

intelligents autonomes, ceux inspirés de la nature, les systèmes basés sur des agents, les réseaux neuronaux, l'apprentissage automatique et les systèmes experts [52].

Ordonnancement basé sur la prédiction

L'ordonnancement basé sur la prédiction repose sur le comportement anticipé des méthodes et des mesures lors de l'allocation des ressources. La capacité à anticiper les besoins en ressources critiques ainsi que la demande future des utilisateurs, grâce à l'allocation automatique ou la réservation des ressources, s'avère cruciale pour une planification efficace des tâches et une allocation optimisée des ressources au sein de l'écosystème du Cloud [53, 43].

2.2.6 Outils de simulation

Il est extrêmement complexe de mettre à l'épreuve de nouvelles stratégies au sein d'un environnement authentique, du fait que certaines expérimentations pourraient potentiellement compromettre la qualité du service rendu à l'utilisateur final. Pour surmonter cette difficulté, divers simulateurs éprouvés se trouvent à disposition, permettant d'explorer de nouvelles stratégies d'ordonnancement tout en évaluant leur efficacité au sein de différents environnements Cloud. Parmi ces outils, la boîte à outils CloudSim occupe une place prépondérante, utilisée dans le but de calculer divers paramètres de qualité de service en étendant les classes préexistantes selon les exigences algorithmiques spécifiques.

Dans la suite de cette sous-section, nous mettrons en lumière certains des outils de simulation prédominants qui ont émergé dans le domaine de Cloud. Dans la réalité actuelle, il est noté que près de 81 % des professionnels se tournent vers des boîtes à outils de simulation, en particulier CloudSim, pour la création d'environnements de test. Cette préférence l'emporte largement sur les essais en environnement réel, qui ne représentent que 19 % des approches adoptées pour l'analyse et l'évaluation des performances de nouvelles approches algorithmiques [54]. Cette prévalence indiscutable témoigne de l'efficacité incontestable des simulations en tant qu'outil d'évaluation pour les solutions émergentes, tout en limitant les risques liés aux tests réalisés dans des environnements réels.

CloudSim

CloudSim émerge comme un outil incontournable pour la simulation et l'expérimentation des infrastructures Cloud. Développé à l'Université de Melbourne en Australie, cette boîte à outils extensible repose sur le langage de programmation Java et capitalise sur la plateforme SimJava. L'architecture de CloudSim met en évidence un courtier de services Cloud (broker), qui orchestre l'ordonnancement et l'approvisionnement en ressources. Les centres de données virtuels agissent en tant que pool de ressources, pouvant instancier un nombre adaptable d'hôtes au sein de chaque centre. Cette adaptabilité est calibrée en fonction des caractéristiques intrinsèques

et des capacités de traitement de chaque centre de données. De la même manière, les machines virtuelles, gérées par des hyperviseurs tels que KVM et XEN, sont déployées sur des hôtes en réponse aux impératifs des utilisateurs. Cette disposition minutieuse tient compte des attributs spécifiques de chaque hôte, incluant sa puissance de calcul, son nombre de cœurs, sa capacité mémoire, et d'autres considérations techniques [55].

Cloud Analyst

Cloud Analyst, une extension puissante de CloudSim, simplifie les tests et l'analyse des applications Internet à grande échelle. Conçu pour s'adapter à la distribution géographique des utilisateurs et des centres de données, Cloud Analyst renforce les capacités d'expérimentation. Cette extension, développée également à l'Université de Melbourne, offre une interface utilisateur graphique (GUI) pour le déploiement en temps réel des centres de données et la surveillance de l'équilibrage des charges. Elle contribue ainsi à une analyse approfondie du comportement des applications dans un environnement Cloud complexe [56].

EMUSIM

EMUSIM, une extension de CloudSim développée à l'Université de Melbourne, exploite les capacités de l'Automated Emulation Framework (AEF) pour émuler les performances de nouveaux algorithmes proposés au sein de l'infrastructure Cloud. En dépit de son absence d'interface graphique, EMUSIM occupe un rôle essentiel dans l'analyse et l'évaluation approfondie des performances algorithmiques. Néanmoins, il présente certaines limitations en termes d'évolutivité en raison de contraintes matérielles et de la complexité liée à la génération de charges de travail substantielles et réalistes [57].

GroundSim

L'outil GroundSim, conçu à l'Université d'Innsbruck en Australie, est une extension open-source basée sur CloudSim. Cette boîte à outils, centrée sur les services d'infrastructure Cloud (IaaS), présente une interface graphique limitée et offre des capacités de simulation en temps réel pour les opérations de stockage Cloud (PaaS). Intégrable à l'environnement ASKALON, GroundSim ouvre la voie à une simulation d'un environnement d'application réel [54].

GreenCloud

Il s'agit d'une boîte à outils open-source mise en œuvre à l'université du Luxembourg à l'aide du langage de programmation C++, et reposant sur la plateforme NS2. Cette boîte à outils offre une interface graphique limitée et un modèle énergétique. Son objectif réside dans l'évaluation minutieuse des algorithmes d'ordonnancement écoénergétiques, en mettant en avant des paramètres de qualité de service tels que la consommation d'énergie des liaisons de communication, des serveurs informatiques et des commutateurs de réseau [58].

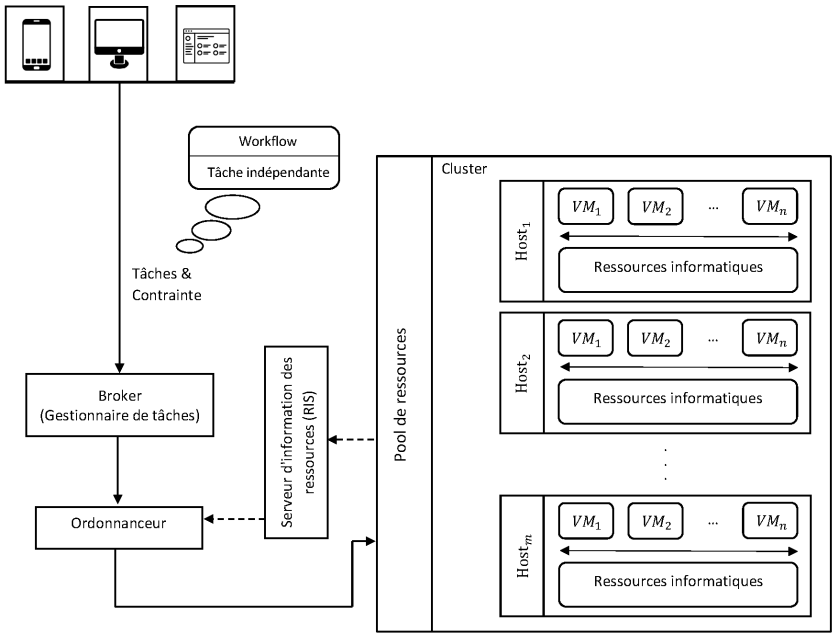


FIGURE 2.3 – *Processus d'ordonnancement dans le Cloud Computing*

Network CloudSim

Il s'agit d'un outil open-source mis en œuvre à l'université de Melbourne à l'aide du langage de programmation Java. Il offre un modèle énergétique aux utilisateurs et se concentre sur l'évaluation des performances des applications informatiques haute performance et des flux de travail complexes. Cet outil étend les capacités de CloudSim en permettant la modélisation de centres de données Cloud réels, malgré l'absence d'interface graphique [59].

2.3 Formulation du problème

Le problème d'ordonnancement se dévoile à travers l'acte d'attribuer des tâches, appelées (Cloudlets), aux machines virtuelles disponibles dans l'environnement du Cloud. Dans cet écosystème, les centres de données Cloud sont structurés en ensembles de ressources, qui peuvent être homogènes ou hétérogènes. Chaque serveur au sein d'un centre de données peut accueillir une ou plusieurs machines virtuelles. Ces ressources informatiques, d'une flexibilité remarquable, sont proposées aux clients sous forme de ressources virtuelles, adaptées à leurs besoins et demandes.

Une vue d'ensemble du processus d'ordonnancement typique est exposée dans la figure 2.3. Lorsqu'un utilisateur soumet des tâches à exécuter, indépendantes ou dans le cadre d'un flux de travail, celles-ci, ainsi que leurs contraintes, sont initialement transmises au broker, également connu sous le nom de gestionnaire de tâches. Ce dernier administre l'ensemble des tâches soumises et les achemine ensuite vers l'ordonnanceur

Cloud. Celui-ci occupe une position cruciale, assignant les machines virtuelles disponibles aux tâches conformément à la stratégie d'ordonnancement préétablie.

Le plan d'ordonnancement crée une liaison cohérente et rigoureuse entre les tâches et les ressources du Cloud, garantissant ainsi une répartition judicieuse des tâches sur les machines virtuelles les plus appropriées pour leur exécution. Dans l'optique de raffiner ce processus, le planificateur coopère avec le Serveur d'Information des Ressources (RIS). Ce dernier recueille des données sur les ressources et capacités des machines virtuelles, permettant ainsi l'identification des ressources idéales pour satisfaire les besoins des utilisateurs.

Cette section scrute le défi de l'ordonnancement en termes de modélisation des centres de données Cloud, des tâches et des machines virtuelles. Elle met en lumière également la complexité intrinsèque à cette problématique.

2.3.1 Modèle de Cloud

Dans le cadre de la conception de systèmes Cloud, la décision concernant la configuration dépend directement des besoins en ressources. Dans le contexte d'un Cloud public, un ensemble de centres de données (DCs) peut être déployé, noté $DC = \{dc_1, dc_2, ..., dc_p\}$, comprenant p centres. Chaque centre de données abrite plusieurs hôtes physiques (PHs). Par exemple, un centre de données dcr possède k hôtes physiques, désignés par $\{ph_1, ph_2, ..., ph_k\}$. Ces hôtes se distinguent par leur nombre de cœurs, déterminant leur puissance exprimée en millions d'instructions par seconde (MIPS), ainsi que par leur mémoire, stockage, bande passante et le Gestionnaire de Machines Virtuelles (VMM) qui assure leur gestion.

2.3.2 Modèle de ressources

Chaque hôte physique ph gère un ensemble de m machines virtuelles $\{vm_1, vm_2, ..., vm_m\}$. Chaque machine virtuelle est définie par des paramètres particuliers : mémoire principale (VM_{mem}), stockage (VM_{st}), nombre de cœurs (VM_{cpus}) et puissance de traitement (VM_{mips}) qui quantifie le nombre d'instructions que la ressource peut traiter par seconde. De plus, chaque machine virtuelle est assortie d'un coût monétaire proportionnel à son utilisation dans le temps.

2.3.3 Modèle de tâches

Les utilisateurs du Cloud soumettent des tâches au fournisseur pour traitement, ignorant les détails de l'infrastructure. Ces tâches varient en termes d'exigences et de ressources nécessaires. Précisément, les utilisateurs envoient n tâches/cloudlets $\{Cl_1, Cl_2, ..., Cl_n\}$ à traiter. Chaque cloudlet nécessite un temps de traitement distinct, lié à sa longueur (C_{li}) exprimée en millions d'instructions (MI).

2.3.4 Complexité du problème

Le défi de l'ordonnancement dans les environnements Cloud tire sa complexité inhérente de multiples facteurs. Les décisions concernant la répartition optimale des tâches entre les machines virtuelles sont influencées par une variété de paramètres, souvent en conflit, et impliquant différents acteurs aux intérêts divergents.

L'évolution des charges de travail, des besoins en ressources et des contraintes spécifiques, ainsi que la diversité des configurations matérielles et virtuelles, amplifie la difficulté du problème. Ainsi, le problème d'ordonnancement des tâches se transforme en une question d'optimisation combinatoire. Il est crucial de noter qu'atteindre une solution optimale globale en temps raisonnable s'avère rarement possible. C'est pourquoi ce problème est catégorisé comme NP-difficile [5], exigeant une gestion en temps polynomial soigneusement conçue pour une utilisation efficace des ressources, tout en respectant les contrats de niveau de service conclus avec les fournisseurs de services.

2.4 Conclusion

Le présent chapitre a été consacré à l'examen de la question cruciale de l'ordonnancement des tâches dans le contexte du Cloud Computing. Cette problématique revêt une importance primordiale, car elle influe directement sur l'efficacité globale du système. L'ordonnancement des tâches, en tant que processus fondamental, englobe la distribution réfléchie des tâches parmi les ressources disponibles. Cette distribution prend en considération à la fois les caractéristiques spécifiques à chaque tâche et les contraintes formulées par les utilisateurs. En effet, Cette prise de décision a un impact majeur non seulement sur les performances globales du système, mais aussi sur la qualité de service fournie à l'ensemble de la communauté d'utilisateurs.

Une analyse en profondeur de cette problématique a mis en évidence sa complexité, illustrée par sa nature de problème NP-difficile bien étudié. Face à cette complexité, des solutions avec temps d'exécution raisonnable sont impératives pour maintenir des performances optimales. Ainsi, le choix judicieux de l'approche de résolution se révèle d'une importance capitale.

Le chapitre suivant se consacrera à exposer diverses méthodes de résolution mises en œuvre dans le contexte de cette problématique. De plus, une revue exhaustive de l'état de l'art des algorithmes d'ordonnancement spécifiquement conçus pour le Cloud Computing sera présentée.

3 Méthodes de résolution pour l'ordonnancement de tâches dans le Cloud Computing

Sommaire

3.1 Introduction

L'ordonnancement des tâches dans un environnement informatique hétérogène distribué représente un défi majeur, se posant comme un problème d'optimisation non linéaire, multi-objectif et NP-difficile. Son objectif essentiel est de maximiser l'utilisation des ressources Cloud tout en répondant aux exigences de qualité de service.

Au vu de ce contexte et de sa complexité croissante, le développement d'algorithmes d'ordonnancement devient de plus en plus compliqué, car il doit gérer un grand nombre de paramètres contradictoires impliquant différents acteurs, dont les intérêts ne sont pas les mêmes. En effet, comme mis en évidence dans le chapitre précédent, l'ordonnancement dans ce cadre présente une caractéristique fondamentale : la coexistence d'objectifs complexes et parfois antagonistes.

Cette particularité rend inefficaces les algorithmes simples pour parvenir à une solution globale optimale. Ainsi, un enjeu majeur se dessine : la nécessité de concevoir une évaluation efficace de ces multiples objectifs, tout en orchestrant judicieusement la répartition des tâches, constituant la charge de travail, au sein du pool des ressources Cloud. Pour y parvenir, il faudrait considérer à la fois les aspects théoriques et pratiques pour mettre en œuvre efficacement et conjointement des méthodes avancées de l'optimisation combinatoire afin d'améliorer la qualité des solutions obtenues.

Dans ce chapitre, notre attention se portera d'abord sur une revue générale des différentes méthodes de résolution des problèmes d'optimisation documentées dans la littérature. Par la suite, nous présenterons un état de l'art exhaustif des algorithmes d'ordonnancement spécifiquement conçus dans le contexte du Cloud Computing. Cette exploration nous permettra d'appréhender la richesse des solutions existantes et d'anticiper les perspectives futures dans ce domaine passionnant et en constante évolution.

3.2 Classification des méthodes de résolution

La diversité et la multitude des problèmes d'optimisation a engendré une pléthore de méthodes visant une résolution efficiente tout en garantissant une qualité de résultat satisfaisante dans un délai raisonnable. Ces méthodes diffèrent par leurs approches, stratégies, performances et qualités des solutions obtenues. De manière générale, deux catégories majeures émergent : les méthodes exactes et les méthodes approchées (voir Figure 3.1). Parallèlement, une catégorie hybride voit le jour, combinant diverses méthodes (exactes-exactes, exactes-approchées, approchées-approchées) afin de tirer parti de leurs atouts respectifs et pallier leurs lacunes.

Le théorème No Free Lunch, établi dans ce contexte, affirme qu'aucune méthode n'est universellement efficace pour tous les types de problèmes.

Dans ce qui suit, nous allons nous intéresser aux méthodes exactes et approchées en mettant l'accent sur les algorithmes et techniques les plus connues.

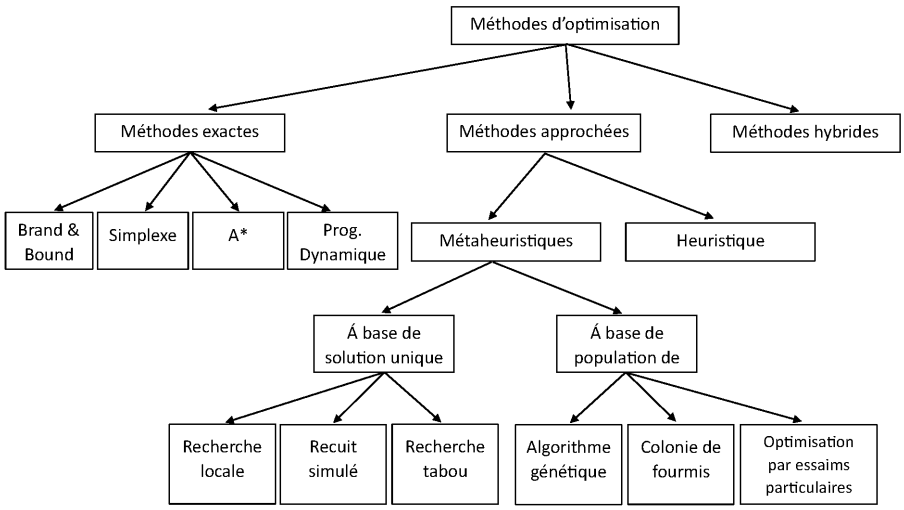


FIGURE 3.1 – *Classification des méthodes de résolution des problèmes d'optimisation*

3.3 MÉTHODES EXACTES

Cette catégorie de méthodes assure l'obtention de solutions optimales du problème traité en parcourant l'ensemble de l'espace de recherche associé à ce dernier d'une manière intelligente. Cependant, elles se révèlent gourmandes en termes de ressources (temps de calcul, mémoire) nécessaires pour accomplir le processus de recherche de la solution optimale et c'est pour cette raison que les méthodes exactes sont plus adaptées aux problèmes faciles de petites tailles.

Parmi les méthodes exactes, l'algorithme de Séparation et Évaluation, connu sous l'appellation anglaise Branch-and-Bound [60], se démarque. Relevant de la famille des approches arborescentes, il explorer la totalité de l'espace de recherche d'une manière intelligente et non exhaustive. En effet, l'exploration est réalisée en représentant le problème sous forme d'un arbre dont la racine correspond à tout l'espace de recherche et chaque éventuelle branche correspond à une décision par exemple une instanciation d'une variable. Ainsi, l'espace de recherche est divisé en plusieurs sous-régions, dont chacune correspond à un sous-arbre. L'efficacité de cette technique réside dans le fait que les branches de l'arbre ne sont pas générées d'une manière systématique. La décision de la non construction d'une branche est conditionnée par une évaluation qui assure que la branche en question ne pourra pas amener à la solution optimale. Par conséquent, certains sous espaces de recherche sont éliminés, aboutissant ainsi à une localisation relativement rapide de la solution optimale.

Une autre méthode est celle de la programmation dynamique, formulée par Richard Bellman dans les années 1950 [61], reposant sur le concept d'optimalité de Bellman : « une solution optimale d'un problème résulte des solutions optimales de ses sous-problèmes ». L'idée de base est donc de décomposer le problème en une suite de sous problèmes plus petits, liés les uns aux autres, résolus de manière récursive, en commençant par les plus petits jusqu'au problème initial. La solution d'un sous problème dans

une étape donnée se calcule en se basant sur la solution du sous-problème précédent.

Toujours dans la famille des méthodes exactes, on peut citer l'algorithme A* [62], les méthodes de backtracking ou de retour sur trace [63], la méthode du simplexe de Dantzig [64], etc.

3.4 Méthodes approchées

Dans des cas pratiques, où des contraintes complexes doivent être satisfaites, les méthodes exactes sont parfois inadaptées. Le temps de calcul exponentiel restreint leur pertinence à des problèmes de petite ou moyenne envergure. Cette réalité a donné naissance aux méthodes approchées, qui, bien qu'elles ne garantissent pas une solution optimale, offrent des solutions de qualité acceptable dans des délais raisonnables.

Pour le délicat problème d'ordonnancement des charges de travail dans le Cloud, les méthodes approchées se sont imposées pour leur praticité. Bien que n'atteignant pas l'optimalité, elles fournissent des solutions sous-optimales qui équilibrent la qualité et le temps de calcul.

Dans ce qui suit, nous allons aborder les deux grandes sous-classes des méthodes approchées, à savoir : les heuristiques et les métaheuristiques.

3.4.1 Heuristiques

Une heuristique est généralement un algorithme conçu spécifiquement pour un problème donné. En exploitant les particularités de ce problème, elle explore seulement une partie de l'espace de recherche pour fournir une solution quasi-optimale en temps polynomial. Les heuristiques offrent donc un compromis intéressant entre le temps d'exécution et la qualité de la solution trouvée. Cependant, en dépit de leur efficacité, les heuristiques par leur nature présentent deux points faibles :

— Les heuristiques sont élaborées pour résoudre un problème bien précis, en utilisant des règles empiriques basées sur l'expérience. Elles requièrent ainsi une maîtrise approfondie et des connaissances assez solides du problème traité.

— Les heuristiques sont guidées par des spécificités liées au problème en question et en sont donc dépendantes, autrement dit, elle ne sont pas applicables aux autres problèmes.

Ces limitations ont conduit à l'apparition des métaheuristiques. Dans le cadre de cette thèse, notre focalisation se tourne vers les métaheuristiques pour résoudre l'ordonnancement des charges de travail dans l'environnement du Cloud Computing.

3.4.2 Métaheuristiques

Contrairement aux heuristiques qui ciblent des problèmes spécifiques, les métaheuristiques sont polyvalentes et applicables à une multitude de

problèmes d'optimisation sans modifications majeures dans leur structure. Le terme "métaheuristique" a été introduit par Fred Glover en 1986 lors de la conception de la recherche tabou pour la différencier des autres heuristiques [65].

D'une manière générale, le fonctionnement des métaheuristiques englobe essentiellement deux phases, à savoir :

— **La diversification ou l'exploration :** la recherche se dirige vers des zones non visitées afin explorer l'espace de recherche d'une manière globale.

— **L'intensification ou l'exploitation :** la recherche se concentre sur des régions locales prometteuses en espérant trouver une solution plus op -timale.

La métaheuristique doit établir un bon équilibre entre ces deux phases pour obtenir une meilleure performance en termes de taux de convergence et de la qualité de solution obtenue.

Dans la littérature, les métaheuristiques sont habituellement classées en deux sous-catégories selon le nombre de solutions manipulées : les métaheuristiques à solution unique, qui travaillent sur un seul point de l'espace de recherche à un instant donné, appelées aussi méthodes à base de voisinage, et les métaheuristiques à base de population, qui travaillent sur un ensemble de points de l'espace de recherche en parallèle.

Métaheuristiques à solution unique

Ces métaheuristiques opèrent sur un seul point de l'espace de recherche à la fois. Elles débutent avec une solution initiale et cherchent à l'améliorer progressivement en explorant son voisinage. Ce type de métaheuristiques appelées également méthodes de recherche locale ou méthodes de trajectoire, car elles construisent une trajectoire dans l'espace des solutions pour converger vers des solutions optimales. Le recuit simulé, la recherche tabou, la recherche à voisinage variable (VNS Variable Neighbourhood Search) sont des instances typiques des méthodes à base de solution unique.

Métaheuristiques à base de population

Les métaheuristiques à base de population travaillent avec une population initiale qu'elles optimisent itérativement pour aboutir à de meilleures solutions. L'utilisation d'une population, plutôt qu'une seule solution, accroît la diversité de la recherche et la probabilité de trouver des solutions de qualité ou même optimales dans un espace de recherche plus large. Cela les rend particulièrement adaptées à l'optimisation de problèmes complexes.

Les travaux de recherche menés dans ce contexte ont permis le développement d'une grande panoplie d'approches et de techniques, telles que les algorithmes génétiques, l'optimisation par essaim de particules, les al-

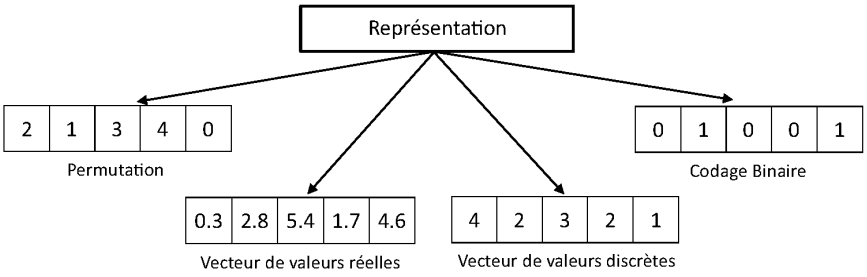


FIGURE 3.2 – *Principaux codages pour des problèmes d'optimisation*

gorithmes de colonies de fourmis, les algorithmes de chauves-souris, la recherche coucou, et bien plus encore.

La plupart des métaheuristiques présentées dans la littérature tirent leur inspiration de phénomènes naturels réels, qu'ils soient physiques ou biologiques. Pour plus de détails sur cette classification des métaheuristiques, référez-vous à la Section 3.6.

3.5 Concepts communs à toutes les métaheuristiques

La conception de toute métaheuristique repose sur deux concepts fondamentaux : la représentation des solutions manipulées par l'algorithme et l'évaluation de ces solutions dans l'espace objectif.

3.5.1 Représentation des solutions

La représentation des solutions est une étape cruciale dans la conception de toute métaheuristique. Elle implique la mise en place d'une structure pour coder les solutions. La pertinence de cette représentation est déterminante pour l'efficacité globale de l'algorithme. Le choix d'une représentation doit être adéquat au problème d'optimisation en question. Différentes représentations peuvent coexister pour un même problème, voire s'appliquer à divers problèmes d'optimisation.

Parmi les principales représentations pour l'optimisation combinatoire, on distingue les vecteurs de valeurs réelles, discrètes, binaires et les permutations. La Figure 3.2 illustre un exemple de chaque représentation. Par exemple, le vecteur de valeurs discrètes trouve souvent son utilisation pour modéliser l'ordonnancement des charge de travaille dans le Cloud Computing.

3.5.2 Évaluation des solutions

L'étape d'évaluation consiste à attribuer des valeurs objectives à une solution donnée. En optimisation mono-objectif, une fonction unique définit l'objectif à atteindre. Cette fonction assigne à chaque solution une valeur unique, établissant ainsi un ordre total entre toutes les solutions. En

cas d'optimisation multi-objectif, plusieurs fonctions objectifs entrent en jeu. L'évaluation se traduit alors par l'attribution d'un vecteur de valeurs à chaque solution. La taille de ce vecteur correspond au nombre d'objectifs et chaque élément quantifie la qualité de la solution en rapport avec une fonction objectif spécifique. Cela génère un ordre partiel entre les solutions, basé sur des relations de dominance.

L'évaluation occupe une place centrale dans la conception des métaheuristiques. Elle oriente le processus vers les solutions adéquates de l'espace de recherche. Toutefois, il est important de noter que, dans la pratique, l'évaluation est souvent l'étape la plus coûteuse en termes de temps de calcul de la méthode de résolution.

3.6 Taxonomie des métaheuristiques

Comme indiqué précédemment, les métaheuristiques sont des algorithmes qui tirent leur conception de processus naturels ou de modèles de comportement observés dans le monde biologique. Cette approche, basée sur l'observation de phénomènes naturels, a donné naissance à une vaste gamme de méthodes d'optimisation regroupées sous le terme d'algorithmes d'optimisation bio-inspirés. Chaque année, cette approche génère une multitude de propositions et d'applications, mettant en évidence l'efficacité de l'adaptation de processus naturels pour atteindre des solutions optimales globales.

Récemment, une étude a établi une taxonomie basée sur l'inspiration naturelle des algorithmes bio-inspirés [2]. À partir de l'analyse minutieuse de plus de 300 algorithmes, cette classification permet une catégorisation plus complète des contributions littéraires. Les résultats de cette classification sont illustrés dans la Figure 3.3, offrant une vue d'ensemble claire de la répartition des propositions.

La catégorie prédominante est celle de l'intelligence en essaim, regroupant près de la moitié des propositions (47 %). Ces algorithmes s'inspirent des comportements collectifs observés chez les insectes sociaux et les animaux migrateurs. Ensuite, la catégorie "Physique et Chimie" représente 19 % des propositions, s'appuyant sur les phénomènes physiques et chimiques pour élaborer des solutions novatrices aux problèmes d'optimisation.

D'autres catégories pertinentes incluent les algorithmes basés sur le comportement social humain (12 %), exploitant les interactions sociales pour guider la recherche de solutions optimales. La catégorie de l'évolution basée sur la reproduction (8 %) s'inspire de la théorie de l'évolution pour créer des populations d'individus et obtenir des solutions évolutives. Une catégorie émergente liée aux mécanismes d'adaptation des plantes (3 %) explore les stratégies végétales pour résoudre des problèmes d'optimisation. Enfin, près de 10 % des propositions ne rentrent pas dans les catégories susmentionnées en raison de leur originalité et singularité.

Pour une classification cohérente, les sous-sections suivantes fourniront des descriptions détaillées des différentes catégories de cette taxono-

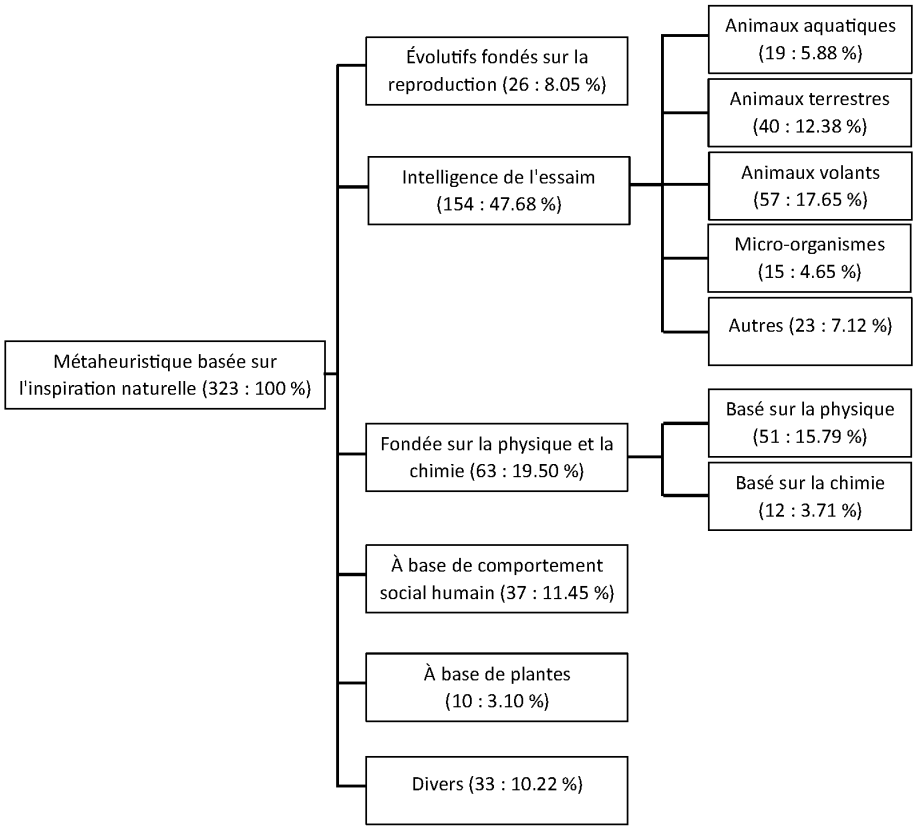


FIGURE 3.3 – *Classification des métaheuristiques à base de sources d'inspiration [2]*

mie, mettant en évidence leurs caractéristiques principales et proposant des exemples d'algorithmes appartenant à chaque catégorie.

3.6.1 Algorithmes évolutionnaires

Cette catégorie regroupe des algorithmes basés sur une population qui s'inspirent des principes de l'évolution naturelle. Chaque individu de la population représente une solution potentielle au problème et possède une valeur de fitness associée. Ces algorithmes reposent sur un processus itératif de reproduction et de sélection à travers les générations successives, permettant à la population de se rapprocher vers des régions potentiellement plus optimales, telles que celles représentées par les meilleures solutions présentes au sein de la population.

Spécifiquement, les algorithmes de cette catégorie se caractérisent par une population dont les individus sont capables de se reproduire et de générer une descendance présentant des variations par rapport à leurs parents. Cette diversité génétique confère aux individus la capacité de s'adapter à leur environnement, favorisant une exploration plus efficace de l'espace des solutions. L'évolution de la population au fil des générations, associée à un processus de sélection basée sur la qualité des solutions (la survie des individus les plus apte), ainsi qu'à d'autres opérations éventuelles, contribue à l'amélioration générale des résultats.

L'originalité des algorithmes évolutionnaires réside principalement dans leur aspect reproductif, qui les distingue des autres approches. Parmi les exemples bien connus de cette catégorie, citons les Algorithmes Génétiques (AG) [66], les Stratégies d'Évolution (ES) [67] et l'Évolution Différentielle (DE) [68].

3.6.2 Algorithmes basés sur l'intelligence en essaim

L'intelligence en essaim est un concept solidement établi dans la communauté scientifique, introduit par Gerardo Beni et Jing Wang en 1989 [69]. Elle se réfère au comportement collectif de systèmes décentralisés et auto-organisés évoluant dans des environnements naturels ou artificiels. Initialement conçu pour les systèmes robotiques, ce concept s'est étendu pour inclure l'émergence d'une intelligence collective à partir d'un groupe d'agents simples régis par des règles comportementales fondamentales.

Les métaheuristiques bio-inspirées basées sur l'intelligence en essaim trouvent leur source d'inspiration dans le comportement collectif observé dans les sociétés animales comme les colonies d'insectes ou les groupes d'oiseaux. Cette intelligence collective permet de résoudre de manière efficace des problèmes d'optimisation. Parmi les premiers algorithmes bio-inspirés basés sur l'intelligence en essaim, figurent l'optimisation par essaim de particules (PSO) [70], rapidement suivi par l'algorithme de colonies de fourmis (ACO) [71]. Ces découvertes pionnières ont engendré de nombreuses autres approches au fil des ans, comme l'algorithme des abeilles artificielles (ABC) [72], ainsi que des algorithmes plus récents tels que l'algorithme de Firefly (FA) [73] et l'algorithme d'optimisation Grasshopper (GOA) [74].

La catégorie des algorithmes basés sur l'intelligence en essaim est riche et se subdivise en sous-catégories, chacune inspirée par un type d'animal. On distingue : (i) les algorithmes inspirés par le vol des oiseaux et des créatures volantes ; (ii) ceux qui s'inspirent des mécanismes de recherche de nourriture et de chasse chez les animaux terrestres ; (iii) les algorithmes s'inspirant du mouvement des bancs de poissons et d'autres animaux aquatiques comme les dauphins ; et (iv) les algorithmes s'inspirant des comportements des micro-organismes tels que les virus, bactéries, algues et autres micro-organismes similaires.

3.6.3 Algorithmes basés sur la physique/chimie

Les algorithmes basés sur la physique et la chimie forment une catégorie distinctive en émulant les comportements des phénomènes physiques et chimiques. Parmi ces phénomènes, on retrouve les forces gravitationnelles, l'électromagnétisme, les charges électriques, le mouvement des particules gazeuses, ainsi que le mouvement de l'eau.

L'origine de cette catégorie remonte à des travaux bien connus du siècle dernier, tels que le recuit simulé, largement appliqué dans différents domaines d'optimisation. L'algorithme de recherche gravitationnelle (GSA) [75], l'optimisation du trou noir (BH) [76], l'algorithme de recherche basé

sur la galaxie (GBSA) [77]. L'harmony search (HS) [78] se distingue également, s'appuyant sur les similitudes entre la composition musicale et les phénomènes physiques pour atteindre des solutions optimales.

3.6.4 Algorithmes basés sur le comportement humain social

Cette catégorie puise son inspiration dans les comportements sociaux humains, en se basant sur des analogies avec la prise de décision, les dynamiques d'expansion et de concurrence idéologique au sein de la société. Elle inclut également des concepts politiques, comme l'algorithme de colonie impérialiste (ICA) [79].

Cette catégorie englobe également des algorithmes qui simulent des compétitions sportives, comme l'algorithme de compétition de la ligue de football (SLC) [80]. Les processus de brainstorming, connus pour stimuler la créativité, ont donné naissance à plusieurs algorithmes prometteurs, tels que brain storm algorithme d'optimisation (BSO) [81] et l'algorithme d'optimisation global-best brain storm (GBSO) [82].

3.6.5 Algorithmes basés sur les plantes

Cette catégorie regroupe principalement tous les algorithmes d'optimisation dont le processus de recherche est inspiré des mécanismes observés chez les plantes. Parmi les plus reconnus, on trouve l'algorithme forest optimization algorithms (FOA) [83], qui puise son inspiration du processus de reproduction des plantes.

3.6.6 Algorithmes d'inspirations diverses

Cette catégorie rassemble les algorithmes qui ne peuvent pas être classés dans les catégories précédentes. Elle représente la diversité des sources d'inspiration existantes dans la littérature scientifique.

Cette catégorie constitue une source d'inspiration féconde pour la communauté scientifique, incitant les chercheurs à explorer de nouvelles idées et à développer des méthodes originales en puisant dans une multitude de domaines, allant des comportements de reproduction naturelle à l'intelligence en essaim, en passant par les phénomènes physiques et chimiques, et bien plus encore. Cette diversité d'approches ouvre des perspectives passionnantes pour l'optimisation et stimule la créativité des chercheurs, en leur offrant une vaste gamme de sources pour développer des solutions novatrices aux problèmes d'optimisation.

Dans la section suivante, nous présentons un état de l'art approfondi, mettant en évidence le potentiel de ces algorithmes et leur capacité à surmonter le problème d'ordonnancement dans les systèmes distribués, en particulier l'ordonnancement des tâches dans le contexte du Cloud Computing.

3.7 Panorama des Approches d'Ordonnancement de Tâches dans le Cloud Computing : État de l'Art

De nombreux chercheurs ont proposé différentes approches basées sur des heuristiques ou des métaheuristiques pour résoudre le problème d'ordonnancement des flux de travail et les tâches/applications indépendantes dans les systèmes distribués. Nous ne faisons pas ici un état de l'art exhaustif, mais nous nous intéressons principalement aux algorithmes proposés récemment et qui sont actuellement répandus dans le Cloud computing.

L'étude menée par Houssein et al. [43] expose une variété de techniques heuristiques qui s'attaquent aux défis sous-jacents de l'ordonnancement des ressources et des tâches. Parmi celles-ci, on trouve les algorithmes Min-Min [84], Max-Min [84], First Come First Serve (FCFS) [85], Shortest Job First [86], Round Robin [87], Heterogeneous Earliest Finish Time (HEFT) [88], Minimum Completion Time (MCT) [89], ainsi que Sufferage [90].

• **Min-Min :** est une heuristique de base dans laquelle elle commence par un ensemble de tâches non planifiées, puis détermine les temps d'exécution minimums pour chaque tâche sur toutes les machines. La tâche dont le temps d'exécution est le plus court est ensuite sélectionnée. Sur la base de cette valeur de mesure, la tâche est programmée sur la machine correspondante. Le temps de disponibilité de cette ressource est mis à jour pour toutes les autres tâches, en ajoutant le temps d'exécution de la tâche assignée au temps d'exécution des autres tâches sur cette machine. La tâche programmée est ensuite retirée de la liste des tâches et la procédure est répétée jusqu'à ce que toutes les tâches non planifiées soient épuisées. Cette approche privilégie la tâche la plus courte, mais les grandes tâches doivent attendre que les plus petites soient exécutées en premier. Le débit global du système est grandement amélioré avec cet algorithme, cependant, des problèmes de famine peuvent survenir avec les tâches lourdes. Des problèmes tels que l'exécution lente des tâches, les difficultés liées aux délais et les problèmes de priorité surgissent également lors de l'exécution de l'algorithme Min-Min [91].

• **Max-Min** : Il suit le même principe que l'algorithme Min-Min, sauf que la tâche la plus longue est choisie en premier et affectée à la VM la plus appropriée qui devrait, parmi toutes les VM disponibles, exécuter la tâche dans le temps d'exécution minimal. Ensuite, le temps de fonctionnement de la machine choisie pour la tâche sélectionnée est mis à jour et la tâche assignée est supprimée de la liste des tâches. Le processus est répété jusqu'à ce que toutes les tâches soient ordonnancées avec succès. Comparé à Min-Min, Max-Min offre un débit plus élevé tout en ayant un makespan plus faible, mais il peut rencontrer le problème de la surutilisation et de la sous-utilisation potentielles des ressources [92].

• **L'algorithme FCFS**, axé sur la planification en temps réel, attribue les tâches en fonction de leur heure d'arrivée, il affecte la tâche à chaque fois qu'elle arrive, aux ressources disponibles. Avec l'algorithme FCFS, la première demande d'arrivée d'une tâche ou d'une ressource est satisfaite en premier, puis la demande suivante dans une file d'attente est exécutée une

fois que celle qui la précède est terminée. Cet algorithme est relativement simple, et ses avantages et inconvénients sont plus évidents. L'avantage est le faible coût, et l'inconvénient de l'utilisation de différents scénarios est que le schéma optimal d'attribution des tâches n'est pas pris en compte, et que davantage de tâches peuvent apparaître dans la file d'attente.

• **HEFT** est fondamentalement une heuristique d'ordonnancement basée sur une liste dans laquelle une liste de priorité de tâches est d'abord établie, afin que les décisions d'allocation optimales soient ensuite prises localement pour chaque tâche en fonction du temps d'exécution estimé de la tâche.

• **RR** se déclenche à l'heure d'arrivée pour allouer immédiatement les ressources disponibles à la tâche entrante ; cependant, les ressources sont allouées à la tâche pendant une certaine durée (quantum de temps). Ensuite, si un temps d'exécution supplémentaire est encore nécessaire, la tâche est préemptée, mise en file d'attente et attend que son exécution reprenne plus tard. Il est plus orienté vers l'équité de l'ordonnancement et peut résoudre le problème d'équilibrage de la charge. Cependant, en même temps, il ignore l'objectif du temps d'exécution et la capacité d'exécution des nœuds de ressources, ce qui entraîne une diminution de l'efficacité globale de l'ordonnancement.

• **L'algorithme MCT** attribue les tâches aux VMs ou aux ressources en fonction du meilleur temps d'achèvement prévisible pour cette tâche dans un ordre aléatoire. Chaque tâche est attribuée à la VM ou à la ressource ayant le temps d'exécution le plus court. Avec l'algorithme MCT, certaines tâches sont attribuées aux VMs ou aux ressources n'ayant pas de temps d'exécution minimum.

• **Sufferage** est une technique heuristique dans laquelle une ressource est immédiatement associée à une tâche qui serait susceptible le plus en fonction d'une valeur seuil de "sufferage" associée à son temps d'achèvement prévu. Dans le cadre de cette technique, le temps d'achèvement de chaque tâche sur chaque ressource est d'abord calculé. Ensuite, la différence entre deux MCT consécutifs pour chaque tâche est calculée et appelée valeur de suffrage. Enfin, une ressource avec le temps d'exécution minimum est attribuée à la tâche ayant la valeur de suffrage maximale, les temps d'exécution de toutes les ressources sont mis à jour, et les étapes ci-dessus sont répétées jusqu'à ce que toutes les tâches soient planifiées avec succès. Malgré ses performances parfaites dans de nombreux cas, cette stratégie présente un inconvénient si plusieurs tâches ont la même valeur de suffrage, où la première tâche entrante est simplement sélectionnée et exécutée en premier sans tenir compte des autres tâches, ce qui peut entraîner un problème de famine.

En outre, de nouvelles approches heuristiques émergent également, comme Load-Balanced Min-Max (LBMM) [93], Task Aware Scheduling Heuristic (TASA) et Resource Aware Scheduling Algorithm (RASA) [94], offrant des alternatives aux problèmes d'ordonnancement. Le Tableau 3.1 présente un résumé de certaines techniques d'heuristiques d'ordonnancement existantes dans la littérature.

TABLE 3.1 – *Résumé de quelques approches heuristiques prometteuses pour l'ordonnancement dans le Cloud Computing*

Contributions	Type de tâche	Avantages	Faiblesses/défis	Simulateur	Année	Réf
Min-Min amélioré basé sur le réordonnancement	Indépendante	• Makespan faible • Utilisation de ressources élevées	• Coût non considéré • Exécution lente des tâches en raison de la replanification	N/A	2019	[91]
Max-Min basé sur la priorité	Indépendante	• Makespan faible • Utilisation de ressources élevées	• Ne pas prendre en compte les tâches dynamiques émergentes • Potentiel élevé de surutilisation/sous-utilisation ressources	CloudSim	2019	[92]
HEFT amélioré	Workflow	• Makespan faible	• Surveillance non continue des ressources • Charge déséquilibrée	CloudSim	2018	[95]
SJF hétérogène dynamique (DHSJF)	Indépendante	• Makespan faible • Consommation d'énergie faible	• Famine potentielle	CloudSim	2019	[96]
RR optimisée multi-objectif basée sur un seuil	Workflow	• Makespan faible • Efficacité élevée • Consommation d'énergie faible	• Charge déséquilibrée	CloudSim	2019	[97]
Algorithme de suffrage basé sur le temps d'achèvement	Indépendante	• Makespan faible • Utilisation de ressources élevées	• Coût non considéré	CloudSim	2018	[98]
Algorithme de suffrage basé sur le temps d'exécution	Indépendante	• Makespan faible • Utilisation de ressources élevées • Charge équilibrée	• Coût et délai non considérés	CloudSim	2019	[90]
SJF et RR avec quantum dynamique	Indépendante	• Débit élevé • Temps de réponse faible	• Famine potentielle • Charge déséquilibrée	CloudSim	2017	[99]
Approche hybride du plus court et du plus long tâche en premier (HSLJF)	Indépendante	• Makespan faible • Utilisation élevée des ressources • Débit élevé	• Coût non considéré • Déséquilibre de la charge	CloudSim	2019	[100]
Hybridation entre FJS et RR (SJFRR)	Indépendante	• Faible temps d'attente • Faible délai moyen d'exécution	• Capacité d'adaptation au changement de contexte	N/A	2020	[101]

Dans l'ensemble, les techniques heuristiques sont fréquemment confrontées à des dilemmes de minima locaux, et c'est là que les algorithmes méta-heuristiques se révèlent essentiels [102, 54]. En fait, ces algorithmes ont fait l'objet d'une évaluation systématique au sein de la communauté de la recherche, suscitant un vif intérêt, notamment dans le contexte des environnements distribués. En tant que solutions éprouvées, les techniques méta-heuristiques sont régulièrement adoptées pour résoudre efficacement le problème d'optimisation étudié dans le cadre de cette thèse. Les auteurs de [103, 7] ont d'ailleurs fourni un aperçu détaillé des différentes métaheuristiques spécialement conçues pour aborder la planification des tâches dans le contexte du Cloud Computing.

Khalili et Babamir [104] ont développé un algorithme basé sur PSO visant à optimiser un objectif unique, à savoir la minimisation du makespan. Leur contribution réside dans l'exploration de différentes stratégies de poids d'inertie, notamment le poids d'inertie décroissant linéairement (LDIW), en combinaison avec l'algorithme PSO. Les résultats obtenus démontrent une amélioration du makespan grâce à l'introduction de la stratégie LDIW.

Dans une perspective similaire, Gabi et al. [105] ont introduit l'optimisation de l'essaim de chats (CSO) comme une approche novatrice pour l'ordonnancement des tâches. Cette méthode intègre l'équation LDIW pour surmonter les optima locaux, un problème courant en recherche locale. L'efficacité de cette technique se manifeste par une convergence plus rapide et une meilleure allocation des tâches aux machines virtuelles, contribuant ainsi à la minimisation du makespan.

Dans une approche visant toujours à minimiser le temps d'achèvement des applications (makespan), Elaziz et al. [106] ont proposé l'algorithme hybride MSDE, qui fusionne deux méta-heuristiques, à savoir l'algorithme de recherche de mites (MSA) et l'algorithme de l'évolution différentielle. L'algorithme MSDE équilibre l'exploration et l'exploitation des solutions en utilisant le concept de vol de Lévy. Il tire parti des capacités d'exploration de MSA, tout en améliorant la capacité d'exploitation par l'intégration de l'algorithme DE pour la recherche locale. Les expériences menées avec CloudSim mettent en évidence la supériorité de l'algorithme MSDE par rapport aux approches heuristiques et méta-heuristiques traditionnelles en termes de makespan et de débit du système Cloud.

Dans [107], Malik et Jain ont présenté une approche d'ordonnancement basée sur l'algorithme de recherche harmonie, visant à minimiser le temps d'achèvement tout en optimisant l'utilisation des ressources pour satisfaire les demandes des utilisateurs.

Matos et al. [108] ont proposé une solution combinant la méta-heuristique GA avec l'algorithme statique Maximum Resource (MR) pour résoudre le problème de partitionnement. Cette fusion a permis de réduire de 20 % le nombre de machines virtuelles requises et d'améliorer le temps de traitement des tâches (makespan). Cependant, certaines limitations se manifestent lorsque la taille des tâches augmente.

Abualigah et al. [109] ont développé une méthode hybride combinant l'optimiseur multivers (MVO) avec l'algorithme génétique dans le but de

minimiser le temps de transfert des tâches en orchestrant efficacement les ressources disponibles. Leur enrichissement du MVO conventionnel par l'intégration de processus de croisement et de mutation renforce la planification des tâches initiée par le MVO.

Attiya et al. [110] ont introduit une combinaison de l'algorithme SA (Simulated Annealing) et de l'algorithme HHO (Harris Hawks Optimization) pour résoudre les problèmes liés à l'optimum local dans la phase d'exploration. L'algorithme résultant, appelé HHOSA, parvient à obtenir une valeur moyenne minimale du makespan par rapport aux algorithmes comparés.

Huang, X. et al. ont présenté une technique d'ordonnancement des tâches basée sur l'optimisation par gradient (GBO) [111]. Cette méthode vise à améliorer les performances des systèmes Cloud en utilisant des ressources informatiques spécifiques. Le schéma d'ordonnancement des tâches est d'abord aligné sur un modèle vectoriel, puis l'algorithme GBO est déployé pour obtenir une solution quasi-optimale. Cette approche a conduit à une amélioration significative du makespan.

La plupart des travaux cités ci-dessus se sont principalement focalisés sur l'optimisation d'une seule métrique de qualité de service, à savoir le makespan. Cependant, avec l'évolution du marché du Cloud Computing, les fournisseurs de services Cloud commerciaux proposent une gamme variée de services et de ressources informatiques, offerts à des tarifs différents en fonction des négociations de niveau de service. Les utilisateurs sont facturés en fonction de leur consommation de ressources, ce qui introduit une dimension économique importante. Prenons, par exemple, Amazon EC2, un fournisseur bien connu d'infrastructures en tant que service (IaaS). Les utilisateurs ont la possibilité de choisir entre des options de tarification pour réduire le temps d'exécution de leurs tâches en payant un peu plus cher ou pour réduire les coûts d'exécution en autorisant un temps d'exécution plus long. En conséquence, les politiques d'ordonnancement doivent impérativement prendre en compte les coûts des ressources, ainsi que d'autres paramètres de qualité de service, pour répondre aux divers besoins des utilisateurs tout en assurant une efficacité globale du système Cloud.

Dans cette perspective, Rekha et Dakshayini [112] ont introduit l'algorithme génétique comme une solution prometteuse pour l'optimisation de l'ordonnancement des tâches, avec pour objectif de réduire le temps d'achèvement. Leurs travaux ont permis d'évaluer l'efficacité de GA en le comparant à des méthodes d'allocation simples, en se concentrant sur des métriques telles que le débit et le makespan.

Dans une optique similaire, les auteurs de [113] ont proposé une modification de l'algorithme génétique appelée MGGS, qui combine GA avec une stratégie gloutonne. Les résultats expérimentaux ont clairement démontré l'efficacité du MGGS dans la résolution des problèmes d'ordonnancement des tâches. En outre, une version hybride innovante de l'algorithme génétique est présentée dans [114], fusionnant GA et PSO dans une méthode appelée GA-PSO. Cette approche permet de minimiser à la fois le makespan et les coûts, offrant ainsi une solution plus complète pour la planification des tâches.

Une version améliorée de PSO est présentée dans le travail de Zhou et al. [115], axée sur la planification des tâches dans les environnements Cloud avec un accent particulier sur la réduction de la consommation d'énergie. L'introduction du coefficient de poids d'inertie dans PSO permet d'améliorer la vitesse de convergence, bien que cette approche ne couvre pas explicitement le makespan et l'utilisation des ressources.

Un mécanisme d'ordonnancement basé sur le recuit simulé et l'ACO est présenté dans [116]. Cette approche combine l'utilisation de SA dans une première phase suivie de l'ACO dans la deuxième phase pour l'ordonnancement des cloudlets. Cette combinaison vise à améliorer significativement les performances du système en tirant parti des avantages de chaque algorithme.

Nasr et al. [117] ont fait progresser la recherche en combinant l'optimisation des réactions chimiques (CRO) et l'algorithme ACO en un algorithme hybride nommé CR-AC. Cette approche novatrice se concentre sur la résolution du problème de planification du flux de travail sous contrainte de délai, dans le but de réduire les coûts monétaires et la complexité temporelle.

Dans une autre étude, Rani et Suri [118] ont fusionné l'algorithme de recherche gravitationnelle et l'algorithme ACO dans un algorithme hybride de planification des tâches visant à résoudre le problème de l'équilibrage de la charge. Cette combinaison a permis d'éviter la convergence prématurée de l'ACO. L'étude considère les machines virtuelles comme des sources potentielles de nourriture et exploite le concept de phéromone pour identifier la meilleure machine virtuelle. Le GSA est utilisé pour mettre à jour la phéromone. Les résultats démontrent une répartition équilibrée de la charge, réduisant ainsi la surcharge des machines virtuelles, améliorant le temps de complétion des tâches et réduisant la consommation de ressources.

Chaudhary et Kumar [119] ont développé une technique d'ordonnancement de la charge de travail, appelée Hybrid Genetic-GSA (HG-GSA), visant à réduire le coût total du calcul, y compris les coûts d'exécution et de transfert, tout en maximisant l'utilisation des ressources disponibles.

Pirozmand et al. [5] ont également contribué à l'avancement de la recherche en proposant un algorithme hybride appelé GSAGA, qui combine l'algorithme de recherche par gravité et l'algorithme génétique pour optimiser à la fois le makespan et le coût de l'ordonnancement des tâches.

Chen et al. [120] ont appliqué la méta-heuristique d'optimisation des baleines (WOA) récemment introduite, à la planification des tâches dans les environnements de Cloud Computing, en utilisant un modèle d'optimisation multi-objectif. Pour pousser plus loin les capacités de recherche de WOA, ils ont proposé une approche avancée appelée Improved WOA for Cloud (IWC). L'IWC a démontré une précision accrue et une convergence plus rapide par rapport aux algorithmes méta-heuristiques existants. Il s'est révélé efficace pour la recherche de plans d'ordonnancement optimaux, améliorant ainsi l'efficacité globale du système en gérant efficacement la charge du système et en optimisant l'utilisation des ressources pour les tâches de petite et de grande envergure. Toutefois, des opportu-

nités d'amélioration subsistent pour équilibrer davantage l'exploration et l'exploitation dans l'IWC.

Han et al. [121] ont présenté l'algorithme HDEA (Hybrid Differential Evolution Algorithm), qui repose principalement sur l'utilisation de l'algorithme méta-heuristique DE, auquel sont intégrées plusieurs politiques d'optimisation. L'objectif principal de HDEA est d'optimiser la planification des tâches dans les environnements Cloud, en prenant en compte à la fois le coût monétaire et le délai d'exécution. Cette approche se distingue par deux méthodes distinctes pour générer la population initiale, en adoptant une stratégie d'ajustement des paramètres et en introduisant une nouvelle stratégie de mutation. De plus, plusieurs techniques de recherche locale sont appliquées pour obtenir des solutions de meilleure qualité.

El-Ashmawi et Ali [122] ont développé un algorithme d'essaim de salp (SSA) modifié pour l'affectation des tâches, réduisant efficacement les coûts de communication et d'exécution. Cette approche a également trouvé des applications dans les domaines de l'ingénierie et des sciences. Natesan et al. [123] ont introduit l'algorithme d'optimisation du loup gris (GWO) pour l'optimisation de l'allocation des tâches et des ressources dans les environnements du Cloud. Dans [124], un algorithme d'ordonnancement hybride a été développé en combinant l'algorithme compétitif impérialiste (ICA) avec l'algorithme firefly dans le but d'améliorer à la fois le makespan et l'équilibrage de la charge.

Milan et al. [125] ont proposé un algorithme d'ordonnancement basé sur BFO (Bacterial Foraging Optimization) pour réduire le temps d'inactivité des machines virtuelles. Le modèle présenté utilise chaque bactérie générée par l'algorithme glouton pour représenter une solution potentielle, avec une indication de la meilleure position et du coût associé à cette bactérie. L'utilisation des étapes de culbute et de déplacement dans le processus de recherche de nourriture a permis de réduire de manière significative le makespan et la consommation d'énergie.

Madni et al. ont apporté une contribution significative en introduisant l'algorithme Multi-Objective CSO (MOCSO) [52], conçu pour résoudre les problèmes d'ordonnancement des ressources dans le contexte de l'infrastructure en tant que service. Cet algorithme vise à réduire les coûts pour les utilisateurs Cloud tout en minimisant le makespan, ce qui contribue à accroître les profits et les revenus des fournisseurs de services Cloud tout en optimisant l'utilisation des ressources.

Sha Alam Khan et Santosh [126] ont présenté une méthode d'ordonnancement des tâches qui prend en compte divers paramètres, notamment le temps d'attente minimal, le temps d'exécution, l'efficacité, et l'utilisation des ressources, parmi d'autres. Les performances de cette méthode d'ordonnancement ont surpassé celles d'autres algorithmes bien connus, tels que ACO et PSO.

Pour une vue générale, le Tableau 3.2 présente une comparaison des diverses méthodes d'ordonnancement qui s'appuient sur les métaheuristiques bio-inspirées mentionnées précédemment, en fonction des objectifs d'ordonnancement.

TABLE 3.2 – *Une comparaison des approches d'ordonnancement présentées en fonction de divers objectifs d'optimisation de l'ordonnancement.*

Réference & Année	Type de tâche	Makespan	Coût	Consommation d'énergie	Débit	Équilibrage de charge	Utilisation de ressources	Fiabilité	Simulateur
[104], 2015	Independente	√	-	-	-	-	-	-	CloudSim
[105], 2019	Independente	√	-	-	-	-	-	-	CloudSim
[106], 2019	Independent	√	-	-	-	-	-	-	CloudSim
[107], 2016	Independente	√	-	-	-	-	-	-	CloudSim
[108], 2019	Workflow	√	-	-	-	-	-	-	CloudSim
[109], 2022	Indépendante	√	-	-	-	-	-	-	MATLAB
[110], 2020	Independent	√	-	-	-	-	-	-	CloudSim
[111], 2022	Indépendante	√	-	-	-	-	-	-	MATLAB
[112], 2019	Independent	√	-	-	√	-	-	-	CloudSim
[113], 2020	Independent	√	-	-	√	-	-	-	CloudSim
[114], 2018	Workflow	√	√	-	√	-	-	-	WorkflowSim
[115], 2018	Workflow	-	√	√	-	-	-	-	CloudSim
[116], 2019	Independent	√	-	-	-	√	-	-	CloudSim
[117], 2019	Workflow	√	√	-	-	-	-	-	CloudSim
[118], 2020	Independent	√	-	-	√	√	√	-	CloudSim
[119], 2019	Independent	-	√	-	-	-	√	-	CloudSim
[5], 2022	Workflow	√	√	-	-	-	-	-	CloudSim
[120], 2020	Independent	-	√	-	-	-	√	-	MATLAB
[121], 2018	Independent	√	√	-	-	-	-	-	CloudSim
[122], 2020	Workflow	-	√	-	-	-	-	-	N/A
[123], 2019	Independent	√	√	-	-	-	-	-	CloudSim
[124], 2020	Workflow	√	-	-	-	√	-	-	N/A
[125], 2020	Independent	√	-	√	-	-	-	-	CloudSim
[52], 2019	Workflow	√	√	-	-	-	√	-	CloudSim
[126], 2022	Independent	√	-	-	-	-	√	-	N/A

3.8 Discussion

Il est important de noter que, en raison de la profusion de contenu dans la littérature portant sur les approches d'ordonnancement des charges de travail dans le Cloud, il est difficile de couvrir toutes les méthodes. Néanmoins, à la lumière de la section précédente, nous ouvrons une discussion systématique sur les orientations potentielles de la recherche d'ordonnancement des tâches. Cette démarche vise à fournir une meilleure compréhension de la situation actuelle, les tendances futures de la recherche, les priorités et les points cruciaux de cette discipline.

• Bien que les heuristiques soient obsolètes, elles sont toujours prises en considération par de nombreux chercheurs. Comme le montre le tableau 3.1, les algorithmes heuristiques ont révélé des performances significatives sur la base de diverses approches d'ordonnancement abordées dans la littérature. De plus, il est remarquable qu'il n'y ait pas d'approche heuristique prépondérante dans toutes les études sélectionnées, car chacune des approches concernées est adaptable et peut exceller en fonction de la nature du problème d'ordonnancement traité. Il est encore plus remarquable qu'aucun algorithme heuristique n'ait surpassé les métaheuristiques considérées, ce qui renforce notre choix judicieux de mettre l'accent sur les méta-heuristiques dans notre étude.

• Plusieurs travaux ont déployés des algorithmes d'ordonnancement basées sur les algorithme évolutionnaires, l'intelligence en essaim, la physique/chimie, voire même les comportements sociaux humains. Toutefois, peu de travaux se sont penchés sur les algorithmes inspirés des mécanismes observés chez les plantes. Pour répondre à ce manque, nous nous appuyons dans nos contributions sur un algorithme de pollinisation des fleurs bio-inspiré, s'inspirant du processus naturel de pollinisation des plantes à fleurs.

• En outre, la plupart de travaux présentés mettant l'accent sur une ou deux métriques de QoS à savoir le makespan et le coût. De plus, peu d'efforts ont été consacrés à prendre en compte la fiabilité des services et la consommation d'énergie. Nos approches visent à combler cette lacune en considérant un ensemble plus complet de métriques QoS, englobant l'ensemble des aspects essentiels de la performance.

• Un autre constat concerne l'approche multi-objectif utilisée. La plupart des travaux présentés sont mono-objectif ou basés sur une agrégation des objectifs. La plupart ne prêtent pas attention à la façon dont chacune des métriques de QoS peut affecter les autres. Dans notre travail, nous avons pris en compte cet aspect, en comparant l'approches d'ordonnancement basées sur la méthode Pareto et celles basées sur l'agrégation. Nous nous sommes basés donc, à cette fin, sur l'optimisation multi-objectif à travers l'algorithmes de pollinisation des fleurs, permettant ainsi d'optimiser simultanément le makespan, le coût et la fiabilité des ressources IaaS Cloud.

• Il convient également de noter que, malgré les améliorations prometteuses apportées par certains algorithmes métaheuristiques dans la recherche de solutions optimales ou relativement optimales pour le problème d'ordonnancement des tâches dans le Cloud, ils présentent souvent

des limites, notamment une convergence prématurée et des difficultés à surmonter les minima locaux, en particulier lorsqu'ils sont confrontés à un grand espace de solution [4]. Ces limitations peuvent entraîner des solutions sous-optimales pour la planification des tâches, affectant ainsi les performances du système et compromettant les garanties de qualité de service. Ces constats soulignent la nécessité impérieuse de développer de nouvelles méthodes adaptatives et efficaces pour résoudre ces problèmes.

3.9 Conclusion

Dans ce chapitre, nous avons d'abord introduit le cadre général des méthodes de résolution des problèmes d'optimisation, qui servira de base pour résoudre le problème d'ordonnancement des tâches. Ensuite, nous avons présenté un état de l'art des approches d'ordonnancement des applications dans le Cloud, mettant en évidence les défis qui sous-tendent cette thèse en décrivant les principales lacunes des travaux de la littérature.

Les chapitres suivants seront consacrés à nos contributions pour l'ordonnancement multi-objectif des tâches dans l'infrastructure IaaS.

4 Ordonnancement multi-objectif basé sur l'algorithme de pollinisation des fleur et l'approche Pareto-TOPSIS

Sommaire

4.1 Introduction

Depuis l'apparition des métaheuristiques jusqu'à nos jours, ces algorithmes font preuve d'un fort potentiel dans la résolution des problèmes d'optimisation complexes, surpassant même les performances des méthodes classiques. Leur utilisation dans le contexte des systèmes Cloud pourrait constituer des opportunités intéressantes et jusqu'ici sous-exploitées pour répondre efficacement aux besoins de qualité de service des utilisateurs.

Dans cette perspective, notre but consiste à proposer une nouvelle approche adaptée à partir des métaheuristiques pour résoudre le problème d'optimisation de l'ordonnancement multi-objectif. Malgré l'abondance de la recherche existante, des questions pertinentes subsistent. En effet, la plupart des études antérieures se concentrent sur un seul objectif, et celles qui abordent un ensemble d'objectifs utilisent souvent une fonction de compromis simple sans tenir compte de la manière dont chaque paramètre pourrait influencer les autres.

Dans ce contexte, le présent chapitre a pour objectif de mettre en avant nos contributions décrites dans [127], qui représentent des avancées significatives dans le domaine de l'ordonnancement multi-objectif de tâches au sein des systèmes cloud hétérogènes. Nos principales contributions englobent une approche novatrice basée sur la pollinisation bio-inspirée des fleurs pour la conception d'un algorithme d'ordonnancement efficace, ainsi que l'introduction d'une méthode d'évaluation multi-objectifs pour la fonction de fitness. Cette dernière combine le principe de l'optimalité de Pareto avec la technique TOPSIS, permettant la sélection judicieuse de la solution la plus pertinente en réponse aux besoins des utilisateurs.

En outre, notre étude aborde à la fois l'optimisation de l'ordonnancement de tâches à objectif unique et multiple. Dans le premier cas, nous formulons une optimisation à objectif unique dans laquelle le makespan est le seul critère pris en compte. Dans la seconde variante, relevant de l'optimisation multi-critère, nous cherchons à équilibrer la relation entre le coût, la fiabilité et l'optimisation du makespan, tout en évaluant leur impact sur les performances globales de l'algorithme que nous proposons. En suivant le fil de ce chapitre, nous examinerons en premier temps les travaux connexes, puis nous présenterons notre approche et son adaptation à notre problématique. Ensuite, nous présenterons les simulations effectuées et les résultats obtenus afin de valoriser l'apport des contributions de l'approche proposée.

4.2 Travaux connexes

L'ordonnancement des tâches dans les systèmes parallèles et distribués est un domaine de recherche d'une importance cruciale. Ce domaine a suscité de nombreuses approches innovantes dans la littérature, chacune avec ses propres hypothèses sous-jacentes et critères de qualité de service pour les utilisateurs finaux. De plus, la plupart des algorithmes d'optimisation multiobjectif existants transforment ce problème en un problème

mono-objectif en utilisant une simple approche de somme pondérée pour évaluer les performances de leur algorithme proposé.

Bezdan et al. [128] ont élaboré un algorithme d'ordonnancement de tâches indépendantes appelé BA-ABC. Il s'appuie sur deux approches métaheuristiques, à savoir l'algorithme de chauve-souris (BA) et celui de colonie d'abeilles. L'objectif est d'assigner les tâches soumises aux types d'instances virtuelles en utilisant l'optimisation hybride de l'algorithme de chauve-souris. Cette approche améliore la phase d'exploration de BA grâce à l'introduction de la recherche d'abeilles observatrices de l'algorithme de colonie d'abeilles artificielles. L'approche BA-ABC adopte une méthodologie d'agrégation pour optimiser simultanément le makespan et le coût d'exécution lors du processus de planification. L'efficacité de BA-ABC a été comparée à celle de quatre autres algorithmes d'optimisation, et les résultats obtenus montrent que la proposition des auteurs présente un avantage relatif par rapport aux algorithmes comparés.

Dans le travail de Sardaraz et al. [129], un algorithme hybride basé sur l'optimisation par essaim de particules est proposé pour la planification des flux de travail scientifiques. L'idée principale de cet algorithme est de prioriser initialement les tâches dans la liste d'attente afin de réduire le temps d'exécution, puis d'optimiser à la fois le makespan et le coût d'exécution dans une seconde étape. L'algorithme surveille également l'équilibrage de la charge pour une utilisation efficace des ressources Cloud. Les performances de l'algorithme proposé sont validées par des simulations numériques comparatives avec le PSO standard, l'algorithme génétique et des approches de planification spécialisées basées sur la technique PSO, telles que PSO-DS [130] et GA-PSO [114].

Adhikari et al. [131] ont développé une stratégie dynamique en deux phases basée sur l'algorithme firefly pour l'ordonnancement des flux de travail. Sur la base de l'approche d'agrégation, plusieurs objectifs contradictoires dans le Cloud IaaS ont été pris en compte, notamment la charge du serveur Cloud, le makespan, l'utilisation des ressources et la fiabilité. La première phase vise à trouver le serveur optimal pour chaque flux de travail tout en équilibrant les charges et l'utilisation des ressources. La seconde phase utilise une affectation de tâches basée sur des politiques pour réduire le makespan du workflow et augmenter la fiabilité des serveurs Cloud.

Dans un récent article, Pirozmand et al. [132] ont introduit un nouvel algorithme hybride nommé GAECS, qui combine l'algorithme génétique avec le modèle Energy-Conscious Scheduling (ECS). Cette approche novatrice tient compte du temps et de la consommation d'énergie dans l'ordonnancement multi-objectif des tâches dans les systèmes de Cloud Computing. L'objectif principal des auteurs est d'attribuer des tâches à des processeurs permettant d'obtenir un meilleur compromis entre le makespan et la consommation d'énergie. Les performances de l'algorithme GAECS proposé sont analysées à l'aide de MATLAB, mettant en évidence qu'il surpasse les autres algorithmes comparés en termes de précision.

Dans [133], les auteurs ont développé un algorithme évolutionnaire multi-objectif appelé Many-Objective Genetic Algorithm Scheduler (MOGAS) pour résoudre le problème d'ordonnancement des conteneurs. Ce

travail s'appuie sur l'algorithme Non-dominated Sorting Genetic Algorithm III (NSGA-III) [134]. L'ordonnanceur proposé distribue un lot de tâches différentes à un groupe hétérogène de nœuds. Divers paramètres, tels que la disponibilité, l'équilibrage de la charge, l'utilisation des ressources, l'énergie et le nombre maximal de tâches attribuées, sont pris en compte. Les résultats des expérimentations confirment les mérites de l'algorithme proposé par rapport à un ordonnanceur basé sur l'optimisation par colonies de fourmis.

Chakravarthi et Shyamala [135] ont apporté une nouvelle contribution en proposant une approche appelée TOPSIS inspired Budget and Deadline Aware Multi-Workflow Scheduling (T-BDMWS) pour la planification de workflows concurrents dynamiques dans des environnements de Cloud Computing. L'objectif est de minimiser le coût d'exécution, le makespan et d'améliorer l'utilisation des ressources des machines virtuelles, tout en respectant les contraintes de délai et de budget spécifiées par l'utilisateur. Cette méthode utilise une somme pondérée du coût, du makespan et du temps de transfert de données pour déterminer la meilleure ressource parmi celles disponibles en fonction des besoins de chaque tâche. Les simulations réalisées avec l'outil CloudSim démontrent que la proposition des auteurs offre des résultats supérieurs en termes de makespan et de coûts par rapport à quatre algorithmes bien connus, à savoir Cloud-based Workflow Scheduling Algorithm (CWSA), Budget and Deadline Constraint Heterogeneous Earliest Finish Time (BDHEFT) et Budget-Heterogeneous Earliest Finish Time (BHEFT).

Dans le travail de Medara et Singh [136], un algorithme d'ordonnancement des flux de travail économe en énergie et axé sur la fiabilité dans un environnement de Cloud Computing est présenté. Cette approche optimise la fiabilité des workflows de tâches tout en réduisant la consommation d'énergie. Les performances de cette méthode ont été évaluées en utilisant deux charges de travail scientifiques réelles, Montage et CyberShake. Les résultats numériques démontrent que cette approche surpasse les approches concurrentes, à savoir HEFT [137], EES [138], et REEWS [47].

En modélisant le Cloud IaaS et les flux de travail des tâches, Han et al. [139] ont élaboré une heuristique efficace nommée CMSWC (Cost and Makespan Scheduling of Workflows in the Cloud) qui vise à minimiser simultanément le coût d'exécution et le makespan des flux de travail. L'algorithme CMSWC opère en deux phases distinctes : le classement et la mise en correspondance. Cette dernière phase est conçue pour éviter d'explorer des ressources inutiles pour les tâches, ce qui réduit significativement l'espace de recherche. Elle utilise une politique de sélection de ressources efficace et une stratégie optimisée de sélection de solutions basée sur une combinaison de deux approches : l'approche de tri rapide non dominée et le Shift-Based Density Estimation (SDE) afin de rapprocher les solutions du front de Pareto. Des expérimentations approfondies sur des flux de travail réels démontrent que cette stratégie présente de meilleures performances en termes de compromis entre makespan et coût par rapport aux approches concurrentes : FDHEFT [140], NSGA-II [141] et MODE [142] dans tous les scénarios testés.

Une autre approche innovante [143] repose sur l'algorithme de l'optimiseur baleine, qui s'inspire du comportement social des baleines à bosse. L'objectif de cette méthode est d'améliorer les contraintes de l'ordonnancement des flux de travail tout en équilibrant la charge entre les ressources utilisées. Les performances du WOA ont été évaluées en tant que problème d'optimisation multi-objectif, en prenant en compte le makespan, l'utilisation des ressources, ainsi que les contraintes de délais. Les résultats montrent que la proposition des auteurs surpasse d'autres techniques existantes telles que le GWO, le PSO, l'ACO et le AG.

Abualigah et Diabat [144] ont introduit un nouvel ordonnancement multi-objectif des tâches qui utilise la méthode d'optimisation AntLion (MALO). Cette approche combine une technique de recherche locale et la stratégie d'évolution différentielle pour améliorer l'exploitabilité de l'ALO et éviter les optima locaux. Les résultats pratiques indiquent une amélioration significative non seulement en termes d'utilisation des ressources et de makespan, mais également en termes de vitesse de convergence du processus d'optimisation.

Pour faire face à l'hétérogénéité et à la dynamique des systèmes Cloud, une nouvelle approche basée sur une technique existante appelée shortest job first est présentée dans [96]. Les tâches sont planifiées pour minimiser la consommation d'énergie, l'utilisation du CPU et le makespan. Les expériences ont révélé que l'approche proposée (DHSFJ) surpasse d'autres techniques telles que le FCFS, le SJF, et sa variante Heterogeneous SJF (HSJF). Ces améliorations proviennent de la prise en compte de l'hétérogénéité à la fois des ressources et de la charge de travail.

Menouer et Darmon [145] ont élaboré une nouvelle méthode d'ordonnancement pour Docker Swarm en combinant les principes de répartition (Spread) et de Bin Packing avec la technique TOPSIS. L'objectif de cette stratégie est de sélectionner, parmi un ensemble de nœuds formant une infrastructure Cloud, le nœud le plus approprié pour exécuter chaque conteneur des tâches soumises par les utilisateurs, tout en réalisant un compromis optimal entre trois critères : le nombre de conteneurs exécutés, le nombre de CPU disponibles et la taille de la mémoire disponible.

Dans [146], un algorithme d'ordonnancement hybride bi-objectif combinant la recherche du coucou (CS pour Cuckoo Search) et PSO, est proposé. Cette méthode vise à réduire à la fois le coût et la valeur du makespan pour générer un mapping tâche-VM dans un environnement Cloud hétérogène. Les simulations ont montré que cette approche améliore non seulement le makespan et le coût, mais également le taux de violation des délais.

Samriya and Kumar [147] ont mis l'accent sur l'énergie, les coûts de migration et l'utilisation éminente des ressources dans les systèmes Cloud. Ils ont introduit une hybridation de PSO et de Fuzzy TOPSIS pour une planification efficace des tâches. Cette approche commence par optimiser la disponibilité des tâches et le nombre de VM à l'aide de l'algorithme PSO. Ensuite, le Fuzzy TOPSIS résout le problème de planification multi-objectif en utilisant une méthode de somme pondérée pour l'énergie, le coût et le temps d'exécution en tant que fonctions objectif. Les résultats des expérimentations ont confirmé que l'approche d'ordonnancement des auteurs

offre une meilleure qualité de service par rapport à d'autres algorithmes tels que PSO et GA.

Ces dernières années, l'algorithme de pollinisation des fleurs a attiré l'attention des chercheurs dans divers domaines, notamment l'ingénierie [148], la géologie [149], les industries pour le contrôle des processus [150], le secteur médical [151], et bien d'autres domaines. Par exemple, Gupta et al. [152] ont introduit une approche d'optimisation bi-objectif pour l'ordonnancement indépendant de tâches sur des ressources virtuelles dans le Cloud en utilisant l'algorithme FPA. Cette méthode a été comparée à trois autres approches métaheuristiques : le PSO, le GA et le GSA. Elle se distingue par un schéma de représentation du pollen efficace et un processus dédié pour déterminer la correspondance entre les tâches et les machines virtuelles, optimisant ainsi le makespan et l'utilisation moyenne des ressources Cloud. Les résultats de la simulation indiquent que l'ordonnancement des tâches basé sur le FPA surpasse les approches métaheuristiques concurrentes. Cependant, le travail proposé manque de dynamisme car il ne traite que de l'ordonnancement statiques des tâches indépendantes et des machines virtuelles.

Dans [153], les auteurs ont utilisé l'algorithme de pollinisation des fleurs pour traiter l'ordonnancement de tâches indépendantes dans les systèmes Cloud. La méthode, nommée Exploration-Enhanced FPA (EEFPA), se concentre sur le maintien de la qualité de service en optimisant que le makespan temporel. Cette recherche met en évidence le fait que l'approche FPA peut avoir des difficultés à explorer efficacement l'espace de recherche au départ en raison de sa faible puissance d'exploration. Pour remédier cette limitation, les auteurs suggèrent une stratégie consistant à éliminer les individus les moins performants de la population et à les remplacer par de nouvelles solutions aléatoires au cours des premiers 30 % d'itérations. Comparée à d'autres méthodes, cette approche démontre sa capacité à réduire le makespan et d'offrir une meilleure vitesse de convergence.

Récemment, Walia et al. [154] ont présenté un algorithme de planification économe en énergie qui repose à la fois sur les algorithmes FPA et GA. L'objectif est de répartir les ressources entre les tâches avec moins d'énergie. L'étude de ces auteurs prend en compte l'utilisation des ressources, le temps d'achèvement, la consommation d'énergie et le coût du calcul comme métriques de performance pour les environnements Cloud homogènes et hétérogènes. La simulation effectuée à l'aide de l'outil ASP.NET a montré que l'algorithme proposé est capable de générer un ordonnancement efficace des tâches et une meilleure gestion des ressources par rapport aux algorithmes standards existants tels que GA et FPA. Cependant, la valeur de l'indice d'équité est inférieure au seuil prescrit, ce qui indique un manque d'équité dans l'allocation des ressources.

À notre connaissance, aucune des recherches précédemment citées n'a pris en compte l'optimisation tridimensionnelle (makespan, coût, fiabilité) en utilisant le schéma de pollinisation des fleurs en combinaison avec l'application simultanée de l'optimalité de Pareto et de la technique multicritère TOPSIS, comme cela est exploré dans ce chapitre. L'objectif principal est de tirer parti de chaque approche pour améliorer de manière significa-

tive les performances globales du processus d'ordonnancement des tâches dans les environnements de Cloud Computing, en cherchant à atteindre un compromis optimal entre le coût, le makespan et la fiabilité.

4.3 Présentation de l'approche adoptée

Rappelons que notre objectif principal est d'assigner des tâches aux machines virtuelles afin de satisfaire un ou plusieurs objectifs d'optimisation de QoS. À cette fin, nous avons considéré deux problèmes d'optimisation distincts. Tout d'abord, nous nous concentrons sur un problème à objectif unique où nous cherchons à minimiser le makespan. Nous comparons notre algorithme à trois algorithmes bien connus pour évaluer ses performances : Round Robin, Max-Min et Min-Min.

Dans le second cas, nous abordons l'optimisation multi-objectif de l'ordonnancement dans des environnements Cloud. À cette fin, nous utilisons un modèle bio-inspiré basé sur le comportement de pollinisation des fleurs pour concevoir un algorithme d'ordonnancement de tâches efficace [155, 156]. Cette technique a été choisie en raison de son efficacité, avec un taux de convergence élevé par rapport à d'autres approches méta-heuristiques telles que GA et PSO, qui sont largement reconnues comme des références majeures dans le domaine de l'optimisation [157].

4.3.1 Algorithme de pollinisation des fleurs

En 2012, Xin-She Yang a proposé un algorithme d'optimisation inspiré de la pollinisation des plantes, appelé FPA (Flower Pollination Algorithm) [157]. Cette méthode s'inspire des caractéristiques de la pollinisation observées chez les plantes à fleurs.

Pour comprendre le FPA, il est essentiel de saisir les deux formes de pollinisation : la pollinisation biotique et la pollinisation abiotique, qui dépendent des mécanismes de transfert du pollen. Environ 90% des plantes à fleurs relèvent de la pollinisation biotique, où le pollen est transféré par un pollinisateur spécifique tel que des insectes, des chauves-souris, des oiseaux ou d'autres animaux. La seconde forme, la pollinisation abiotique, est moins courante car elle ne nécessite pas d'organisme spécifique. Le vent, la gravité ou la diffusion dans l'eau sont responsables du transfert du pollen, et l'herbe en est le bon exemple de cette catégorie.

De plus, la pollinisation peut également être soit auto-pollinisée (pollinisation locale) soit croisée (pollinisation globale). La première se produit lorsque le pollen d'une fleur féconde la même fleur ou d'autres fleurs de la même plante, généralement grâce à des facteurs environnementaux. La pollinisation croisée, quant à elle, se produit sur de longues distances lorsque le pollen d'une fleur est apporté par un pollinisateur depuis une autre plante.

D'un point de vue de l'évolution biologique, le but ultime de la pollinisation des fleurs est d'assurer la reproduction optimale des plantes en favorisant la survie des fleurs les mieux adaptées dans le règne végétal [157].

Au cours du processus d'optimisation, l'exploration de l'espace de recherche de l'algorithme FPA, que nous étudions dans ce travail, se fait par pollinisation biotique et croisée où le mouvement du pollen est représenté par le processus de vol stochastique markovien de Lévy [158]. Ce dernier est une marche aléatoire entrecoupée de longs sauts à partir de sa position actuelle selon une loi de puissance, basée sur un pas aléatoire de la distribution de Lévy pour imiter efficacement la caractéristique des déplacements à longue distance des insectes. Par conséquent, nous pouvons idéaliser les caractéristiques du schéma de pollinisation, la constance des fleurs et le comportement des pollinisateurs sur la base des quatre règles principales énumérées ci-dessous :

- **R1** : La pollinisation biotique et croisée agit comme un processus de pollinisation global via le vol de Lévy.
- **R2** : La pollinisation abiotique et l'auto-pollinisation sont considérées comme des formes de pollinisation locales.
- **R3** : La constance des fleurs peut être impliquée en raison de la similarité entre deux fleurs.
- **R4** : La pollinisation locale (exploitation) et la pollinisation globale (exploration) sont contrôlées par une probabilité de commutation $p \in [0, 1]$.

En se basant sur ces quatre règles, les principales étapes de l'algorithme FPA standard sont présentées dans l'Algorithme 1.

Algorithm 1 Pseudo-code standard de l'algorithme FPA

Fonction objectif min ou max $f(x), x = (x_1, x_2, ...)$
Définir une probabilité de commutation $p \in [0, 1]$
Générer initialement une population de fleurs de manière aléatoire
Trouver la meilleure solution g_* dans la population initiale
while critère d'arrêt **do**
  for each $i = 1 : n$ (toutes les n fleurs de la population) **do**
    if $rand() < p$ **then**
      Tirer une taille de pas L selon une distribution de Lévy
      Pollinisation globale via $x_i^{t+1} = x_i^t + L\left(x_i^t - g_*\right)$
    else
      Tirer ε à partir d'une distribution uniforme dans l'intervalle $[0, 1]$.
      Sélectionnez x_j^t et x_k^t aléatoirement parmi toutes les solutions
      Pollinisation locale via $x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right)$
    end if
    Évaluer la nouvelle solution
    if la nouvelle solution est meilleure **then**
      Remplacer x_i^t par x_i^{t+1}
    end if
  end for
  Trouver la meilleure solution actuelle g_*
end while

L'algorithme commence par générer aléatoirement la population initiale, qui sera évaluée pour déterminer la meilleure solution actuelle. Pour calculer une nouvelle solution, le type de pollinisation doit d'abord être défini selon une probabilité prédéterminée (R4). En d'autres termes, un nombre aléatoire est généré, et s'il est inférieur à p, alors la pollinisation globale et la constance des fleurs (R1 et R3) peuvent être appliquées comme suit :

$$x_i^{t+1} = x_i^t + L\left(x_i^t - g_*\right) \tag{4.1}$$

où x_i^t est une solution i à l'itération t, x_i^{t+1} est le vecteur de solution généré à l'itération $t+1$, g_* est la meilleure solution actuelle trouvée parmi toutes les solutions de la génération. Le paramètre L est une taille de pas basée sur le principe des vols de Lévy. Comme les insectes peuvent se déplacer sur une longue distance avec différents pas de distance, ceci est modélisé en utilisant une distribution de Lévy [158] selon l'équation suivante :

$$L \sim \frac{\lambda \Gamma(\lambda) \sin\left(\frac{\pi\lambda}{2}\right)}{\pi} \frac{1}{s^{1+\lambda}}, (s > 0) \tag{4.2}$$

Dans l'équation ci-dessus, $\Gamma(\lambda)$ représente la fonction gamma standard, et cette distribution est valide pour les grands pas $s > 0$ qui sont générés à l'aide de la formule suivante :

$$s = \frac{U}{|V|^{1/\lambda}} \tag{4.3}$$

où U et V représentent deux échantillons aléatoires issus d'une distribution gaussienne normale avec une moyenne de zéro et des écarts-types de σ_u et σ_v :

$$U \sim \left(0, \sigma_u^2\right), V \sim \left(0, \sigma_v^2\right) \tag{4.4}$$

$$\sigma_u = \left[\frac{\Gamma(1+\lambda)}{\lambda \Gamma\left((1+\lambda)/2\right)} \frac{\sin\left(\frac{\pi\lambda}{2}\right)}{2^{(\lambda-1)/2}}\right]^{1/\lambda}, \sigma_v = 1 \tag{4.5}$$

Sinon, si le nombre aléatoire généré est supérieur à p, la pollinisation locale et la constance des fleurs (R2 et R3) sont effectuées comme suit :

$$x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right) \tag{4.6}$$

où, x_j^t et x_k^t représentent deux solutions différentes sélectionnées de manière aléatoire, et $\varepsilon \in [0,1]$ est un nombre aléatoire utilisé pour rendre cette sélection plus proche d'une marche aléatoire locale [157]. Ensuite, une nouvelle solution sera évaluée en fonction de sa performance (fonction objectif). La nouvelle génération sera également évaluée pour sélectionner la plus prometteuse, et le processus de recherche sera répété jusqu'à ce que le critère d'arrêt soit atteint.

4.3.2 Pareto optimalité

L'optimisation multi-objectif est un domaine exigeant la spécification préalable de l'importance relative de chaque objectif, ce qui suppose généralement une connaissance préalable des solutions potentielles. Cependant, le concept de Pareto offre une solution à cette contrainte en éliminant la nécessité de connaître à l'avance les solutions possibles. Cette caractéristique particulière du concept de Pareto est l'une des raisons principales de sa popularité dans le domaine de l'optimisation [159].

Dans ce qui suit, nous allons définir en détail les concepts liés à l'optimisation multi-objectif basée sur le principe de Pareto.

- **Dominance** : Dans le contexte d'un problème de minimisation, une solution X domine une solution Y si, $\forall i = 1, 2, ..., k.\ f_i(X) \leq f_i(Y) \wedge \exists k \in \{1, 2, ..., k\}$ tel que $f_k(X) < f_k(Y)$.
- **Solutions Pareto-Optimales** : Une solution X est définie comme Pareto-optimale si elle n'est dominée par aucune autre solution réalisable. Les solutions Pareto-optimales, également connues sous le nom de solutions non-dominées, sont celles qui ne s'influencent pas mutuellement. Autrement dit, il est impossible de trouver une solution qui améliore la valeur d'un objectif sans détériorer simultanément d'autres objectifs. Cette notion se formule comme suit :

$$P^* = \{x \in X \ / \ \nexists\, x' \in X, x' < x\} \tag{4.7}$$

- **Front de Pareto** : La relation de dominance de Pareto définit un ordre partiel dans le contexte de l'optimisation multi-objectif. En conséquence, la solution à un problème d'optimisation multi-objectif est représentée par un ensemble de points non divisibles. Cet ensemble est désigné sous le nom de "front de Pareto", comme illustré dans la Figure 4.1. Sa définition est la suivante :

$$PF^* = F(x), x \in P^* \tag{4.8}$$

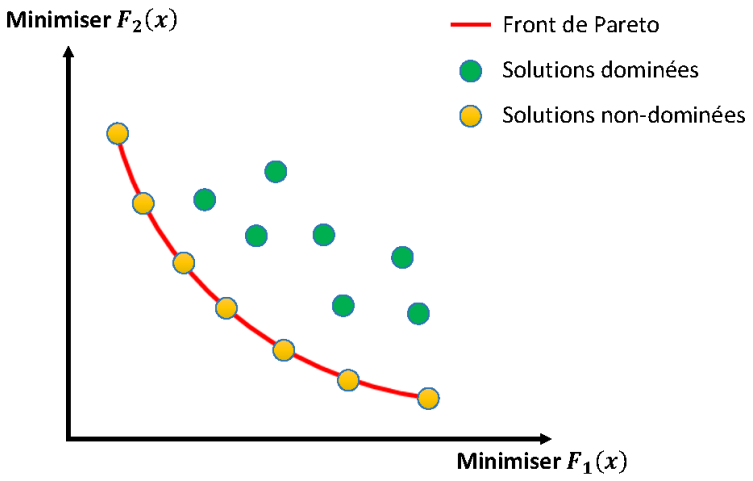


FIGURE 4.1 – *Exemple d'ensemble de solutions dominées, non dominées et de front de pareto pour un problème d'optimisation bi-critère*

Dans ce contexte d'optimisation multi-critères, notre objectif principal est d'explorer le front de Pareto dans sa totalité ou d'obtenir une approximation de cet ensemble, où les valeurs des fonctions objectifs se rapprochent des solutions Pareto-optimales. Cependant, il est crucial de noter que le front de Pareto peut souvent contenir un nombre exponentiel de solutions, ce qui rend impératif de quantifier l'intérêt des ensembles de solutions non-dominées afin d'évaluer la qualité des résultats obtenus. C'est pourquoi l'utilisation d'une méthode d'analyse de décision multi-critères, telle que TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution), s'avère indispensable pour parvenir à des solutions de compromis efficaces au sein du front de Pareto.

La méthode TOPSIS est une technique d'analyse de décision multi-critères qui a été initialement développée par Hwang et Yoon en 1981 [160]. Elle est utilisée pour classer les alternatives et obtenir les meilleures performances dans la prise de décision multi-critères. Sa caractéristique principale réside dans le choix de l'action la plus proche de la solution idéale tout en étant la plus éloignée de la solution anti-idéale. Pour une compréhension approfondie de la mise en œuvre de la méthode TOPSIS, veuillez vous référer à la sous-section 4.3.3.

4.3.3 TOPSIS

L'approche TOPSIS repose sur deux alternatives potentielles : la Solution Idéale Positive (SIP) et la Solution Idéale Négative (SIN). La SIP vise à maximiser les critères de bénéfice et à minimiser les critères de coût, tandis que la SIN vise à maximiser les critères de coût et à minimiser les critères de bénéfice. Les critères de bénéfice sont destinés à être maximisés, tandis que les critères de coût sont destinés à être minimisés. La meilleure alternative est celle qui présente la distance la plus proche par rapport à la SIP et la distance la plus éloignée par rapport à la SIN. Les différentes étapes de l'algorithme TOPSIS pour trouver la meilleure solution sont décrites ci-dessous.

Étant donné une Matrice de Décision (MD) comprenant q alternatives et r critères : le makespan (MS), le coût (C) et la fiabilité (F).

$$MD = \begin{pmatrix} MS_{T11} & C_{T12} & F_{T13} \\ MS_{T21} & C_{T22} & F_{T23} \\ \vdots & \vdots & \vdots \\ MS_{Tq1} & C_{Tq2} & F_{Tq3} \end{pmatrix} \tag{4.9}$$

- **Étape 1 :** Normalisation de la Matrice de Décision (NMD)

$$NMD_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{q} X_{ij}^2}} \quad pour \quad j = 1, 2, \ldots, r \tag{4.10}$$

où X_{ij} représente la valeur de la i^{me} alternative par rapport au j^{me} critère.

- **Étape 2 :** calcul de la matrice de décision normalisée pondérée

$$\begin{aligned} MS_T &= w_1 * MS_T \\ C_T &= w_2 * C_T \\ F_T &= w_3 * F_T \end{aligned} \tag{4.11}$$

où w_1, w_2, w_3 représentent les valeurs de pondération prescrites pour chaque critère.

- **Étape 3 :** identification des solutions idéales positives S^+ et les solutions idéales négatives S^- en fonction de l'impact positif et négatif des critères.

$$\begin{aligned} S^+ &= \{V_1^+, V_2^+, \ldots, V_r^+\} \\ S^- &= \{V_1^-, V_2^-, \ldots, V_r^-\} \end{aligned} \tag{4.12}$$

où S_j^+ et S_j^- désignent respectivement les meilleures et les pires valeurs du critère j pour chaque alternative idéale.

- **Étape 4 :** calcul des mesures de séparation pour chaque alternative en utilisant la distance euclidienne par rapport aux solutions idéales S^+ et anti-idéales S^-.

$$\begin{aligned} D_i^+ &= \sqrt{\sum_{j=1}^{r} (X_{ij} - S_j^+)^2} \\ D_i^- &= \sqrt{\sum_{j=1}^{r} (X_{ij} - S_j^-)^2} \end{aligned} \tag{4.13}$$

- **Étape 5 :** évaluation de la Proximité Relative (PR) par rapport à la solution idéale. Plus la valeur de RC est élevée, plus l'alternative est proche de la solution idéale positive et éloignée de la solution idéale négative.

$$PR_i = \frac{D_i^-}{D_i^+ + D_i^-} \quad pour \;\; i = 1, 2, \ldots, q, \;\; 0 \leq RC_i \leq 1 \tag{4.14}$$

- **Étape 6 :** dans cette étape finale, nous classons les q alternatives en fonction de leur valeur de PR_i. La proximité relative est considérée comme la fonction de fitness, et la meilleure solution est celle qui obtient la valeur de fitness la plus élevée.

4.3.4 Adaptation du FPA

La métaheuristique FPA est adaptée à notre problème en considérant un ensemble de machines virtuelles comme une population et une machine virtuelle comme un nouveau candidat à la solution générée pendant le processus d'optimisation, dans le but de trouver la machine virtuelle appropriée pour chaque tâche soumise. Le pseudo-code de l'algorithme basé sur FPA adapté est présenté dans l'Algorithme 2.

Algorithm 2 Algorithme FPA adapté à la planification des tâches

1: Initialiser les paramètres de l'algorithme : taille de la population, nombre maximum d'itérations et probabilité de commutation $p \in [0, 1]$
2: **while** $T \neq \phi$ **do**
3: Initialiser la population initiale avec des VMs aléatoires
4: Trouver la meilleure machine virtuelle V_{best} pour la tâche T_i dans la population initiale en utilisant l'approche Pareto-TOPSIS
5: **while** critère d'arrêt **do**
6: **for each** machine virtuelle V_i dans la population **do**
7: **if** $rand() < p$ **then**
8: Tirer une taille de pas L selon une distribution de Lévy
9: Effectuer une pollinisation globale via $V_i^{'} = V_i + L(V_i - V_{best})$
10: **else**
11: Tirer ε selon une distribution uniforme dans $[0, 1]$
12: Choisir aléatoirement deux machines virtuelles V_j et V_k parmi toutes les solutions
13: Effectuer une pollinisation locale via $V_i^{'} = V_i + \varepsilon(V_j - V_k)$
14: **end if**
15: Évaluer la machine virtuelle $V_i^{'}$ en suivant Pareto-TOPSIS
16: **if** la nouvelle machine générée est meilleure **then**
17: Remplacer V_i par $V_i^{'}$
18: **end if**
19: **end for**
20: Mettre à jour V_{best} (la meilleure machine dans la population) selon Pareto-TOPSIS
21: **end while**
22: Mettre à jour T
23: **end while**

L'algorithme commence par la définition de ses paramètres de contrôle : la probabilité de commutation, la taille de la population et le nombre maximum d'itérations. Dans les étapes 3 et 4, un ensemble aléatoire de machines virtuelles, appelé population initiale, est généré. Ensuite, la valeur de fitness est calculée pour évaluer les performances de chaque machine virtuelle. La meilleure machine est identifiée et sauvegardée en tant que V_{best}. À chaque itération de la boucle interne (lignes 5 à 21), pour chaque machine, un nombre aléatoire entre 0 et 1 est généré. Si la valeur obtenue est supérieure au paramètre de probabilité de commutation p, une pollinisation globale est réalisée en utilisant une distribution de Lévy pour calculer une nouvelle solution ; sinon, une pollinisation locale est appliquée en utilisant une distribution uniforme. Ensuite, la nouvelle solution générée est examinée pour déterminer si elle est meilleure que la solution actuelle, et si c'est le cas, la population est mise à jour. Enfin, le processus est itéré jusqu'à ce qu'il n'y ait plus de tâches à planifier. Il convient de noter que les tâches arrivées ne sont pas immédiatement attribuées aux machines virtuelles. Une phase de pré-simulation du processus de mapping à l'aide de FPA adapté est nécessaire pour trouver la machine virtuelle appropriée pour chaque tâche avant de l'envoyer au broker.

Pour déterminer les meilleures solutions de compromis, il est important de suivre un processus en deux étapes. Tout d'abord, ce processus exploite le concept d'optimalité de Pareto pour identifier un ensemble non-dominé de solutions. Ensuite, il extrait la solution préférée en fonction de certains critères parmi le front de Pareto en utilisant la technique TOPSIS.

4.3.5 Critères d'évaluation de QoS

Dans le cadre de la conception d'un algorithme d'optimisation pour un problème donné, il est essentiel de définir des critères d'évaluation pertinents, en fonction de la nature du problème à traiter. Dans cette étude, nous évaluons la sélection des machines virtuelles au cours du processus d'ordonnancement sur la base de trois critères de qualité de service perçue par l'utilisateur final. Ces critères sont le makespan, le coût d'exécution et la fiabilité.

Makespan : il s'agit du temps global nécessaire pour achever l'exécution de toutes les tâches. Il est calculé comme la différence entre le moment de la soumission de la première tâche et le moment de la réception des résultats (le moment où la dernière tâche est terminée). Mathématiquement, le makespan s'exprime comme suit :

$$Makespan = max_{j \in TSK} \left(FinishTime_j\right) - min_{i \in TSK} \left(SubmissionTime_i\right) \quad (4.15)$$

où TSK est l'ensemble des tâches soumises à l'ordonnanceur pour exécution.

Coût : la plupart des fournisseurs de services Cloud facturent leurs services. Dans notre cas, le coût total d'exécution se compose de la somme des coûts individuels associés aux VMs utilisées pour l'ensemble des tâches. Le coût d'utilisation pour une instance donnée est calculé à l'aide de la fonction de facturation, qui prend en compte deux paramètres : le prix unitaire de la ressource et le temps d'exécution nécessaire pour accomplir la tâche. Sa formule est la suivante :

$$Coût = \sum_{i=1}^{n} (vm_i^{time} * vm_i^{Uprice}) \quad (4.16)$$

où n représente le nombre de tâches dans l'application, vm^{time} est le temps d'exécution d'une VM pour une tâche donnée et vm^{Uprice} est le prix unitaire de la VM pour l'exécution d'une tâche.

Fiabilité : elle représente la probabilité qu'une tâche soit exécutée avec succès, sans aucune défaillance des ressources. Notre modèle de mesure de la fiabilité s'inspire de celui présenté dans [161]. Il est basé sur un taux de défaillance λ qui est une propriété intrinsèque de la ressource prenant des valeurs allant de 10^{-5} à 10^{-7} [47]. La formule pour calculer la fiabilité est la suivante :

$$Fiabilité = \exp^{-\sum_{i=1}^{n} TE(T_i) * \lambda_j} \quad (4.17)$$

où $TE(T_i)$ représente le temps d'exécution de la tâche T_i et λ_j est le taux de défaillance de la machine exécutant la tâche.

4.3.6 Fonction fitness

Nous avons comparé notre approche à la technique d'agrégation par somme pondérée utilisée dans l'étude de Gupta et al. [152], qui est, pour autant que nous le sachions, le travail le plus proche de celui présenté dans cette contribution. Cette technique consiste à agréger les différents critères de qualité de service mentionnés précédemment en un seul objectif, comme défini par l'équation (éq. 4.18).

$$F = w_1 * Makespan + w_2 * Coût + w_3 * \left(\frac{1}{Fiabilité}\right) \tag{4.18}$$

où $w = \{w_1, w_2, w_3\}$ désigne le vecteur de pondération qui reflète l'importance ou les exigences de l'utilisateur pour chaque critère.

Afin d'optimiser simultanément des objectifs divers, souvent en conflit, et de fournir une plus grande flexibilité à l'utilisateur final en fonction de ses besoins en objectifs prescrits, il est nécessaire d'utiliser une fonction de compromis appropriée qui prend en compte les paramètres des critères considérés. Cela permet d'obtenir des compromis efficaces entre les objectifs ciblés pour répondre aux exigences de qualité de service de l'utilisateur. Par conséquent, nous avons exploré des solutions d'agrégation et la méthode de prise de décision multicritères TOPSIS en attribuant différentes pondérations à chaque objectif.

Dans cette étude, nous avons utilisé deux vecteurs de pondération avec des poids différents, à savoir $\{0.5, 0.2, 0.3\}$ et $\{0.2, 0.5, 0.3\}$, correspondant respectivement aux objectifs considérés {makespan, coût, fiabilité}. Cela signifie que nous accordons plus d'importance à l'objectif makespan dans le premier vecteur, et au coût dans le second vecteur de pondération.

Il convient de noter que nous avons trois métriques de qualité de service à optimiser ; réparties en deux catégories : les métriques de QoS à maximiser (fiabilité) et celles à minimiser (makespan, coût). Dans les deux cas, les métriques de QoS ont des échelles et des valeurs très différentes. Par conséquent, une étape de normalisation est nécessaire pour que la fonction de fitness ne soit pas biaisée par l'une des métriques ayant une valeur significativement plus élevée que les autres. Dans notre approche, la valeur normalisée pour une métrique de qualité de service donnée $QoS(i)$ est calculée selon l'éq. (4.19) [162, 163].

$$QoS_{iNormalis} = \begin{cases} \frac{Max\_QoS_i - QoS_i}{Max\_QoS_i - Min\_QoS_i} & \text{si } QoS_i \text{ à minimiser} \\ \\ \frac{QoS_i - Min\_QoS_i}{Max\_QoS_i - Min\_QoS_i} & \text{si } QoS_i \text{ à maximiser} \end{cases} \tag{4.19}$$

où, les valeurs Max_QoS_i et Min_QoS_i sont continuellement mises à jour au cours du processus d'ordonnancement, reflétant respectivement la pire et la meilleure mesure QoS_i à une itération donnée.

4.4 Méthode de la somme pondérée

Notre étude vise à comparer notre proposition avec l'approche de planification multicritère basée sur la Méthode de Somme Pondérée (Weigh-

ted Sum Model, ou WSM). Dans cette perspective, nous décrivons brièvement ci-après les principales caractéristiques de cette méthode.

En théorie de la décision, la technique WSM se distingue en tant que méthode multi-objectif la plus connue en raison de sa simplicité et de sa rapidité. Elle est conçue pour la prise de décision en temps réel, évaluant de multiples alternatives en se basant sur des critères de décision quantitatifs partageant les mêmes unités de mesure. Le concept de cette méthode implique l'agrégation des objectifs d'optimisation divers en une fonction objectif unique. Si nécessaire, un paramètre de pondération peut être introduit pour exprimer l'importance relative d'un objectif par rapport à d'autres. De cette manière, une fois que la formulation du problème et l'importance de chaque critère sont attribués, une seule solution est générée comme indiqué par l'équation suivante :

$$f(x) = \sum_{i=1}^{n} w_i f_i \tag{4.20}$$

où les poids $w_1, w_2, ...w_n$ doivent respecter les deux contraintes suivantes :

$$\sum_{i=1}^{n} w_i = 1, \quad 0 \leq w_i \leq 1 \tag{4.21}$$

Compte tenu la nature conflictuelle des objectifs d'optimisation visés, il est souvent difficile d'atteindre une optimisation équitable pour tous ces objectifs simultanément. Par conséquent, l'introduction de paramètres de pondération permet à l'utilisateur d'exprimer ses préférences en favorisant certains objectifs par rapport à d'autres. L'idée de base est qu'un critère qui est en conflit avec un critère favorisé reçoit la valeur minimale parmi les poids possibles (les poids restants après l'opération : 1 - le poids du critère pertinent ; [164]). Par conséquent, le processus d'optimisation recherche une solution de compromis entre tous les objectifs, tout en favorisant ceux qui correspondent aux préférences de l'utilisateur.

4.5 Résultats de simulation et discussions

Dans cette section, nous présentons les résultats obtenus suite aux simulations réalisées en utilisant CloudSim. Avant d'aborder les résultats, nous exposons tout d'abord les différentes hypothèses relatives à l'environnement de simulation, ainsi que les divers paramètres associés à notre approche.

4.5.1 Environnement de simulation

Nous modélisons une application sous forme d'un ensemble de n clusters qui définissent les tâches soumises par l'utilisateur $C = \{c_1, c_2, ..., c_n\}$. Chaque ensemble c_i peut comporter un nombre variable de tâches $T = \{T_{i1}, T_{i2}, ..., T_{ik}\}$, qui peuvent être exécutées soit séquentiellement, soit en

parallèle. Chaque tâche T_{ik} est définie par un couple (a_{ik}, l_{ik}), où a_{ik} représente l'heure d'arrivée de la tâche T_{ik} et l_{ik} correspond à la longueur de la tâche T_{ik}.

Considérons un ensemble de m machines virtuelles $S = \{VM_1, VM_2, ..., VM_m\}$. Chaque machine virtuelle VM_j est associée à un triplet (s_j, u_j, r_j), où s_j représente la vitesse de traitement de la j-ième machine virtuelle, exprimée en MIPS, u_j est le coût unitaire de l'utilisation de la ressource et r_j est le taux de fiabilité de la ressource.

Dans nos simulations, nous partons des hypothèses suivantes :

(1) À un instant donné, chaque tâche doit être traitée sur une seule ressource sélectionnée, tandis que chaque ressource peut exécuter plusieurs tâches.

(2) Chaque cluster peut contenir un nombre différent de tâches. Il est important de noter qu'il n'existe aucune dépendance de traitement entre les tâches au sein d'un même cluster. En d'autres termes, les tâches au sein d'un cluster peuvent être exécutées dans n'importe quel ordre.

(3) Le cluster parent doit être exécuté avant que tout autre cluster ne commence son exécution. En d'autres termes, le cluster désigné par c_1 doit être achevé avant que celui désigné par c_2 puisse commencer son exécution (voir Figure 4.2).

(4) Certains paramètres doivent être précalculés pour chaque tâche sur chaque ressource attribuée.

(5) Une fois qu'une tâche a été traitée, toute interruption est ignorée.

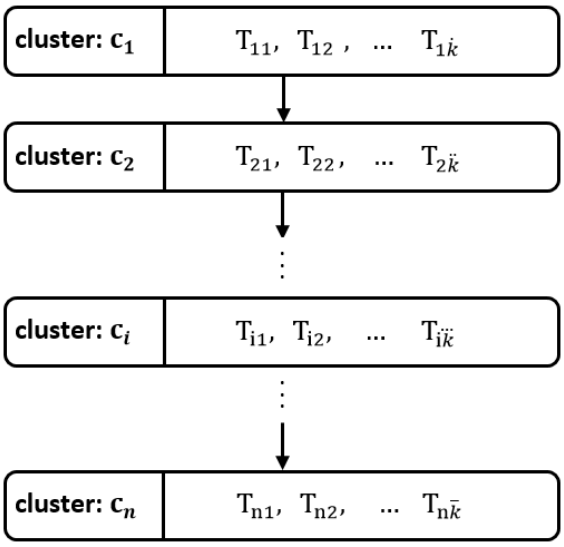


FIGURE 4.2 – *Exemple simple d'application*

Comme mentionné ci-dessus, les tâches sont regroupées en clusters, où leurs dépendances indiquent l'ordre de leur exécution. Dans nos simulations, nous avons établi 15 contraintes de précédence à satisfaire, i.e. nous utilisons 15 clusters. Tous les paramètres des simulations effectuées sont indiqués dans la Table 4.1. Notons que la configuration expérimentale ainsi

que les paramètres utilisés dans notre étude, tels que la distribution de la charge de travail, les temps de soumission des tâches et la définition du Cloud, sont choisis de manière à être représentatifs et conformes à ceux utilisés dans la littérature [165, 166, 167].

Type d'entité	Parametre	Valeur
Data-center	No. de data-centers	4
Machine physique	No. de PHs	8
	PES	4 (Quad core)
	MIPS	6 000
	RAM	20 GB
	Stockage	1 TB
	Bandwidth	10 GB
Machine virtuelle	No. de VMs	20 - 40 - 60 - 80
	MIPS	1 000 à 5 000
	Coût	1.0 à 30.0
	Taux d'échec	10^{-5} à 10^{-7}
	RAM	1 GB à 5 GB
	Stockage	10 GB
	Bandwidth	100 MB à 500 MB
	Politique	Time Shared
Cloudlets	No. de cloudlets	500 - 1 000 - 1 500
	Taille	3 000 à 10 000
	No. de clusters	15
	Type	Hétérogène
	Temps de soumission	Distribution de Poisson du paramètre λ

TABLE 4.1 – *Paramètres de simulation CloudSim*

4.5.2 Configuration expérimentale

Le bon fonctionnement d'un l'algorithme dépend fortement d'un bon paramétrage. En réalité, il n'existe pas de valeurs standard pour les différents paramètres spécifiques à chaque algorithme qui garantissent un fonctionnement optimal. Il est important de souligner que le paramétrage est étroitement lié au problème traité, car les paramètres varient d'un problème à un autre. Ainsi, pour optimiser les performances de l'algorithme FPA utilisé, nous avons déterminé le paramétrage approprié de manière empirique.

En effet, trois paramètres cruciaux sont définis pour l'approche basée sur FPA : la taille de la population, le paramètre de contrôle λ, et la probabilité de commutation p, qui détermine le pourcentage de diversification et d'intensification lors de la recherche. Pour toutes nos simulations, nous avons utilisé une taille de population égale à 20 % du nombre total de VMs créées. La probabilité de commutation et le paramètre de contrôle sont fixés respectivement à $0,8$ et $1,5$, conformément au travail de Yang et al. [157].

D'un autre côté, l'algorithme conçu appartient à la catégorie des approches dites stochastiques ou non déterministes en raison de ses paramètres aléatoires. Ceci pose un problème lors de la mesure des résultats car il n'est pas approprié de se baser sur une seule mesure pour établir une conclusion. Il est donc indispensable d'effectuer plusieurs mesures afin de consolider les résultats. Ainsi, pour chaque type d'ordonnancement, 10 simulations ont été effectuées et c'est la moyenne de ces dernières qui sera prise en compte et présentée dans les sections suivantes dans les tableaux et figures comparatifs.

4.5.3 Discussion des résultats

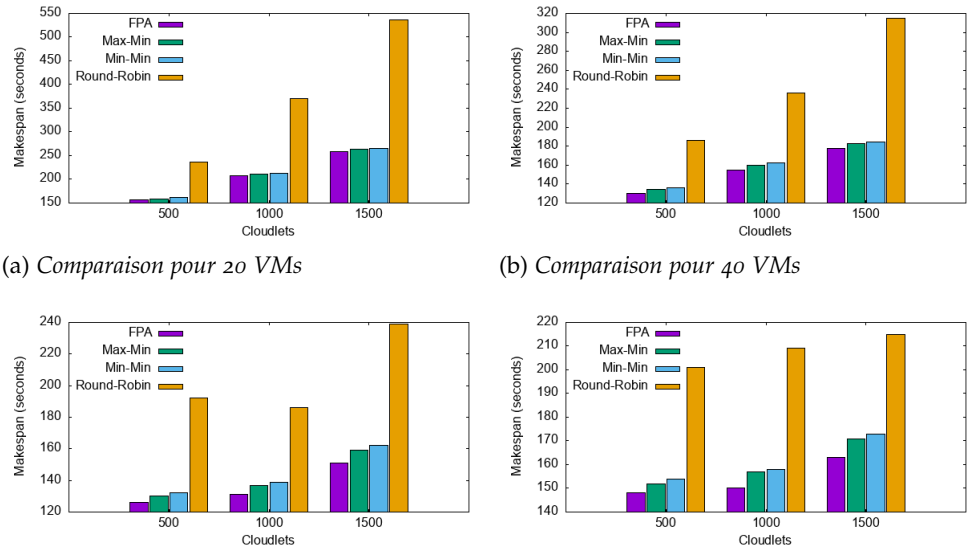
Evaluation mono objectif

Les résultats de la comparaison entre l'algorithme proposé et les heuristiques Max-Min, Min-Min et Round Robin sont présentés dans la Table 4.2. Nous rappelons que, dans cette série d'expériences, nous ne considérons qu'un seul critère, à savoir la valeur du temps makespan. Le nombre de machines virtuelles varie de 20 à 80 avec des incréments de 20, et trois ensembles de simulations avec le même nombre de tâches sont réalisés : 500, 1000 et 1500. Les résultats obtenus sont illustrés dans les Figures 4.3(a)(b)(c)(d).

Nombre de VMs	Cloudlets	Méthode utilisée			
		FPA	Max-Min	Min-Min	Round Robin
20	500	157	159	162	236
	1 000	208	211	213	370
	1 500	259	263	266	537
40	500	130	134	136	186
	1 000	155	160	162	236
	1 500	178	183	184	315
60	500	126	130	132	192
	1 000	131	137	139	186
	1 500	151	159	162	239
80	500	148	152	154	201
	1 000	150	157	158	209
	1 500	163	171	173	215

TABLE 4.2 – *Résultats comparatifs de l'algorithme proposé par rapport à d'autres heuristiques d'ordonnancement*

Comme prévu, nous pouvons observer que l'objectif du makespan augmente à mesure que le nombre de tâches augmente. Ceci s'explique par le fait que la capacité des ressources à exécuter les tâches diminue progressivement à mesure que la charge de travail augmente au fil du temps. De plus, notre proposition présente de meilleures performances que toutes les autres techniques heuristiques, en particulier pour un grand nombre de tâches. Par exemple, dans le cas de 60 VMs, les performances de l'algorithme FPA adapté sont supérieures à celles de Max-Min de 4 et 8 secondes



(a) *Comparaison pour 20 VMs*

(b) *Comparaison pour 40 VMs*

(c) *Comparaison pour 60 VMs*

(d) *Comparaison pour 80 VMs*

FIGURE 4.3 – *Comparaison entre FPA et les heuristiques*

pour les ensembles de 500 et 1500 tâches respectivement. Cela peut s'expliquer par le fait que la politique du FPA adapté explore efficacement l'espace de recherche en utilisant des sauts de Lévy. Cela conduit à une amélioration notable de l'objectif d'optimisation global par rapport aux autres heuristiques, quel que soit le nombre de VMs utilisées ou le nombre de tâches soumises lors du processus de planification.

Évaluation multi-objectifs

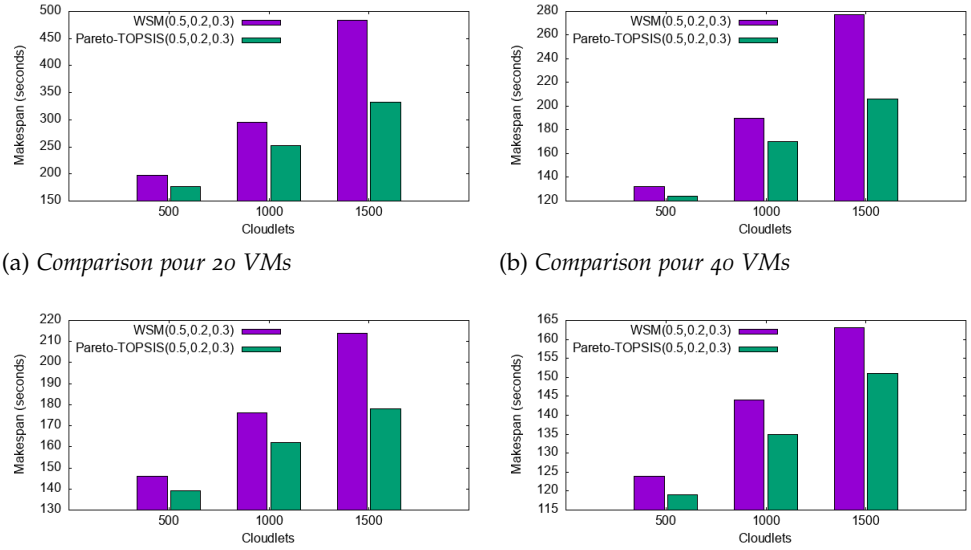
Le deuxième ensemble d'expérimentations est dédié au cas de l'ordonnancement multi-objectif. Une comparaison a été réalisée en utilisant deux variantes de la fonction de compromis pour évaluer la qualité des solutions du FPA. Comme indiqué dans les sections précédentes, l'objectif est d'évaluer l'efficacité du concept d'optimalité de Pareto avec la technique TOPSIS par rapport à la somme pondérée [152] qui fusionne les différentes métriques de QoS (voir Section 4.3.6).

Tout d'abord, nous évaluons le makespan des approches comparées. Comme présenté dans la Table 4.3 et les Figures 4.4(a)(b)(c)(d), il est évident que l'approche Pareto-TOPSIS présente de meilleures performances avec le vecteur de poids $\{0.5, 0.2, 0.3\}$ qui favorise l'objectif du temps makespan. Par exemple, par rapport à la méthode de somme pondérée, le taux de réduction de notre proposition dans le cas de 40 VMs est de 6%, 10.5% et 25.6% pour 500, 1000 et 1500 tâches, respectivement.

Cette expérience montre que la stratégie Pareto-TOPSIS atténue ce problème en trouvant de meilleures solutions de compromis entre les objectifs considérés. Cela s'explique par le concept de domination de Pareto qui ne permet pas à un objectif de dominer un autre. Nous pouvons également observer à partir du Tableau 4.3 que lorsque nous fixons le nombre de tâches et augmentons le nombre de VMs, le taux de réduction diminue progressivement. Par exemple, dans le cas de 500 tâches, le taux de réduc-

Nombre de VMs	Cloudlets	Technique utilisée		Taux de réduction (%)
		WSM [152]	Pareto-TOPSIS	
20	500	198	177	10.6
	1 000	295	252	14.5
	1 500	484	333	31.1
40	500	132	124	6
	1 000	190	170	10.5
	1 500	277	206	25.6
60	500	146	139	4.7
	1 000	176	162	7.9
	1 500	214	178	16.8
80	500	124	119	4.0
	1 000	144	135	6.2
	1 500	163	151	7.3

TABLE 4.3 – *Comparaison des performances de makespan avec le vecteur de pondération* $\{0.5, 0.2, 0.3\}$



(a) *Comparison pour 20 VMs*

(b) *Comparison pour 40 VMs*

(c) *Comparison pour 60 VMs*

(d) *Comparison pour 80 VMs*

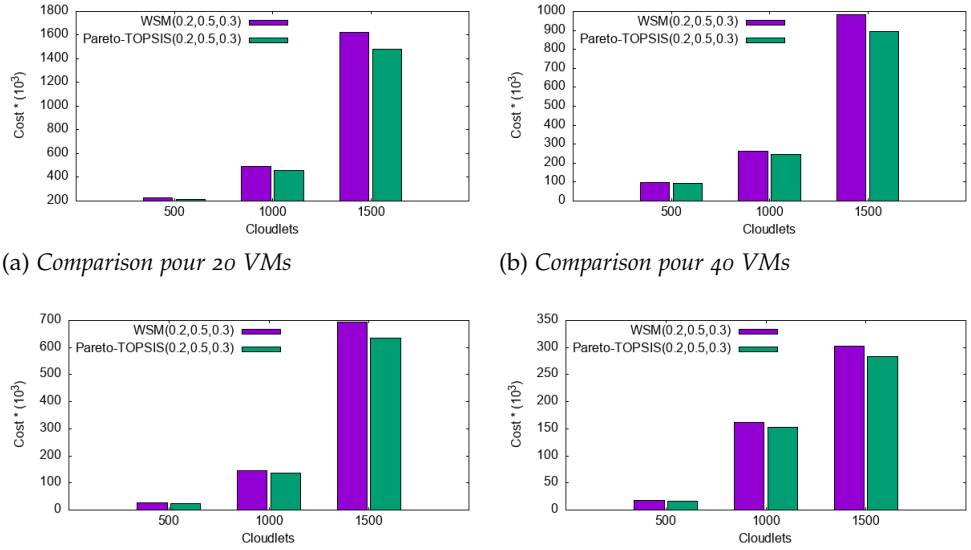
FIGURE 4.4 – *Comparaison en termes de makespan*

tion est de 10.6%, 6%, 4.7% et 4.0% pour 20 VMs, 40 VMs, 60 VMs et 80 VMs, respectivement. Cependant, cela n'est pas particulièrement surprenant étant donné que les VMs ont une plus grande capacité à influencer sur le temps makespan.

En comparant les résultats de la somme pondérée avec ceux de Pareto-TOPSIS en termes de coût d'exécution, nous pouvons observer dans la Table 4.4 et à partir des Figures 4.5(a)(b)(c)(d) respectivement que ce dernier obtient de meilleures valeurs de performance que la somme pondérée dans tous les scénarios testés avec un vecteur de poids favorisant le coût d'exécution. Par exemple, dans le cas de 60 VMs, l'approche basée sur Pareto-TOPSIS atteint des solutions qui sont 8.6% plus efficaces en termes

Nombre de VMs	Cloudlets	Technique utilisée		Taux de réduction (%)
		WSM [152]	Pareto-TOPSIS	
20	500	225 694	213 626	5.3
	1 000	490 613	460 142	6.2
	1 500	1 626 716	1 478 813	9.0
40	500	97 390	92 552	4.9
	1 000	262 573	247 856	5.6
	1 500	984 540	895 000	9.0
60	500	25 297	24 542	2.9
	1 000	143 927	135 419	5.9
	1 500	694 443	634 251	8.6
80	500	17 141	16 776	2.1
	1 000	161 064	152 121	5.5
	1 500	302 622	282 601	6.6

TABLE 4.4 – *Comparaison des performances de coût avec le vecteur de pondération* $\{0.2, 0.5, 0.3\}$



(a) *Comparison pour 20 VMs*

(b) *Comparison pour 40 VMs*

(c) *Comparison pour 60 VMs*

(d) *Comparison pour 80 VMs*

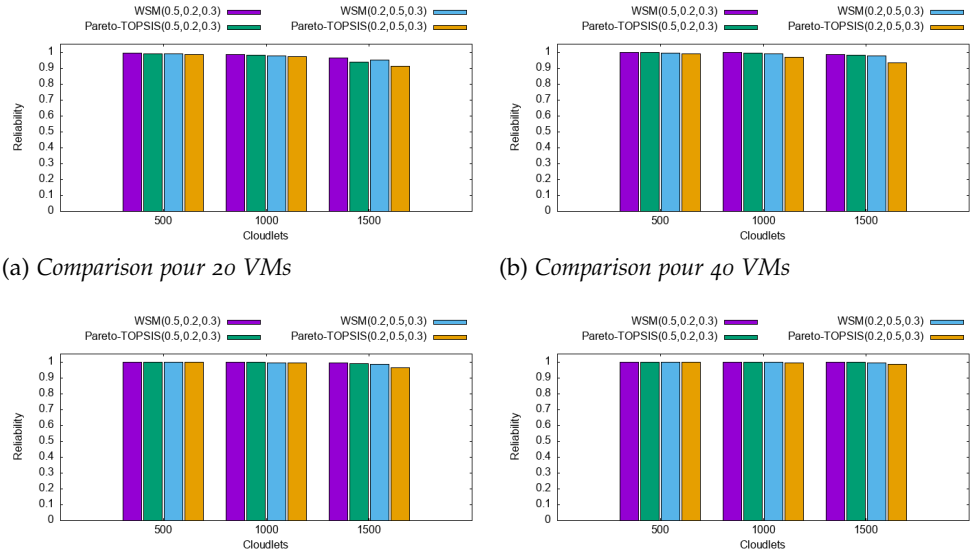
FIGURE 4.5 – *Comparaison en termes de coût*

de coût d'exécution que celles fournies par la méthode de la somme pondérée. Cette amélioration est due au fait que l'approche basée sur Pareto-TOPSIS cherche à assurer des compromis efficaces entre les trois objectifs considérés en raison de leur importance. Cela conduit à une minimisation efficace des coûts au cours du processus d'ordonnancement. Cependant, en termes d'optimisation de la fiabilité, nous pouvons remarquer à partir de la Table 4.5 et les Figures 4.6(a)(b)(c)(d) respectivement que la différence entre les performances obtenues par les méthodes comparées est presque similaire, quel que soit le nombre de VMs et le nombre de tâches.

Dans l'ensemble, pour les objectifs de makespan et de coût, il est plus intéressant d'utiliser l'approche basée sur Pareto-TOPSIS avec le vecteur de poids approprié. En effet, la somme pondérée est une simple projec-

Nombre de VMs	Cloudlets	Technique utilisée			
		$\{0.5, 0.2, 0.3\}$		$\{0.2, 0.5, 0.3\}$	
		WSM [152]	Pareto-TOPSIS	WSM [152]	Pareto-TOPSIS
20	500	0.993	0.99	0.988	0.986
	1 000	0.986	0.979	0.978	0.97
	1 500	0.964	0.937	0.952	0.909
40	500	0.998	0.998	0.994	0.988
	1 000	0.996	0.994	0.988	0.967
	1 500	0.987	0.981	0.977	0.933
60	500	0.999	0.999	0.999	0.998
	1 000	0.998	0.998	0.995	0.992
	1 500	0.994	0.991	0.987	0.962
80	500	0.999	0.999	0.999	0.999
	1 000	0.999	0.998	0.996	0.993
	1 500	0.998	0.997	0.994	0.987

TABLE 4.5 – *Comparaison des performances en matière de fiabilité*



(a) *Comparison pour 20 VMs*

(b) *Comparison pour 40 VMs*

(c) *Comparison pour 60 VMs*

(d) *Comparison pour 80 VMs*

FIGURE 4.6 – *Comparaison en termes de fiabilité*

tion de l'optimisation multi-objectif vers l'optimisation mono-objectif. Une approche purement multi-objectif comme Pareto-TOPSIS est plus recommandée dans ce contexte. En ce qui concerne la fiabilité, il s'agit d'une mesure un peu particulière car dans le Cloud, les VMs sont considérées comme fiables, les fournisseurs de services Cloud offrant des services avec des taux de défaillance très faibles, pouvant aller jusqu'à 10^{-7}. Pour cette raison, la méthode utilisée n'a pas beaucoup d'impact sur cet objectif.

4.6 Conclusion

Dans ce chapitre, nous avons présenté notre première contribution dans le cadre de cette thèse. Il s'agit d'un nouvel algorithme d'ordonnancement qui traite à la fois les variantes d'optimisation mono-objectif et multi-objectif en vue de satisfaire les exigences des utilisateurs Cloud. Cet algorithme repose sur une combinaison d'une approche de pollinisation inspirée de la nature et de l'optimalité de Pareto. De plus, il fait usage de la technique TOPSIS pour assurer un compromis efficace entre les objectifs ciblés, à savoir le makespan, le coût et la fiabilité. L'évaluation expérimentale détaillée et les différents scénarios étudiés à l'aide de CloudSim confirment les mérites de notre proposition.

Le chapitre suivant sera consacré à la présentation de notre deuxième contribution.

5 Nouvelles approches d'ordonnancement multi-objectif combinant l'intelligence en essaim et le comportement de pollinisation

Sommaire

5.1 Introduction

Le système Cloud se compose principalement de deux entités : les consommateurs et les fournisseurs de services Cloud. Ces deux parties sont tenues de respecter leurs engagements, tels qu'énoncés dans l'accord de niveau de service, afin d'éviter des pénalités. Les fournisseurs de services Cloud doivent honorer leurs engagements en termes de puissance de calcul, la fiabilité du service, la sécurité et la disponibilité des ressources, dans le but de satisfaire leurs clients tout en cherchant à maximiser leurs profits. D'un autre côté, les consommateurs cherchent à accomplir leurs tâches dans le laps de temps le plus court possible. Cependant, les contraintes de qualité de service peuvent être divisées en deux catégories principales : les besoins du fournisseur et les préférences du consommateur. Par conséquent, les stratégies d'ordonnancement ne doivent pas seulement viser à répondre aux exigences de qualité de service définies par l'utilisateur via les SLA, mais aussi à garantir que les bénéfices du fournisseur ne soient pas significativement impactés.

D'autre part, les diverses approches d'optimisation métaheuristiques que nous explorons dans le cadre de cette thèse diffèrent dans leur manière d'aborder l'exploration (diversification) ou l'exploitation (intensification) de l'espace de recherche. En effet, chacune de ces approches combine de manière unique des stratégies de diversification et d'intensification pour atteindre des solutions efficaces. Cependant, cette catégorie de techniques d'optimisation présente quelques faiblesses, notamment le problème du phénomène de convergence prématurée et la difficulté à échapper aux optima locaux, en particulier lorsqu'elle est confrontée à un vaste espace de solutions, ce qui peut avoir un impact sur la qualité de service et les performances du système. Par conséquent, il devient important de développer des stratégies d'ordonnancement des tâches efficaces et adaptables visant à atteindre un bon équilibre entre l'exploration et l'exploitation, tout en répondant aux exigences des fournisseurs et en satisfaisant les attentes des utilisateurs Cloud.

Pour relever ce défi, ce chapitre introduit deux contributions majeures. Nos propositions se matérialisent sous la forme de nouveaux algorithmes d'optimisation hybrides, tenant compte à la fois des exigences des fournisseurs de services et des besoins des utilisateurs. À notre connaissance, aucune étude antérieure dans la littérature n'a exploré les améliorations que nous proposons dans le présent chapitre, en combinant les caractéristiques clés des métaheuristiques pour aborder l'ordonnancement multicritère des tâches dans les environnements Cloud hétérogènes. Dans un premier temps, nous allons exposer les travaux connexes. Ensuite, nous détaillons les approches proposées ainsi que leurs adaptations à notre problématique. Enfin, pour valider l'apport des algorithmes que nous proposons, ce chapitre se conclura par une étude comparative des performances obtenues à travers différentes simulations.

5.2 Travaux connexes

La littérature scientifique regorge de nombreuses stratégies visant à résoudre le problème complexe de l'ordonnancement des tâches dans le Cloud. Ces stratégies varient de l'utilisation d'heuristiques simples à l'exploration de métaheuristiques, en passant par des stratégies d'ordonnancement hybrides. Dans cette section, notre attention se focalisera principalement sur les algorithmes d'ordonnancement hybrides pertinents en lien avec les contributions apportées dans ce chapitre.

Miglani et al. [168] ont récemment introduit un ordonnancement de tâches élastique et persuasif basé sur un algorithme de pollinisation des fleurs modifié. L'approche proposée, appelée planificateur de flux de travail multi-objectif basé sur la fiabilité, vise à attribuer des tâches aux machines les plus adaptées en termes de temps d'exécution, d'efficacité et de coût. Les résultats de simulations ont démontré que cette approche surpasse des méthodes de référence telles que FPA, GWO et GA.

Dans le travail de Gokuldhev et al. [169], un Optimiseur de Loup Gris basé sur la Pollinisation Locale (LPGWO) avec un concept énergétique a été conçu en combinant deux algorithmes d'optimisation, GWO et le concept de pollinisation dans la nature. L'objectif des auteurs est d'équilibrer efficacement le makespan et la consommation d'énergie des ressources. Le LPGWO se compose de trois phases : une phase d'initialisation, une phase GWO et une phase FPA. La phase d'initialisation utilise une cartographie chaotique et la méthode d'apprentissage basée sur l'opposition (OBL) pour fournir une solution d'ordonnancement initiale appropriée. La phase GWO est utilisée pour améliorer la vitesse de convergence, tandis que la phase FPA distribue les données aux prochaines solutions candidates grâce à la pollinisation locale. Le LPGWO proposé a été évalué sur des machines physiques à faible et forte hétérogénéité, et les résultats ont montré qu'il surpassait les approches existantes telles que GA, PSO, Bacteria Foraging Algorithm (BFA) et Multi-Hybrid Bacteria Foraging Algorithm (MHBFA).

Dans une autre étude similaire, un nouvel algorithme d'ordonnancement de tâches multi-objectif, appelé Local Pollination based Moth Search Algorithm (LPMSA), a été développé pour traiter les problèmes d'hétérogénéité et de dynamisme dans les systèmes Cloud [170]. Le LPMSA combine la recherche locale de l'approche FPA avec l'algorithme de recherche de mites pour améliorer son exploitabilité et éviter les optima locaux. L'efficacité du LPMSA a été évaluée sur des machines à hétérogénéité variable avec des paramètres uniformes et non uniformes. Les résultats indiquent une amélioration significative du makespan, de la consommation d'énergie et de la vitesse de convergence du processus d'optimisation par rapport aux approches concurrentes.

Alsadie a élaboré un algorithme d'ordonnancement de Tâches Multi-Objectif basée sur l'Optimiseur de Loup Gris, présenté sous le nom de TSMGWO [171]. Cette technique vise à trouver une solution optimale ou quasi-optimale au problème de planification des tâches dans les systèmes Cloud IaaS en prenant en compte simultanément plusieurs objectifs contradictoires tels que le makespan, l'utilisation des ressources, le degré de déséquilibre et le débit. Les performances de l'algorithme ont été évaluées à

l'aide de trois ensembles de données de référence différents, à savoir GoCJ, Synthetic et HSCP, et les résultats numériques montrent que cette méthode surpasse significativement les heuristiques traditionnelles telles que FCFS et la méthode Throttle Modifiée (MT), ainsi que les métaheuristiques PSO, GA et WOA.

Natesan et Chokkalingam [172] ont développé une version plus précise et efficace de l'algorithme d'optimisation standard du loup gris appelée Mean-GWO. Les résultats de comparaison à l'aide de CloudSim sur deux ensembles de données, à distribution asymétrique à gauche et à distribution asymétrique à droite, ont montré que les équations d'encerclement et de chasse améliorées avaient le potentiel de surpasser les techniques concurrentes, notamment PSO et le GWO standard, en termes de temps d'achèvement et de consommation d'énergie.

Un algorithme multi-objectif basé sur Pareto, appelé PGWO (Pareto-based multi-objective Grey Wolf Optimizer), a été proposé par Khalili et al. [173] pour l'ordonnancement de workflows dans les systèmes Cloud. L'objectif est de trouver un équilibre entre des objectifs contradictoires, à savoir la minimisation du makespan, du coûts et la maximisation du débit des ressources pour le fournisseur. Les performances de l'algorithme ont été évaluées à l'aide de deux modèles de flux de travail : déséquilibré (Montage) et équilibré (Epigenomics). Les résultats des simulations montrent que cet algorithme multi-objectif étendu offre un meilleur compromis entre les objectifs considérés, avec une dispersion maximale des solutions et un taux de couverture plus élevé que Strength Pareto Evolutionary Algorithm 2 (SPEA2).

Gobalakrishnan et Arun [174] ont développé une approche hybride multi-objectif, connue sous le nom Genetic Gray Wolves Optimization (GGWO), dans le but d'améliorer les performances de la planification de workflows en termes de temps total, de coût de migration, d'utilisation de la charge et de consommation d'énergie, tout en équilibrant la charge entre les ressources de calcul. Les auteurs utilisent GA en combinaison avec GWO pour améliorer les performances et accélérer le processus d'optimisation. L'analyse GGWO a été réalisée à l'aide de cinq flux de travail scientifiques courants, à savoir LIGO, Montage, Epigenomics, SIPHT et Cybershake. Les expériences ont révélé que GGWO peut améliorer les performances du processus de planification des tâches par rapport au GWO standard et à l'algorithme GA.

Khurana et Singh [175] ont proposé une approche d'ordonnancement hybride bi-objectif pour les workflows scientifiques, combinant des algorithmes d'inspiration biologique et d'intelligence en essaim. L'algorithme utilise GWO et FPA avec l'algorithme PEFT (Predict Earliest Finish Time) pour l'optimisation globale et locale. L'objectif est de minimiser à la fois le coût monétaire et le temps d'exécution des tâches dans le Cloud en attribuant les tâches soumises aux machines virtuelles disponibles. L'efficacité de la technique proposée a été démontrée par des simulations numériques, comparées à la pollinisation des fleurs et à l'algorithme génétique.

Dans une autre étude, Amer et al. [176] ont introduit une version améliorée de l'optimiseur de harris hawks, appelée Elite Learning Harris Hawks Optimizer (ELHHO), pour traiter le problème d'ordonnancement

multi-objectif. Cette stratégie vise à explorer plus efficacement l'espace de recherche et à améliorer le processus d'exploration de l'algorithme HHO en combinant deux méthodes d'intelligence scientifique : Elite Opposition-Based Learning (EOBL) et minimum completion time. L'EOBL est utilisé dans le processus d'exploration pour améliorer la capacité de recherche globale dans le HHO afin d'équilibrer l'exploration et l'exploitation, tandis que le MCT génère la liste de planification initiale pour servir de population initiale. Les simulations effectuées avec CloudSim ont clairement montré que le travail des auteurs attribue efficacement les tâches soumises aux machines virtuelles disponibles avec un degré d'équilibre, un débit et une utilisation des ressources élevés, tout en réduisant le makespan et le coût d'exécution des ressources.

Dans l'ensemble, les méthodes d'optimisation hybrides d'inspiration biologique et d'intelligence en essaim, étudiées précédemment, présentent un potentiel considérable pour l'optimisation de l'ordonnancement des tâches dans les systèmes de Cloud Computing. Cependant, pour tirer pleinement parti de ce potentiel, des améliorations supplémentaires doivent se concentrer sur l'équilibre entre l'exploration et l'exploitation, en utilisant des opérateurs appropriés, qui jouent un rôle déterminant dans l'éfficacité et la performance globale des algorithmes. C'est dans cette perspective que nos contributions introduiront de nouveaux algorithmes d'optimisation hybride pour l'orchestration multicritères des tâches dans les systèmes Cloud, tout en maintenant la cohérence avec les études précédentes en utilisant une configuration expérimentale et des paramètres similaires, y compris les mêmes métriques de qualité de service et le même jeu de données.

5.3 Présentation des approches proposées

Dans cette section, nous présenterons nos contributions. Toutefois, avant de détailler les approches que nous proposons, nous commençons par exposer les caractéristiques fondamentales des algorithmes qui sont à la base de nos propositions. Il s'agit de l'algorithme de pollinisation des fleurs, déjà exposé dans le chapitre précédent, de l'optimiseur du loup gris, et de l'algorithme des grenouilles.

5.3.1 Algorithme de gray wolf optimizer

En 2014, Mirjalili et Lewis ont présenté l'algorithme d'optimisation métaheuristique GWO (Grey Wolf Optimizer), inspiré du comportement social de chasse des loups gris dans la nature [177]. GWO fonctionne en maintenant un groupe de solutions potentielles appelées "loups", qui sont utilisées pour explorer l'espace de recherche et identifier la meilleure solution. Le modèle mathématique de l'algorithme GWO est divisé en quatre phases, comme décrites ci-dessous [177, 178] :

Hiérarchie sociale : Les loups gris d'une meute établissent une hiérarchie sociale à travers un mécanisme basé sur la dominance, où le loup le plus fort, l'alpha, a le plus de contrôle et prend les décisions pour le reste de la meute. Le deuxième loup le plus fort, le bêta, agit en tant que consultant

de l'alpha. Le troisième loup le plus fort, le delta, est un subordonné qui se soumet aux niveaux supérieurs (alpha et bêta) tout en servant de leader aux loups oméga, les membres les moins gradés de la meute.

Chaque valeur de fitness d'un loup définit son classement dans la hiérarchie sociale pendant la phase de hiérarchie sociale. Les loups les plus forts sont désignés α, β et δ, respectivement, et ils jouent des rôles spécifiques dans les étapes suivantes, tandis que les autres membres de la meute sont considérés comme des oméga(s).

Encerclement de la proie : Une fois que les leaders de la meute sont déterminés, la meute commence à encercler leur proie en se déplaçant en mouvement circulaire. Les leaders guident la meute vers la proie, et les autres membres les suivent. Ce comportement d'encerclement est mathématiquement modélisé à l'aide d'équations suivantes.

$$\begin{aligned} \vec{D} &= |\vec{C}.\vec{X}_p(t) - \vec{X}(t)| \\ \vec{X}(t+1) &= \vec{X}_p(t) - \vec{A}.\vec{D} \end{aligned} \tag{5.1}$$

où $\vec{X}$ représente la position du loup gris actuel et $\vec{X}_p$ représente la position de la proie. Les vecteurs de coefficients $\vec{A}$ et $\vec{C}$ sont calculés comme suit :

$$\vec{A} = 2.\vec{a}.\vec{r_1} - \vec{a} \tag{5.2}$$

$$\vec{C} = 2.\vec{r_2} \tag{5.3}$$

où le vecteur $\vec{a}$ diminue progressivement de manière linéaire pour mettre l'accent sur l'exploration et l'exploitation, respectivement, tandis que $\vec{r_1}$ et $\vec{r_2}$ sont des vecteurs de nombres réels aléatoires générés dans l'intervalle $[0, 1]$.

Chasse : Comme mentionné précédemment, les loups leaders occupent les positions les plus favorables dans la population et jouent un rôle crucial dans la chasse. Les positions des trois meilleurs agents sont enregistrées, et une position de référence pour la proie est calculée en utilisant α, β, et δ. Cette position estimée de la proie sert de guide pour les autres loups afin d'ajuster leurs positions de manière coordonnée, comme indiqué dans les équations suivantes.

$$\vec{D}_\alpha = |\vec{C}_1.\vec{X}_\alpha - \vec{X}|, \;\; \vec{D}_\beta = |\vec{C}_2.\vec{X}_\beta - \vec{X}|, \;\; \vec{D}_\delta = |\vec{C}_3.\vec{X}_\delta - \vec{X}| \tag{5.4}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.\vec{D}_\alpha, \;\; \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.\vec{D}_\beta, \;\; \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.\vec{D}_\delta \tag{5.5}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{5.6}$$

Attaque : Dans cette dernière phase, la meute attaque sa proie et tente de la capturer et de la tuer. Les loups gris utilisent leurs compétences pour la traquer et potentiellement trouver la meilleure solution possible. Plus précisément, les compétences des loups peuvent conduire à la découverte d'une solution globale efficace. Étant donné que le coefficient $\vec{A}$ joue un rôle crucial dans ce processus, une valeur $|A| < 1$ incite les loups à se concentrer sur la capture de la proie actuelle (exploitation), tandis qu'une valeur $|A| > 1$ signifie que les loups gris s'éloigneront de la proie pour rechercher une proie plus appropriée (exploration).

Algorithm 3 Pseudo-code standard de l'algorithme GWO

Initialiser la population de loups gris $X = (x_1,\ x_2,\ ...)$
Initialiser a, A et C
Calculer la fitness de chaque agent de recherche
Premier meilleur agent dénoté par X_α
Deuxième meilleur agent dénoté par X_β
Troisième meilleur agent dénoté par X_δ
while critère d'arrêt **do**
  for each $i = 1 : n$ (tous les n agents de recherche de la population) **do**
    Mettez à jour la position de l'agent de recherche actuel en utilisant éq. (5.6)
  end for
  Mettre à jour a, A et C
  Calculer la valeur de fitness de tous les agents de recherche
  Mettre à jour les positions de X_α, X_β et X_δ.
end while
Retourner X_α

Les quatre phases de l'algorithme d'optimisation GWO sont utilisées pour trouver une solution efficace pour un problème donné. Chaque phase correspond aux actions des loups gris de la meute, où les loups leaders correspondent à la meilleure solution trouvée jusqu'à présent, tandis que les autres représentent les solutions restantes dans la population. Le pseudo-code de l'approche GWO est présenté dans l'Algorithme 3.

5.3.2 Algorithme de shuffled frog leaping

L'algorithme SFLA (Shuffled Frog-Leaping Algorithm) est une métaheuristique de recherche coopérative basée sur la population, conçue en 2003 par Eusuff et Lansey [179]. Il tire son inspiration du comportement social des grenouilles vivant dans des habitats humides, où elles se déplacent en groupes pour optimiser leur recherche de nourriture en minimisant le nombre de tentatives. Cette stratégie naturelle a été adaptée avec succès à divers domaines technologiques, notamment l'optimisation multi-objectif, l'analyse de clusters, les technologies d'optimisation, ainsi que d'autres problèmes d'optimisation combinatoire [180].

En effet, cette métaheuristique combine les avantages de l'algorithme mémétique (MA), qui s'inspire de la génétique, avec ceux de l'algorithme PSO, centré sur le comportement social. L'objectif principal des grenouilles dans cette approche est d'explorer la zone de recherche pour trouver la nourriture la plus abondante tout en minimisant leurs efforts. Pour atteindre cet objectif, le SFLA utilise un modèle abstrait de population virtuelle basé sur la recherche locale et l'échange d'informations globales. L'algorithme fonctionne avec une population de grenouilles réparties en plusieurs communautés appelées "memeplexes", où chaque grenouille possède son propre ensemble de données. Dans chaque sous-population (memeplex), une stratégie de recherche locale vise à améliorer les posi-

tions des grenouilles les moins performantes en utilisant la position de la meilleure grenouille locale ou globale [179, 181].

Le processus de l'algorithme SFLA commence par l'initialisation des paramètres. Ensuite, la valeur de fitness de chaque grenouille est calculée après la génération initiale aléatoire de la population de grenouilles dans l'espace des décisions. Par la suite, la population est triée en fonction des valeurs de fitness, en plaçant le meilleur membre en première position. Cette population triée est ensuite divisée en memeplexes selon l'équation (5.7), et chaque memeplex subit un processus évolutif memétique (recherche locale). Au cours de cette étape, la grenouille la moins performante X_w saute vers la meilleure solution X_b, conformément à l'équation (5.8).

$$Y_k = [(X_i)_k | X_i = X(k + m * (i - 1)), i = 1, \ldots, n], k = 1, \ldots, m \tag{5.7}$$

où m désigne le nombre de memeplexes et n le nombre de membres dans chaque memeplex. Il est assuré que les membres sont répartis de manière équilibrée parmi les memeplexes. Supposons que $m = 3$; le premier membre va au premier memeplex, le deuxième membre au deuxième memeplex, le troisième membre au troisième memeplex, le quatrième membre au premier memeplex, et ainsi de suite.

$$\begin{aligned} S &= r * |X_b - X_w| \\ X_{w'} &= X_w + S \end{aligned} \tag{5.8}$$

Une fois qu'un certain nombre d'itérations sont complétées, les memeplexes sont regroupés et mélangés. Cette étape favorise l'échange d'informations entre les grenouilles provenant de différents memeplexes, contribuant ainsi à atteindre une position optimale. À moins que les critères de terminaison ne soient satisfaits, le processus d'itération reprend en réorganisant la population dans un ordre décroissant de performance. Les étapes décrites ci-dessus sont présentées en pseudo-code dans l'Algorithme 4.

5.3.3 Algorithmes proposés

Malgré l'abondante littérature et les résultats obtenus dans le domaine de l'ordonnancement multi-objectif des tâches dans les systèmes Cloud, plusieurs préoccupations demeurent encore non explorées [43]. Une direction de recherche prometteuse consiste à combiner diverses heuristiques ou métaheuristiques afin d'exploiter leurs points forts et de surmonter leurs limitations. Notre hypothèse de base consiste à ce que l'optimisation de l'ordonnancement dans le Cloud peut être améliorée en utilisant des algorithmes d'intelligence en essaim. À cette fin, nous avons élaboré deux algorithmes d'optimisation hybrides distincts.

Dans un premier temps, nous adoptons l'algorithme de pollinisation des fleurs, en renforçant la phase d'exploration avec l'optimiseur de loups gris et en intégrant les opérateurs de croisement des algorithmes génétiques [182]. Cette approche vise à prévenir la convergence prématurée vers des optima locaux tout en équilibrant de manière appropriée l'ex-

Algorithm 4 Pseudo-code standard de l'algorithme SFLA

1: Initialiser la taille de la population, le nombre M de memeplexes, le nombre d'itérations globales N et locales P
2: Générer une population aléatoire et évaluer chaque solution
3: Trier la population et déterminer la meilleure solution X_g
4: Partitionner la population en M memeplexes
5: *Recherche locale :* Pour chaque memeplex $i = 1 : M$, répéter P itérations
6: Déterminez la meilleure solution X_b et la pire solution X_w
7: Calculer $X_w^{'}$ à partir de X_b (appliquer l'équations 5.8)
8: **if** $X_w^{'}$ est meilleur que X_w **then**
9: Remplacer X_w par $X_w^{'}$
10: **else**
11: Calculer $X_w^{'}$ à partir de X_g (appliquer l'équations 5.8 en remplaçant X_b par X_g)
12: **if** $X_w^{'}$ est meilleur que X_w **then**
13: Remplacer X_w par $X_w^{'}$
14: **else**
15: Générer aléatoirement $X_w^{'}$ et remplacer X_w par $X_w^{'}$
16: **end if**
17: **end if**
18: Regrouper les M memeplexes pour reconstituer la population
19: Aller à l'étape 3 si le critère d'arrêt global n'est pas atteint
20: Retourner X_g

ploration et l'exploitation. Le pseudo-code de notre première approche hybride multi-objectif est présenté dans l'Algorithme 5.

L'algorithme commence par définir ses paramètres de contrôle, notamment la probabilité de commutation, la taille de la population, le nombre maximal d'itérations, le nombre de tâches soumises et le nombre de machines virtuelles disponibles. Aux étapes 2 et 3, un ensemble aléatoire de solutions est généré, correspondant à la population initiale. La création de chaque solution tient compte d'un ensemble spécifié de tâches-utilisateurs et de machines virtuelles, utilisant un codage qui sera détaillé ultérieurement. Initialement, ces tâches sont réparties de manière aléatoire parmi les différentes machines virtuelles disponibles dans le centre de données. Par la suite, les solutions sont ajustées en fonction de la stratégie utilisée. Une fois que toutes les solutions initiales sont générées, elles sont évaluées individuellement en utilisant une fonction de fitness. La solution la plus performante est alors identifiée et sauvegardée en tant que S_α. À chaque étape d'itération de la boucle interne (lignes 4 à 33), pour chaque solution, une nouvelle solution est générée en utilisant soit la pollinisation globale, soit la pollinisation locale, en fonction d'une probabilité de commutation p. La nouvelle solution est examinée et comparée à la solution actuelle, puis elle est incorporée dans la population si elle est meilleure. Sinon, l'algorithme propose d'utiliser un opérateur de croisement d'algorithme évolutionnaire (ligne 18) comme dans MODE-RMO [183, 184] pour améliorer le processus d'exploitation et découvrir de meilleures solutions dans le voisinage en combinant la solution actuelle avec le vecteur de référence à une itération donnée.

Algorithm 5 Pseudo-code de la première approche hybride proposée

Initialiser les paramètres de l'algorithme : taille de la population (n), nombre maximum d'itérations et probabilité de commutation $p \in [0, 1]$
Générer la population initiale S_i, $i = 1, 2, \ldots, n$
Trouver la meilleure solution S_α dans la population initiale
while critère d'arrêt **do**
  for each élément dans la population **do**
    if $rand() < p$ **then**
      Tirer une taille de pas L selon une distribution de Lévy
      Effectuer une pollinisation globale en utilisant éq. (4.1)
    else
      Tirer ε selon une distribution uniforme dans $[0, 1]$
      Choisir aléatoirement deux machines virtuelles S_j et S_k parmi toutes les solutions
      Effectuer une pollinisation locale en utilisant éq. (4.6)
    end if
    Évaluer la nouvelle solution $S_i^{'}$
    if la nouvelle solution générée est meilleure **then**
      Remplacer S_i par $S_i^{'}$
    else
      Appliquez l'opération de croisement sur l'élément actuel S_i de la population en utilisant l'équation (5.9), puis évaluez la nouvelle solution $S_i^{'}$.
      if la nouvelle solution générée est meilleure **then**
        Remplacez S_i par la nouvelle solution générée $S_i^{'}$
      end if
    end if
  end for
  Trier la population par ordre de valeur de fitness
  Mettre à jour le meilleur élément S_α
  Sélectionner le deuxième meilleur élément S_β
  Sélectionner le troisième meilleur élément S_δ
  Déterminer le pire élément S_w
  Appliquer l'équation (5.6) à la pire solution candidate
  if la nouvelle solution générée meilleure **then**
    Remplacer la pire solution S_w par la nouvelle solution $S_w^{'}$
  end if
end while
Retourner S_α en tant que meilleure solution dans l'espace de recherche

L'opérateur de croisement implémenté nécessite un vecteur de référence R et un vecteur cible T dans un espace à D-dimensions, où D représente la taille de la charge de travail. Le vecteur de référence est sélectionné comme étant la meilleure solution de la population actuelle. L'opération de croisement est définie par l'équation (5.9), avec $i = 1, 2, 3, \ldots, D$.

$$T_i^{'} = \begin{cases} T_i & \text{if } (rand \leq CR \text{ or } i == s) \\ R_i & \text{otherwise} \end{cases} \tag{5.9}$$

où la constante de taux de croisement CR est une valeur comprise entre

$[0,1]$, s et *rand* sont des nombres aléatoires générés dans $[1,D]$ et $[0,1]$, respectivement.

Ensuite, l'algorithme propose d'utiliser la stratégie de l'optimiseur de loups gris pour améliorer la solution la plus défavorable (lignes 24 à 29). Selon cette stratégie, les trois meilleures solutions sont sélectionnées en fonction de leurs valeurs de fitness, et une nouvelle solution est calculée en utilisant l'équation (5.6), en faisant varier la valeur de a à chaque itération de l'algorithme. Cela renforce le processus d'exploration pour trouver d'autres solutions potentielles dans l'espace, tout en garantissant que la qualité des solutions s'améliore progressivement et qu'elle ne reste pas piégée dans un optimum local. Enfin, ce processus est répété jusqu'à ce que l'algorithme converge ou atteigne le nombre maximal d'itérations. Il convient de noter que les tâches arrivées ne sont pas immédiatement attribuées aux machines virtuelles ; une phase de pré-simulation du processus de mappage utilisant la version hybride proposée est nécessaire pour trouver la VM appropriée pour chaque tâche avant de l'envoyer au broker.

Dans notre seconde contribution, nous apportons des modifications substantielles à l'algorithme précédemment décrit. L'objectif principal de ces ajustements est d'exploiter de manière plus efficiente l'approche déjà présentée en vue de résoudre des problèmes d'ordonnancement à optimisation multi-objectif.

L'algorithme hybride que nous avons décrit précédemment repose en grande partie sur l'orientation de la recherche vers les positions optimales des membres de la population, notamment alpha, beta et delta. Cette stratégie encourage généralement les individus à explorer l'espace de recherche aux côtés de leurs leaders. Afin d'optimiser l'exploration de l'espace solution tout en évitant une exploration excessive à proximité des leaders, nous avons introduit la stratégie du "memeplex", inspirée de l'algorithme de saut de grenouille. Cette dernière est incorporée à notre algorithme d'optimisation hybride, où chaque memeplex possède sa propre équipe de dirigeants, composée des agents alpha, beta et delta les plus performants [185].

Pour illustrer cette stratégie, supposons une taille de population $P = 20$ et un nombre de memeplexes $m = 4$. Les quatre premières solutions (S1 à S4) sont assignées séquentiellement aux memeplexes (m1 à m4). Ensuite, les quatre solutions suivantes (S5 à S8) sont attribuées aux memeplexes de la même manière, et ce processus se poursuit jusqu'à ce que tous les membres de la population soient répartis dans les memeplexes. Cette stratégie assure une répartition équilibrée de la population en groupes distincts. La Figure 5.1 illustre la formalisation mathématique de cette stratégie de séparation en memeplexes, décrite dans l'équation (5.7).

Nous pouvons donc considérer cette deuxième contribution comme une version distribuée de la première contribution de ce chapitre. Dans cette approche, la population de solutions candidates est répartie et organisée en memeplexes, qui sont de petites sous-populations indépendantes, alignées sur le modèle utilisé dans la SFLA. À l'intérieur de chaque memeplex, une copie de l'algorithme 5, de l'étape 4 à l'étape 33, est mise en œuvre. Le pseudo-code de cette dernière approche hybride multi-objectif est présenté dans l'algorithme 6.

Algorithm 6 Pseudo-code de la deuxième approche hybride proposée

Initialiser les paramètres de l'algorithme : taille de la population (F), probabilité de commutation $p \in [0,1]$, le nombre de memeplexes (M), le nombre maximal d'itérations globales (N) et locales (P)
Générer la population initiale $S_i,\ i = 1,\ 2,\ \ldots,\ F$
Trier la population et déterminer la meilleure solution S_g
Partitionner la population en M memeplexes
Pour chaque memeplex $M_j,\ j = 1,\ 2,\ \ldots,\ M$, répéter P itérations
Attribuez l'élément 1^{er} du memeplex M_j en tant que S_α
for each élément dans le memeplex M_j **do**
  if $rand() < p$ **then**
    Tirer une taille de pas L selon une distribution de Lévy
    Effectuer une pollinisation globale en utilisant éq. (4.1)
  else
    Tirer ε selon une distribution uniforme dans $[0,1]$
    Choisir aléatoirement deux éléments S_j et S_k parmi toutes les solutions du memeplex M_j
    Effectuer une pollinisation locale en utilisant éq. (4.6)
  end if
  Évaluer la nouvelle solution $S_i^{'}$
  if la nouvelle solution générée est meilleure **then**
    Remplacer S_i par $S_i^{'}$
  else
    Appliquer l'opération de croisement sur l'élément actuel S_i du memeplex M_j en utilisant éq. (5.9), puis évaluez la nouvelle solution $S_i^{'}$
    if la nouvelle solution générée est meilleure **then**
      Remplacez S_i par la nouvelle solution générée $S_i^{'}$
    end if
  end if
end for
Trier les élément du memeplex M_j par ordre de valeur de fitness
Mettre à jour le meilleur élément S_α
Sélectionner le deuxième meilleur élément S_β
Sélectionner le troisième meilleur élément S_δ
Déterminer le pire élément S_w
Appliquer éq. (5.6) à la pire solution candidate
if la nouvelle solution générée meilleure **then**
  Remplacer la pire solution S_w par la nouvelle solution $S_w^{'}$
end if
Regrouper les M memeplexes pour reconstituer la population
Aller à l'étape 3 si le critère d'arrêt global n'est pas atteint
retourner S_g en tant que meilleure solution dans l'espace de recherche

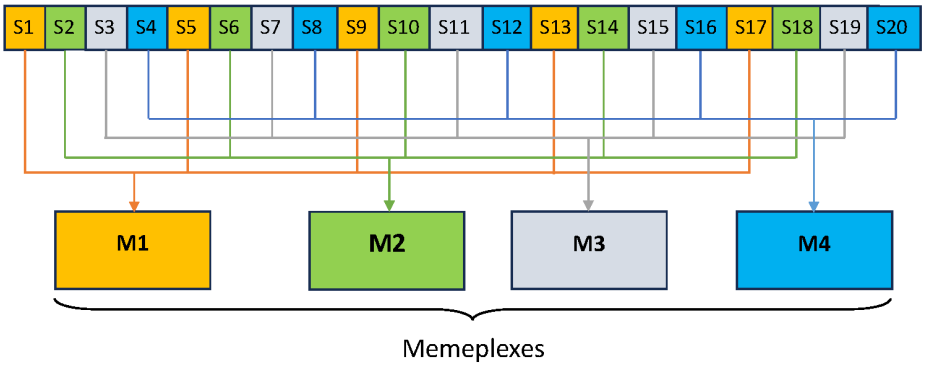


FIGURE 5.1 – *Stratégie de répartition en memeplexes*

5.3.4 Critères d'évaluation de QoS

L'objectif fondamental de l'ordonnancement est de répondre efficacement aux demandes des utilisateurs en allouant les ressources appropriées qui optimisent un ou plusieurs objectifs. Dans la littérature, les travaux ont principalement porté leur attention sur des critères axés sur la qualité de service du côté des utilisateurs finaux, tels que le makespan et le coût d'exécution [155]. Cependant, d'autres paramètres de QoS pertinents du fournisseur de services, susceptibles d'influencer la performance globale du système, sont souvent négligés [176].

Il est essentiel de prendre en compte à la fois les besoins des consommateurs et des fournisseurs dans la conception et la mise en œuvre des services Cloud, afin de garantir que le service soit rentable pour le fournisseur tout en étant économique pour le consommateur. Cela nécessite souvent un équilibre délicat et des compromis des deux côtés. Ainsi, l'objectif de cette étude est de prendre en considération ces objectifs en évaluant des mesures telles que le makespan, le coût, l'utilisation des ressources, le débit et le degré de déséquilibre, qui seront définis plus en détail ci-dessous.

Makespan : comme expliqué dans la section 4.3.5 du chapitre précédent, le makespan correspond au temps total nécessaire à l'accomplissement d'un ensemble de tâches. Il est calculé conformément à l'équation (4.15).

Coût : représente le prix total de l'exécution de l'application utilisateur. Il tend à être la mesure la plus mesurable aujourd'hui, mais il est important d'exprimer ce coût en fonction des caractéristiques des ressources fournies. L'utilisateur vise à réduire à la fois le coût et la durée du makespan pour obtenir une réponse rapide au moindre coût. Nous tenons à préciser que, le coût d'exécution est déterminé de la même façon que dans le chapitre précédent, selon l'équation (4.16).

Utilisation des ressources : elle mesure l'efficacité et l'efficience de l'utilisation des ressources d'un système. Cette métrique revêt une importance particulière pour les fournisseurs de services Cloud, car elle influe directement sur leur rentabilité. Dans le cadre de cette étude, nous avons évalué les performances des approches proposées en termes d'utilisation des

ressources, mesurée par le taux moyen d'utilisation des ressources (RUR) selon l'équation suivante [186, 187, 188, 152, 189] :

$$RUR = \frac{\sum_{j=1}^{m} CT_j}{Makespan * m} \tag{5.10}$$

où m représente le nombre de machines et CT_j est le temps d'achèvement de la *j-ime* machine virtuelle après l'exécution de sa dernière tâche. Un taux plus élevé indique que le système tire un meilleur parti de ses ressources, tandis qu'un taux plus bas suggère que les ressources soient sous-utilisées.

Degré de déséquilibre : il s'agit d'une métrique qui évalue le déséquilibre de la répartition de la charge de travail dans le Cloud entre les différentes machines virtuelles en fonction de leurs capacités [190]. Son objectif est de garantir une répartition équilibrée de la charge de travail et d'éviter les goulets d'étranglement dans le système, de sorte qu'aucune ressource ne soit surchargée tandis que d'autres sont sous-chargées ou inactives. Le degré de déséquilibre est calculé comme suit :

$$Deg\_Déséquilibre = \frac{VM\_CT_{max} - VM\_CT_{min}}{VM\_CT_{avg}} \tag{5.11}$$

où VM_CT_{max} et VM_CT_{min} représentent respectivement le temps d'achèvement le plus long et le plus court obtenu. VM_CT_{avg} désigne le temps moyen d'achèvement de toutes les machines virtuelles. Un *Deg_Déséquilibre* élevé indique un déséquilibre plus marqué entre les ressources, ce qui peut avoir un impact négatif sur les performances du système. À l'inverse, un *Deg_Déséquilibre* plus faible montre une répartition plus équilibrée des ressources, ce qui peut conduire à de meilleures performances.

Débit : il mesure la rapidité avec laquelle un système ou un processus particulier peut accomplir les tâches qui lui sont confiées. Il est généralement exprimé en termes du nombre de tâches terminées ou de la quantité de données traitées dans un laps de temps donné. Une valeur de débit élevée indique qu'un système ou un processus peut traiter une grande quantité de données en peu de temps. Cela est généralement considéré comme une caractéristique souhaitable, car il peut entraîner des temps de réponse plus rapides et une productivité accrue pour l'ensemble du système. À l'inverse, une valeur de débit faible peut refléter l'incapacité d'un système ou d'un processus à suivre les demandes assignées, ce qui peut entraîner des retards et une baisse de la productivité. Le débit est calculé selon la formule suivante [191] :

$$Débit = \frac{Nombre\ de\ tâches}{Makespan} \tag{5.12}$$

5.3.5 Représentation de la solution

L'efficacité de la conception d'une approche basée sur les métaheuristiques dépend largement de la représentation pertinente d'une solution,

un élément crucial dans la détermination du succès de l'algorithme. En effet, la manière dont les solutions sont traitées par les opérateurs de recherche et lors de l'étape d'évaluation est fortement influencée par cette représentation. Plusieurs représentations peuvent être envisagées pour un problème donné. Dans notre cas d'étude, nous avons choisi d'adopter le vecteur de valeurs discrètes pour notre solution, où chaque tâche T_i est assignée à une machine virtuelle VM_j dans le Cloud IaaS. La taille de la solution est déterminée par le nombre de tâches constituant la charge de travail. Une version simplifiée de la représentation est illustrée dans la Figure 5.2. En termes simples, cela signifie que la tâche T_1 est assignée à la VM_4, la tâche T_2 à la VM_1, et ainsi de suite.

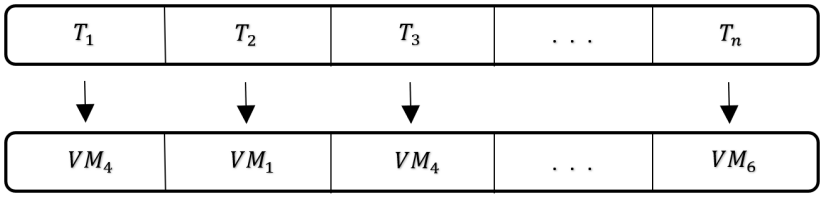


FIGURE 5.2 – *Représentation simplifiée d'une solution*

5.3.6 Fonction fitness

L'étape d'évaluation revêt également une importance cruciale dans la conception d'un algorithme d'optimisation. Elle permet de guider la méthode vers les solutions appropriées de l'espace de recherche. Dans le cadre d'un problème d'optimisation multi-objectif, plusieurs objectifs sont généralement impliquées. Par conséquent, l'évaluation consiste à attribuer un vecteur de valeurs à chaque solution, la taille de ce vecteur correspondant au nombre d'objectifs considérés. Chaque élément du vecteur reflète la qualité de la solution par rapport à la QoS correspondante (voir la section 5.3.4). Pour notre problématique, nous avons opté pour une fonction de fitness d'agrégation par somme pondérée, largement répandue dans la littérature en raison de sa simplicité et de son adaptabilité à divers objectifs numériques. La fonction multi-objectif que nous avons utilisée prend en compte les cinq objectifs mentionnés précédemment, comme définie dans l'équation (5.13).

$$\begin{aligned} \textit{Fonction fitness} = w_1 * \textit{Makespan} + w_2 * \textit{Coût} + w_3 * \textit{RUR} + \\ w_4 * \textit{Deg\_Déséquilibre} + w_5 * \textit{Débit} \end{aligned} \tag{5.13}$$

où w_1, w_2, w_3, w_4 et w_5 représentent les facteurs de pondération reflétant l'importance ou les exigences de chaque critère. Dans cette étude, tous les objectifs ont été attribués un poids égal de $0,2$ pour garantir l'équité et éviter tout biais envers un objectif spécifique. Par conséquence, cette dernière servira de fonction d'aptitude pour les deux algorithmes d'optimisation hybrides proposés.

Il convient de noter que les valeurs des cinq métriques de qualité de service à optimiser pour chaque solution sont normalisées dans l'intervalle $[0, 1]$ en utilisant la méthode de normalisation min-max, comme indiqué dans l'éq. 4.19.

Finalement, la fonction objectif F est définie comme suit :

$$F = Minimiser(Fonction\ fitness) \tag{5.14}$$

5.4 Résultats de simulation et discussions

Dans cette section, nous présenterons l'évaluation expérimentale des algorithmes d'optimisation hybrides proposés. Nous détaillerons les différents paramètres relatifs à la charge de travail de l'utilisateur, le modèle d'infrastructure IaaS du fournisseur utilisé, ainsi que les paramètres spécifiques associés aux algorithmes comparés.

5.4.1 Environnement de simulation

Afin d'investiguer et d'évaluer l'efficacité des l'algorithmes proposés, nous avons sélectionné un ensemble de jeux de données de benchmark bien étudiés issus de la littérature [192, 193]. Ces jeux de données couvrent une charge de travail Cloud réaliste ainsi que deux ensembles de données de référence standard couramment utilisés pour l'évaluation des performances dans les systèmes parallèles et distribués, à savoir, le High-Performance Computing Center North (HPC2N) et le jeu de données Google Cloud Jobs (GoCJ).

Ensemble de données synthétiques

Les données synthétiques désignent des charges de travail générées artificiellement utilisées comme modèle pour évaluer un système en cours d'étude. Les chercheurs de la communauté de l'ordonnancement utilisent des ensembles de données synthétiques comportant des tâches ayant des temps de traitement différents afin d'évaluer l'efficacité de leurs approches [162, 194, 195, 176, 110, 153, 172]. De même, à des fins de comparaison, nous avons également utilisé des charges de travail synthétiques similaires et conformes à celles utilisées dans [171]. L'ensemble de données synthétiques est généré en utilisant différentes proportions et distributions de charges de travail de petite, moyenne, grande et très grande taille, mesurées en MI, comme indiqué dans le Tableau 5.1.

Type de tâche	MI	Distribution
Très petite	200	35%
Petite	1 000	40%
Moyenne	5 000	5%
Grande	15 000	15%
Très grande	45 000	5%

Table 5.1 – *Type de tâche du jeu de données synthétiques*

Ensemble de données GoCJ

Hussain et Aleem [192] ont mis à disposition un ensemble de données Cloud réel dérivé des comportements de charge observés dans les traces de clusters Google. Un échantillon original du jeu de données GoCJ est formulé et archivé dans le référentiel de données Mendeley [196], afin de permettre à d'autres chercheurs de comparer les performances de leurs nouveaux algorithmes en utilisant un benchmark open-source. GoCJ est constitué de différents fichiers contenant des données sur la taille des tâches en termes de MI. Chaque fichier contient un ensemble différent de lignes, équivalent au nombre de tâches, indiquant la longueur des tâches en MI. Ces valeurs varient de 15 000 MI à 900 000 MI. Ce jeu de données est composé de cinq types de tâches avec des proportions différentes, basées sur l'analyse des traces des clusters Google, comme présenté dans le Tableau 5.2.

Type de tâche	MI	Distribution
Petite	15 000 - 55 000	20%
Moyenne	59 000 - 99 000	40%
Grande	101 000 - 135 000	30%
Très grande	150 000 - 337 500	6%
Énorme	525 000 - 900 000	4%

TABLE 5.2 – *Type de tâche du jeu de données GoCJ*

Ensemble de données HPC2N

Le High Performance Computing Center North (HPC2N) est un centre national de calcul scientifique et parallèle qui fait partie de l'Infrastructure Nationale de Calcul Suédoise (SNIC). Il met à disposition des chercheurs et des organisations des ressources de calcul haute performance ainsi que des services dans divers domaines, notamment l'ingénierie, les sciences sociales et de la vie. Le centre propose des formations et un support aux chercheurs sur la manière d'utiliser les ressources, ainsi qu'un accès à un large éventail d'outils logiciels pour l'analyse de données et la simulation. Conformément aux recommandations, nous avons utilisé HPC2N-2002-2.2-cln.swf dans nos expériences, qui a pris trois ans et demi de relevés comptables du HPC2N en Suède [193].

Nos simulations sont réalisées en utilisant CloudSim, une plateforme largement adoptée par la communauté de recherche pour la modélisation et la simulation des infrastructures Cloud à grande échelle. Les paramètres des simulations ont été soigneusement configurés pour représenter un environnement Cloud hétérogène.

Dans un souci de réalisme accru, nous avons basé nos simulations sur les caractéristiques des instances de machines virtuelles fournies par Amazon EC2 [197]. Chaque instance de machine virtuelle possède ses propres capacités de traitement et est associée à un coût spécifique. Les caractéristiques de ces VM sont indiquées dans la Table 5.3 [198].

Type	CPU	MIPS	Coût ($/h)	BW(Gb/s)
a1.medium	1	4400	0.0255	10
a1.large	2	8800	0.051	10
a1.xlarge	4	17 600	0.102	10
t3.xlarge	4	17 600	0.1664	5
m5n.2xlarge	8	35 200	0.476	25
m5zn.3xlarge	12	52 800	0.991	25

TABLE 5.3 – *Types de machines virtuelles utilisées*

5.4.2 Configuration expérimentale

Les résultats et les performances des l'algorithmes proposés sont comparés à des stratégies d'optimisation bien connues et bien établies, à savoir TSMGWO [171], GGWO [174], FPA [152] et LPGWO [169]. Ces algorithmes couvrent à la fois des techniques récemment proposées telles que OBL et chaotic mapping [199], ainsi que les optimiseurs relativement les plus utilisés dans le domaine tels que GA, FPA et GWO.

Pour des raisons de comparaison, nous avons évalué et enregistré les résultats en fonction des performances des algorithmes évalués sur 10 exécutions indépendantes, puis la moyenne des résultats est rapportée. Les paramètres de contrôle des algorithmes d'ordonnancement comparés sont présentés dans la Table 5.4. Les valeurs des paramètres de notre approche ont été déterminées expérimentalement sur les trois ensembles de données de charge de travail scientifiques, avec des valeurs variées. Les valeurs sélectionnées ont permis d'obtenir les meilleurs résultats globaux. Pour garantir l'équité des expériences, la population initiale de tous les algorithmes est fixée à 100. De plus, il est important de souligner que les paramètres utilisés dans notre étude pour les algorithmes comparés sont choisis de manière à être représentatifs et conformes à ceux utilisés dans la littérature [174], [152], [171] et [169].

Paramètre	Valeur
Taille de population	100
Nombre de memeplexes	10
p	0.8
λ	1.5
CR	0.7

TABLE 5.4 – *Paramétrage des algorithmes*

5.4.3 Discussion des résultats

Cette section se consacre à la discussion des résultats expérimentaux obtenus à partir des approches hybrides d'ordonnancement des tâches que nous avons proposées. Dans ce cadre, une application est représentée sous la forme d'une collection de tâches pouvant être exécutées simultanément. Les expériences menées visent à évaluer l'efficacité de nos propositions en générant un ensemble de solutions d'équilibre entre différentes métriques

de qualité de service. Comme indiqué précédemment, nous comparons les performances des algorithmes proposés avec quatre algorithmes métaheuristiques bien connus, à savoir FPA [152], LGWO [169], GGWO [174] et TSMGWO [171]. De plus, nous avons évalué nos approches en utilisant trois types de charges de travail scientifiques différents afin d'examiner l'impact de la charge de travail sur le comportement de nos algorithmes. Les résultats sont présentés sous forme de graphiques basés sur les métriques de performance considérées : makespan, coût, utilisation des ressources, degré de déséquilibre et débit. Nous notons que ces expérimentations ont été menées en tenant compte des spécificités de la plateforme Cloud Amazon EC2, en utilisant 80 machines virtuelles. Nous avons examiné les performances pour différentes tailles de tâches, à savoir 100, 200, 300, 400, 500 et 600, quel que soit l'ensemble de traces de charge de travail utilisé.

Résultats obtenus par rapport au makespan

Ensemble de données	Cloudlets	Méthode utilisée					
		TSMGWO	GGWO	FPA	LPGWO	Approche proposée 1	Approche proposée 2
Synthetic	100	3.662	3.864	3.226	3.196	3.06	2.914
	200	6.202	6.376	4.99	4.85	4.046	3.946
	300	8.618	8.828	7.268	7.056	5.724	5.212
	400	10.332	10.684	8.874	7.524	6.698	6.084
	500	12.063	12.403	10.23	8.443	7.943	7.18
	600	13.116	13.583	11.526	10.233	9.236	8.553
GoCJ	100	72.496	70.262	57.622	56.566	50.014	44.118
	200	125.156	139.43	114.896	83.386	76.333	64.306
	300	196.786	210.983	145.466	139.026	113.16	90.72
	400	206.27	232.076	151.713	142.483	122.966	113
	500	261.97	265.743	220.01	179.25	156.313	139.73
	600	319.823	339.4	267.19	256.93	173.243	162.71
HPC2N	100	197.726	185.128	151.23	146.626	141.55	123.154
	200	250.946	282.676	245.372	201.538	176.642	168.798
	300	359.334	377.766	328.042	284.832	227.906	202.128
	400	469.87	504.662	446.472	368.486	309.826	285.6
	500	605.724	677.95	619.432	511.474	385.088	367.954
	600	658.722	698.428	634.764	524.382	423.464	375.37

TABLE 5.5 – *Résultats du makespan des approches proposées et des algorithmes comparés*

Comme le montrent les figures 5.3, 5.4 et 5.5, qui représentent les résultats du temps de makespan obtenu pour les charges de travail synthétique, GoCJ et HPC2N respectivement, on peut constater que, pour tous les ensembles de données de référence identifiés, les algorithmes proposés donnent les meilleurs résultats et leurs performances restent systématiquement supérieures à celles des autres concurrents. Cela montre clairement que les algorithmes concurrents ne parviennent pas à atteindre un point optimal, ou même à s'en approcher. En revanche, les algorithmes propo-

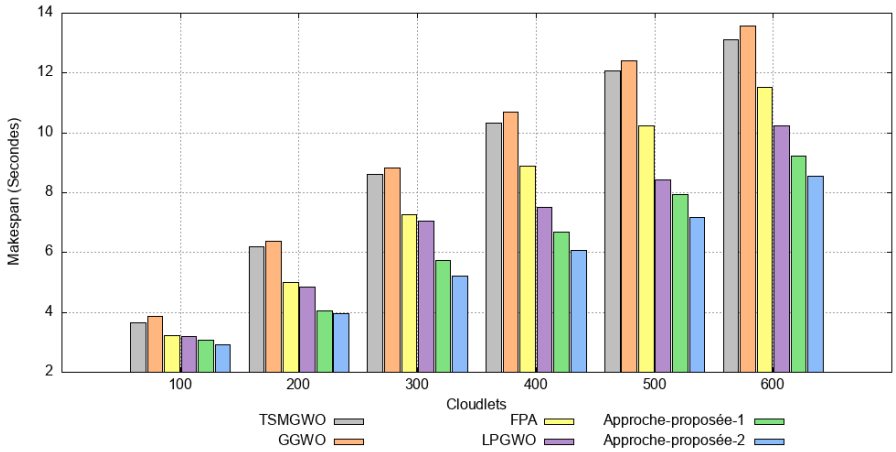


FIGURE 5.3 – *Résultats comparatifs en termes de makespan pour la charge de travail synthétique*

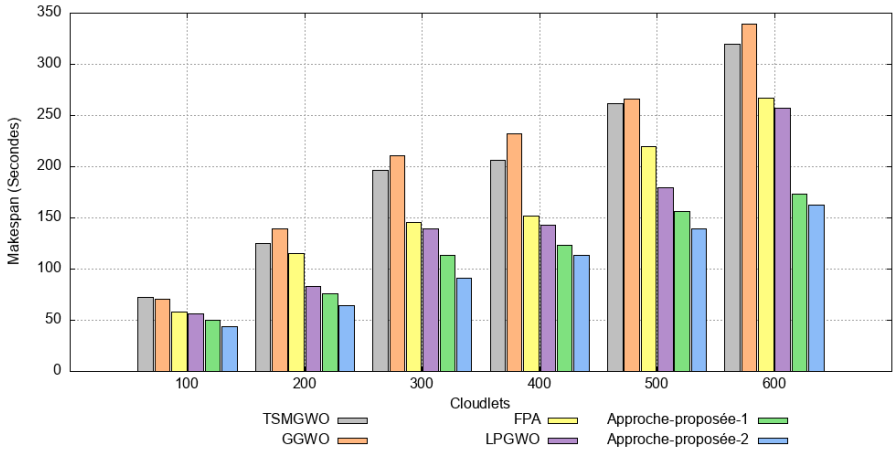


FIGURE 5.4 – *Résultats comparatifs en termes de makespan pour la charge de travail GoCJ*

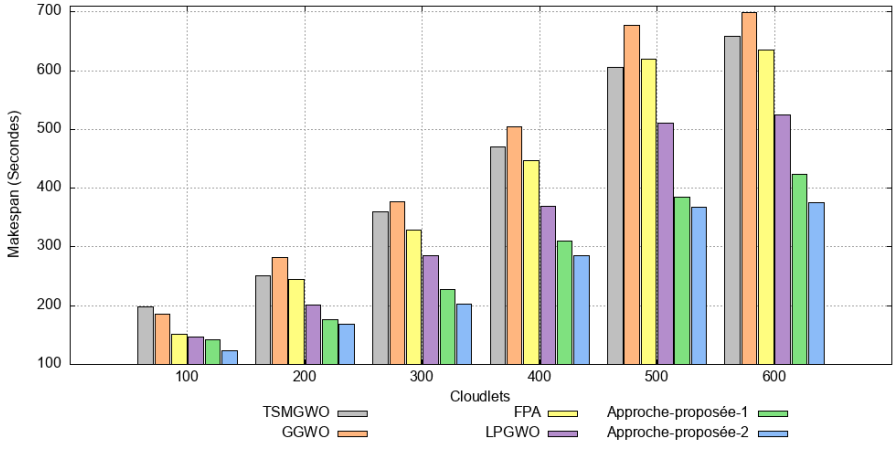


FIGURE 5.5 – *Résultats comparatifs en termes de makespan pour la charge de travail HPC2N*

sés produisent les meilleures valeurs de makespan sur toutes les instances testées avec différents nombres de tâches. Notons que notre deuxième proposition se distingue en termes de temps de makespan, suivie de près par

notre première proposition, qui demeure plus performante que les autres algorithmes comparés.

En examinant les résultats présentés dans la Table 5.5, nous pouvons constater que nos approches surpassent leurs concurrents directs avec des taux de réduction significatifs. Par exemple, pour la charge de travail synthétique, notre deuxième proposition parvient à réduire le makespan jusqu'à 43.05%. De même, avec les charge de travail HPC2N et GoCJ, ce taux peut atteindre respectivement 43.74% et. Nous pouvons également observer à partir du Tableau 5.5, que les résultats de performance des méthodes comparées diminuent considérablement à mesure que le nombre de tâches augmente. Par exemple, dans la charge de travail HPC2N, le makespan de notre deuxième algorithme proposé est de 123.154, 168.798, 202.128, 285.6, 367.954 et 375.37 pour 100, 200, 300, 400, 500 et 600 tâches, respectivement. Cependant, cela n'est pas surprenant, étant donné que les machines virtuelles ont une charge de travail plus élevée, ce qui influe sur la valeur du makespan lors du processus l'ordonnancement.

Résultats obtenus par rapport au coût

Ensemble de données	Cloudlets	Méthode utilisée					
		TSMGWO	GGWO	FPA	LPGWO	Approche proposée 1	Approche proposée 2
Synthetic	100	7.5	6.956	11.93	9.612	5.878	5.662
	200	15.246	12.934	20.982	17	12.676	12.048
	300	21.466	20.038	31.09	26.964	19.362	17.324
	400	27.882	27.004	39.34	38.042	24.634	23.966
	500	36.163	34.21	50.266	48.213	30.76	28.796
	600	45.18	43.603	54.226	50.98	37.143	35.693
GoCJ	100	184.078	170.408	250.896	214.644	149.738	147.688
	200	364.933	337.653	425.483	477.79	333.053	320.053
	300	523.796	519.796	719.926	749.48	502.903	486.743
	400	725.286	715.553	948.276	982.353	696.273	630.31
	500	963.503	891.636	1095.686	1193.666	858.186	835.33
	600	1272.543	1213.73	1396.29	1329.703	1121.083	1091.503
HPC2N	100	403.952	397.766	570.982	505.124	329.84	308.568
	200	669.938	578.01	807.208	773.696	562.058	505.098
	300	924.338	874.378	1250.448	1167.526	861.794	759.114
	400	1345.972	1254.244	1687.872	1724.206	1211.114	1106.078
	500	1902.332	1799.058	2349.61	2286.048	1720.19	1663.186
	600	2015.276	1785.584	2461.536	2338.638	1760.214	1651.796

TABLE 5.6 – *Résultats du coût des approches proposées et des algorithmes comparés*

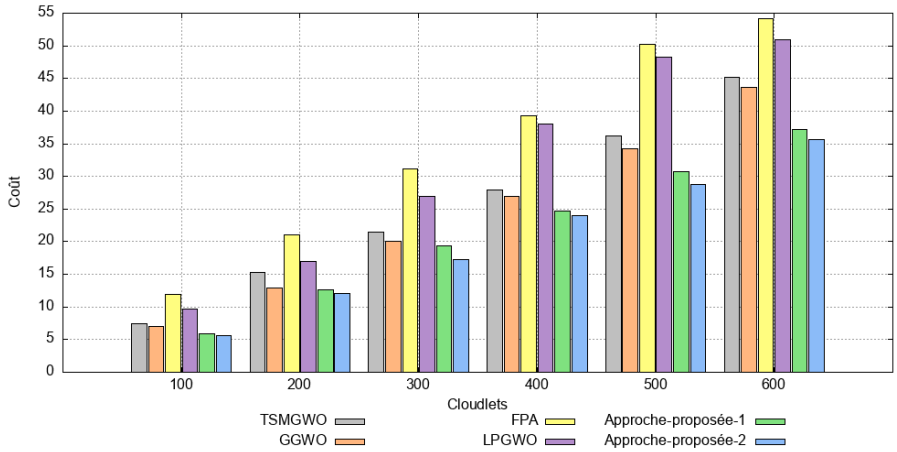


FIGURE 5.6 – *Résultats comparatifs en termes de coût pour la charge de travail synthétique*

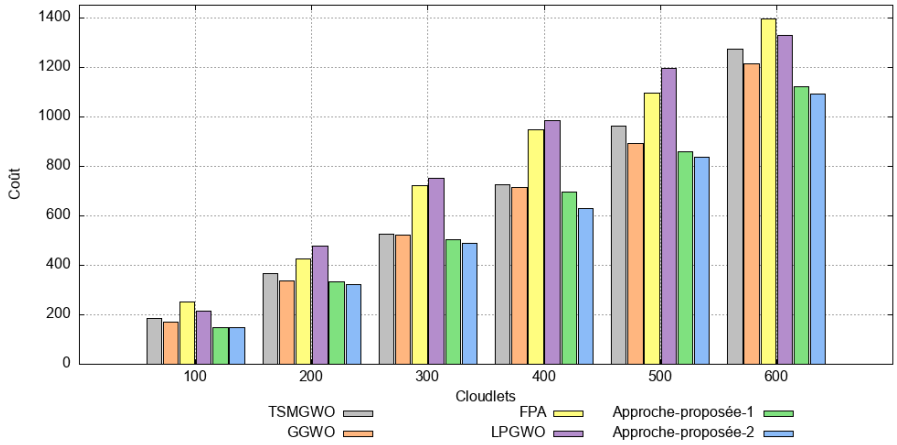


FIGURE 5.7 – *Résultats comparatifs en termes de coût pour la charge de travail GoCJ*

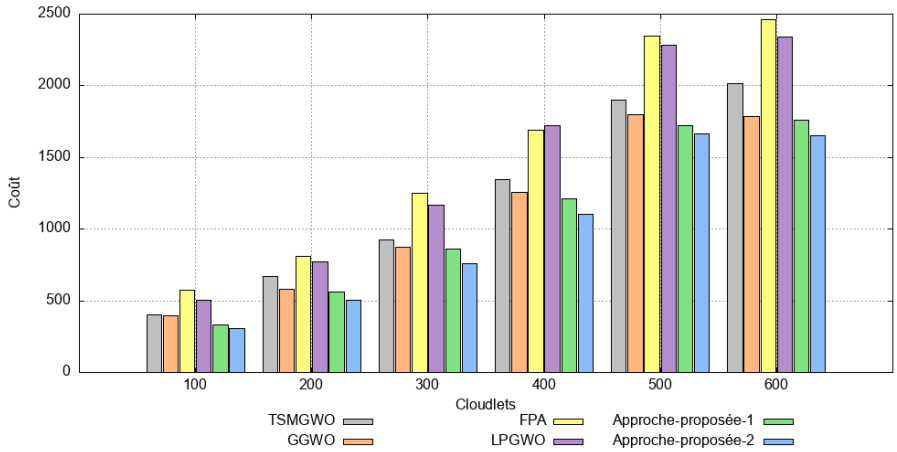


FIGURE 5.8 – *Résultats comparatifs en termes de coût pour la charge de travail HPC2N*

En comparant les résultats obtenus par les algorithmes TSMGWO, LPGWO, GGWO et FPA avec ceux de nos propositions en termes de coût, nous constatons, respectivement dans la Table 5.6 et sur les Figures 5.6, 5.7 et 5.8, que nos approches continuent de se démarquer en minimisant consi-

dérablement le coût d'exécution nécessaire pour accomplir les tâches soumises, et ceci pour toutes les instances examinées. Nous pouvons observer également, que la deuxième contribution a révélé son potentiel en matière d'optimisation des coûts tout en maintenant un makespan compétitif par rapport au premier algorithme que nous avons proposé. À titre d'exemple, dans le cas du jeu de données HPC2N, notre approche basée sur le comportement des memeplexes atteint des solutions qui sont de 22.42 %, 24.60 %, 38.91 % et 45.95 % plus efficaces en termes de coût que celles fournies par les algorithmes GGWO, TSMGWO, LPGWO et FPA respectivement. Cette supériorité est manifeste, car les algorithmes concurrents ne parviennent pas à trouver le bon équilibre entre les deux métriques, le makespan et le coût, qui sont bien connues pour leur nature conflictuelle.

Résultats obtenus par rapport au taux d'utilisation des ressources

Ensemble de données	Cloudlets	Méthode utilisée					
		TSMGWO	GGWO	FPA	LPGWO	Approche proposée 1	Approche proposée 2
Synthetic	100	16.032	16.284	17.608	17.538	28.528	31.022
	200	20.186	20.588	23.934	22.918	39.112	43.964
	300	21.158	21.578	25.94	25.674	43.498	48.338
	400	24.09	24.744	26.162	26.422	47.122	51.358
	500	23.976	24.71	30.816	29.663	48.666	56.14
	600	27.096	27.29	30.22	31.543	49.1	54.72
GoCJ	100	20.294	20.898	23.424	23.67	40.982	47.298
	200	23.846	22.323	26.566	29.43	48.29	56.446
	300	25.29	23.913	32.703	28.633	50.63	61.293
	400	27.696	27.953	35.87	33.716	55.913	61.913
	500	28.666	30.03	34.696	35.043	55.813	62.44
	600	27.803	30.163	34.25	36.003	58.54	64.446
HPC2N	100	16.828	18.084	20.62	20.798	35.702	42.804
	200	19.728	20.852	22.576	23.91	42.59	47.438
	300	24.084	25.436	26.432	26.528	46.484	53.234
	400	25.534	26.066	28.612	28.724	47.948	54.276
	500	27.076	27.292	28.84	29.574	50.994	56.564
	600	25.308	26.864	29.126	30.81	49.08	57.616

TABLE 5.7 – *Résultats du taux d'utilisation des ressources des approches proposées et des algorithmes comparés*

Les figures 5.9, 5.10 et 5.11, qui illustrent le taux d'utilisation des ressources lors de l'affectation de différents cloudlets aux VMs, montrent que les algorithmes proposées offrent de meilleures taux de performances que le TSMGWO, le LPGWO, le GGWO et l'algorithme FPA, tant pour les charges de travail synthétiques que pour les charges de travail standard. De plus, la deuxième proposition, qui intègre le comportement mémétique du SFLA, a démontré une performance supérieure par rapport à la première proposition.

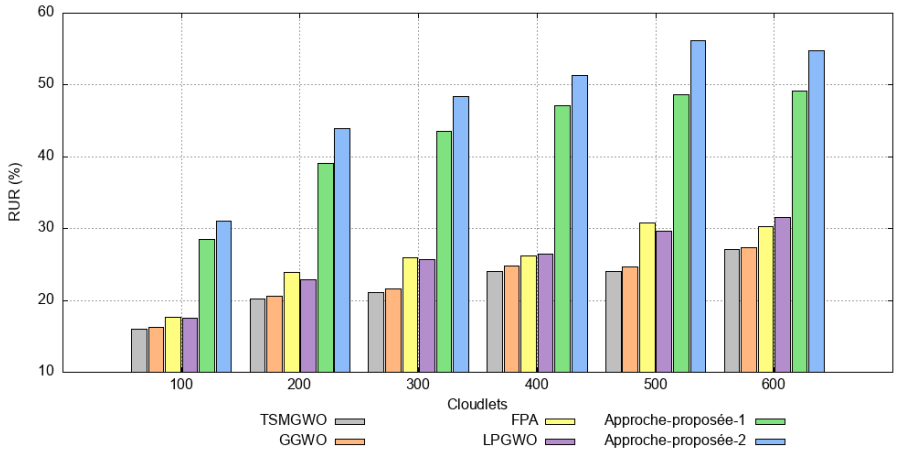


FIGURE 5.9 – *Résultats comparatifs en termes de taux d'utilisation des ressources pour la charge de travail synthétique*

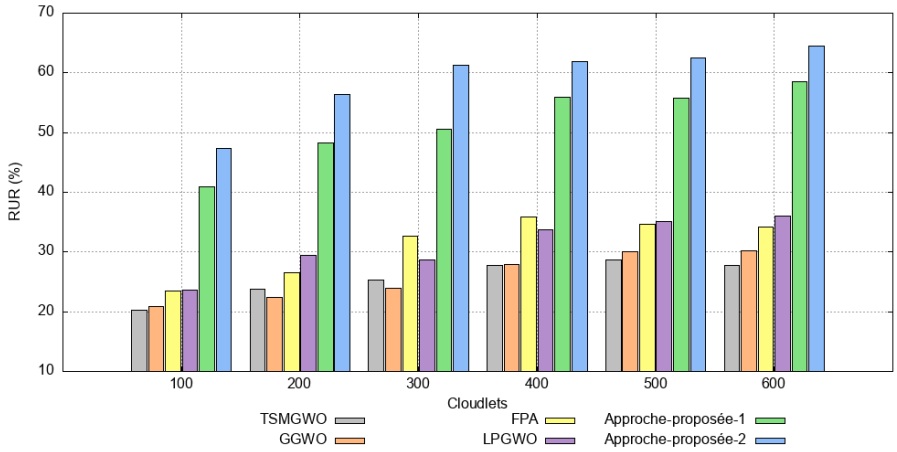


FIGURE 5.10 – *Résultats comparatifs en termes de taux d'utilisation des ressources pour la charge de travail GoCJ*

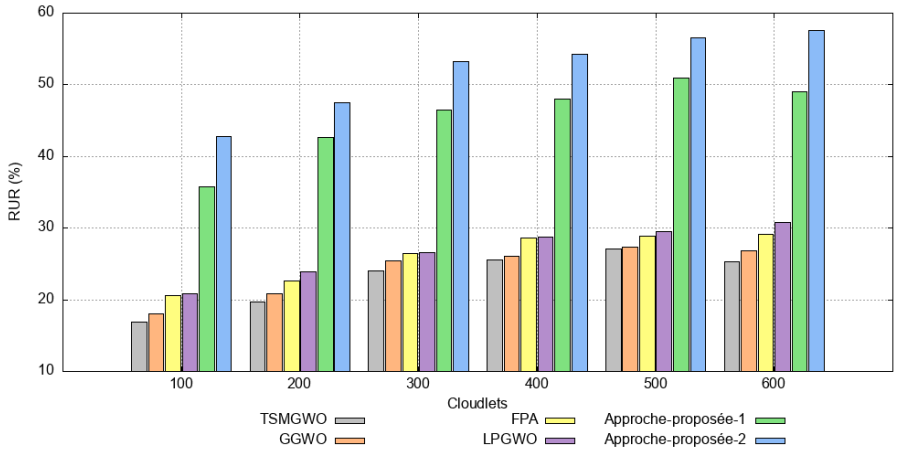


FIGURE 5.11 – *Résultats comparatifs en termes de taux d'utilisation des ressources pour la charge de travail HPC2N*

Les résultats présentés dans la Table 5.7 confirment cette tendance. Ils révèlent que nos techniques hybrides peuvent engendrer une aug-

mentation maximale de RUR allant jusqu'à 60.98% pour toutes les instances considérées dans cette étude. De plus, nos propositions surpassent le TSMGWO, le GGWO, le LPGWO et le FPA en termes de RUR, avec une valeur supérieure à 64%. Cette supériorité s'explique par le fait que nos approches nouvellement conçues mobilisent davantage de ressources pour réduire le makespan, ce qui se traduit par une utilisation plus efficace des ressources Cloud lors du processus d'ordonnancement.

Résultats obtenus par rapport au degré de déséquilibre

Ensemble de données	Cloudlets	Méthode utilisée					
		TSMGWO	GGWO	FPA	LPGWO	Approche proposée 1	Approche proposée 2
Synthetic	100	4.582	5.138	5.602	5.02	2.122	1.876
	200	5.93	5.504	6.664	5.346	3.058	2.314
	300	6.514	6.304	7.806	6.17	3.782	3.05
	400	6.522	6.278	8.528	6.83	4.152	3.242
	500	6.526	6.866	8.0766	6.513	4.293	3.636
	600	6.996	6.94	8.98	7.4	4.52	3.793
GoCJ	100	2.454	2.848	3.236	2.616	1.55	1.2
	200	2.836	2.92	3.9	3.103	1.92	1.476
	300	3.17	3.093	3.81	3.326	2.016	1.616
	400	3.01	2.81	3.443	2.986	2.07	1.693
	500	2.926	3.2	3.84	3.15	2.14	1.866
	600	3.176	3.196	3.793	3.473	2.24	1.886
HPC2N	100	3.922	4.1	4.504	3.808	2.198	1.734
	200	5.228	5.562	6.982	5.088	3.262	2.708
	300	5.788	5.62	7.498	6.19	3.94	2.998
	400	5.53	5.63	7.38	6.06	3.982	3.354
	500	5.904	6.346	8.154	6.474	4.2	3.646
	600	6.962	6.476	8.57	7.354	4.836	4.096

TABLE 5.8 – *Résultats du degrée de déséquilibre des approches proposées et des algorithmes comparés*

En comparant les résultats obtenus par les algorithmes TSMGWO, LPGWO, GGWO et FPA avec ceux de nos propositions en termes de degré de déséquilibre, nous pouvons observer, respectivement dans la Table 5.8 et sur les Figures 5.12, 5.13 et 5.14, que nos approches offrent des performances élevées avec une amélioration significative par rapport aux autres métaheuristiques comparées dans tous les scénarios testés. De manière notable, la deuxième proposition se révèle supérieure à la première proposition que nous avons formulée. À titre d'exemple, dans le cas du jeu de données HPC2N, l'approche basée sur la deuxième hybridation proposée atteint des solutions qui sont de 54.46 %, 55.78 %, 57.70 % et 61.50 % plus efficaces en termes de degré de déséquilibre que celles fournies par les algorithmes LPGWO, TSMGWO, GGWO et FPA respectivement. Cette amélioration découle de la stratégie des techniques hybrides, qui explore efficacement le nombre de VM disponibles et s'efforce de garantir une

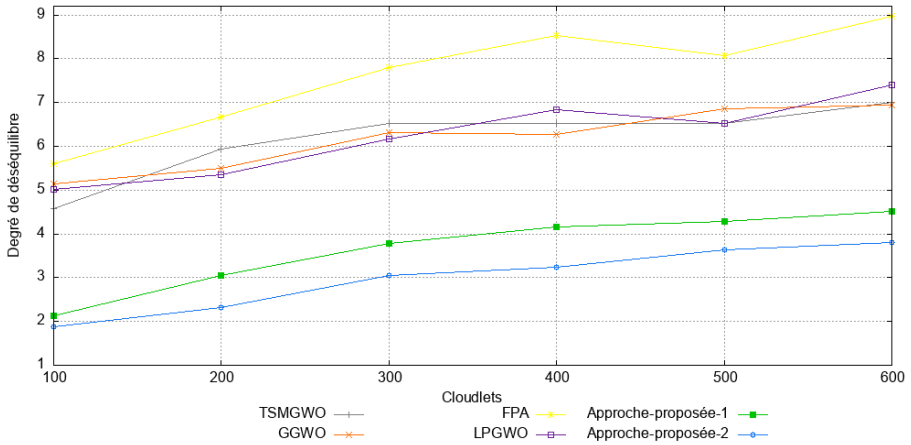


FIGURE 5.12 – *Résultats comparatifs en termes de degré de déséquilibre pour une charge de travail synthétique*

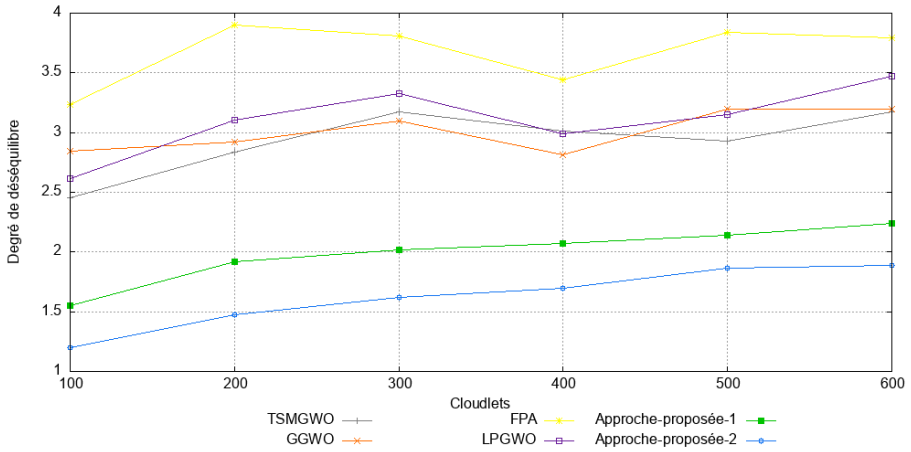


FIGURE 5.13 – *Résultats comparatifs en termes de degré de déséquilibre pour la charge de travail GoCJ*

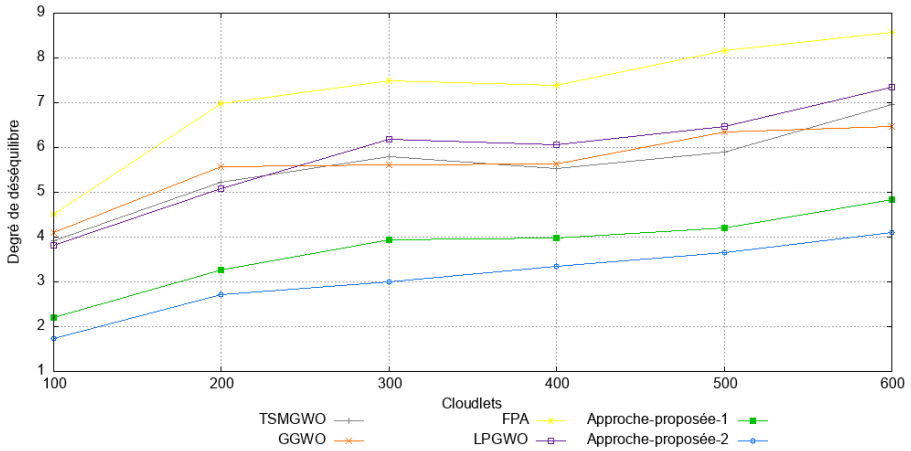


FIGURE 5.14 – *Résultats comparatifs en termes de degré de déséquilibre pour la charge de travail HPC2N*

valeur minimale du degré de déséquilibre en fonction des capacités des ressources. Ainsi, cela permet une distribution équilibrée et efficace de la

charge de travail soumise entre les machines virtuelles Cloud disponibles, tout en évitant qu'une VM ne soit surchargée à aucun moment au cours du processus d'ordonnancement.

Résultats obtenus par rapport au débit

Ensemble de données	Cloudlets	Méthode utilisée					
		TSMGWO	GGWO	FPA	LPGWO	Approche proposée 1	Approche proposée 2
Synthetic	100	27.834	26.048	32.214	32.978	33.262	34.32
	200	33.79	31.672	41.406	42.494	50.068	51.154
	300	34.212	34.008	42.682	45.656	53.716	58.064
	400	38.992	37.678	45.936	54.06	60.516	66.038
	500	41.91	40.366	48.953	61.53	63.19	74.31
	600	44.713	44.47	45.926	59.433	65.073	70.213
GoCJ	100	1.388	1.44	1.772	1.86	2.028	2.292
	200	1.62	1.433	1.753	2.44	2.653	3.13
	300	1.543	1.43	2.066	2.203	2.673	3.35
	400	1.963	1.733	2.666	2.816	3.256	3.546
	500	1.956	1.906	2.336	2.796	3.23	3.59
	600	1.88	1.77	2.24	2.34	3.523	3.69
HPC2N	100	0.53	0.554	0.668	0.726	0.71	0.816
	200	0.816	0.716	0.842	1.054	1.182	1.186
	300	0.844	0.796	0.918	1.076	1.322	1.49
	400	0.864	0.796	0.91	1.108	1.306	1.418
	500	0.838	0.742	0.816	0.982	1.304	1.37
	600	0.936	0.872	0.954	1.15	1.426	1.606

TABLE 5.9 – *Résultats du débit des approches proposées et des algorithmes comparés*

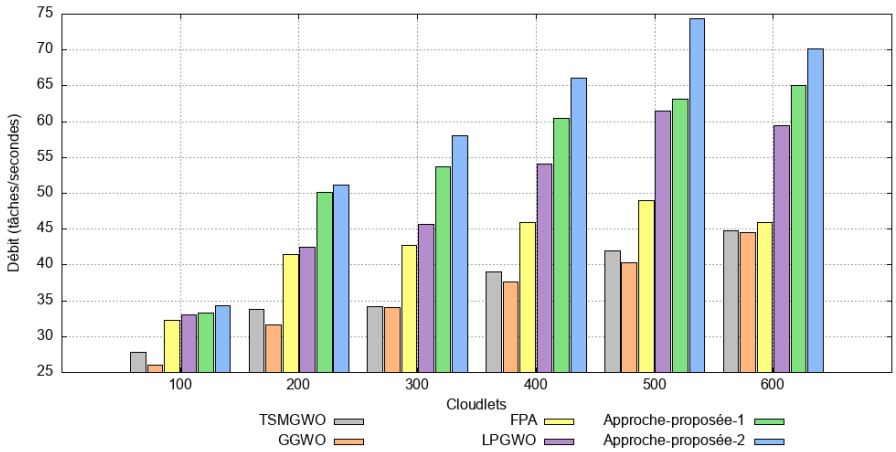


FIGURE 5.15 – *Résultats comparatifs en termes de débit pour une charge de travail synthétique*

Selon les résultats présentés dans la Table 5.9 et illustrés sur les Figures 5.15, 5.16 et 5.17, il est clairement évident que nos approches proposées

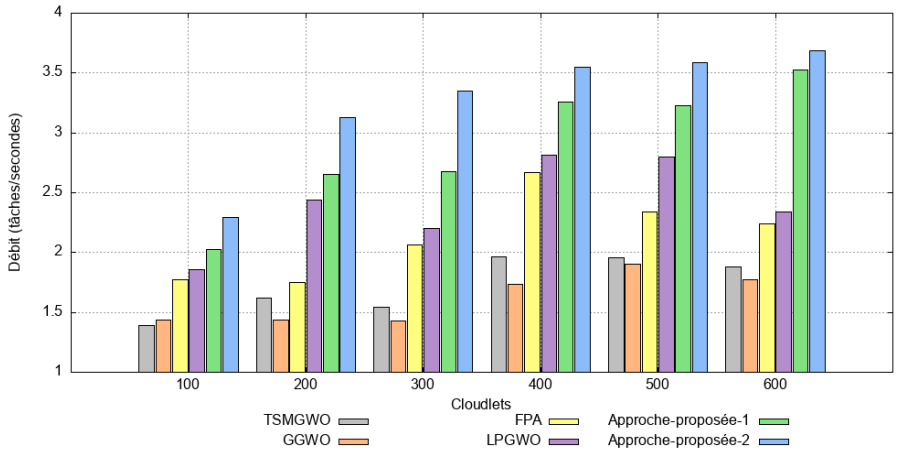


FIGURE 5.16 – *Résultats comparatifs en termes de débit pour la charge de travail GoCJ*

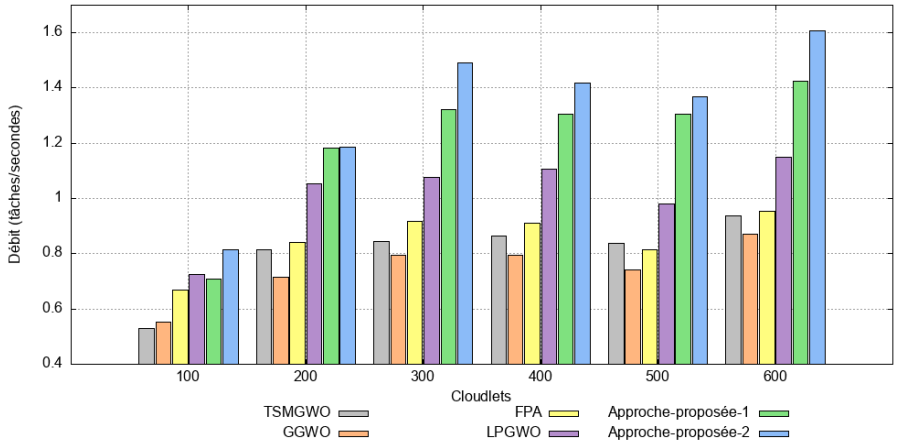


FIGURE 5.17 – *Résultats comparatifs en termes de débit pour la charge de travail HPC2N*

continuent d'exceller en maximisant considérablement le débit par rapport aux optimiseurs LPGWO, TSMGWO, GGWO et FPA. Les données du Tableau 5.9 confirment sans équivoque l'écart significatif entre les résultats de nos propositions et ceux des autres optimiseurs pour toutes les situations de charge de travail examinées. À titre d'exemple, dans le cas des traces GoCJ, la réduction du débit grâce à notre deuxieme stratégie est de 36.58 %, 43.99 %, 53.94 % et 57.31 % par rapport à LPGWO, FPA, TSMGWO et GGWO, respectivement. Cette amélioration s'explique par la politique d'ordonnancement que nous avons mise en place, favorisant l'obtention d'un makespan minimal, ce qui se traduit par une augmentation du nombre de tâches achevées dans un laps de temps donné.

Comparaison par rapport à la convergence de la fonction objectif

Pour mettre en évidence l'efficacité de nos algorithmes hybrides proposés, nous avons également comparé le comportement de convergence de chaque algorithme au fil des itérations. Les courbes de convergence des algorithmes TSMGWO, LPGWO, GGWO, FPA et de nos approches sont illustrées dans la Figure 5.18. Dans cette simulation, nous avons planifié

l'ordonnancement de 600 tâches issues d'une charge de travail HPC2N. La figure révèle clairement que les six algorithmes testés optimisent la fonction objectif. Cependant, les résultats obtenus montrent la supériorité de nos approches nouvellement conçues en termes de comportement de convergence global par rapport aux approches concurrentes, qui ont tendance à être piégées dans des optima locaux.

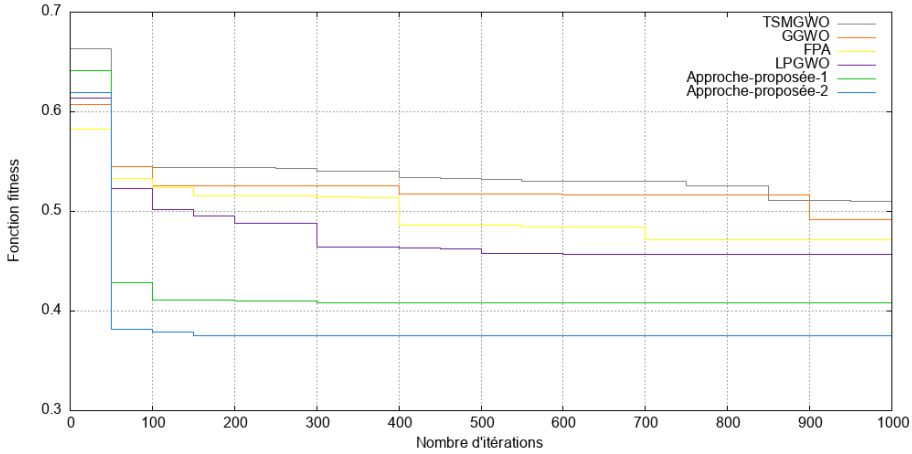


FIGURE 5.18 – *Convergence de la fonction objectif avec 600 tâches de la charge de travail HPC2N*

En outre, en mettant l'accent sur l'aspect temps réel, qui revêt une importance cruciale car il démontre la capacité du système Cloud à s'adapter rapidement à un environnement en constante évolution, nous constatons que la deuxième approche établie est capable d'atteindre de meilleures solutions globales en moins de temps de convergence que ses concurrents directs. Ainsi, l'algorithme hybride proposé réussit avec succès à trouver un bon équilibre entre les phases d'exploration et d'exploitation tout au long du processus de recherche.

Dans l'ensemble, les résultats de scalabilité obtenus sur différents ensembles de données de charge de travail réelle soulignent que les approches hybrides introduites maintiennent un bon équilibre entre leurs inclinations pour l'exploration et l'exploitation, ce qui se traduit par des performances constamment supérieures. De plus, les résultats indiquent également que les algorithmes proposés parviennent à atteindre leurs objectifs d'amélioration des performances globales du processus d'ordonnancement, en ce qui concerne diverses mesures telles que le makespan, le coût, l'utilisation des ressources, le degré de déséquilibre et le débit. Les caractéristiques suivantes expliquent l'utilisation potentielle des stratégies hybrides proposées et leur supériorité par rapport à leurs concurrents directs LPGWO, FPA, GGWO et TSMGWO :

— La combinaison du comportement de pollinisation des fleurs et de l'optimiseur du loup gris avec l'utilisation d'opérateurs de croisement d'algorithmes évolutionnaires, ainsi que la stratégie du "memeplex" inspirée de l'algorithme de saut de grenouille, permet d'atteindre un équilibre harmonieux entre les phases d'exploration et d'exploitation.

— L'amélioration progressive des pires individus de la population au fil des itérations s'avère bénéfique pour éviter de rester piégé dans des optima locaux au cours du processus d'optimisation.

— La stratégie de recherche basée sur les trois meilleures solutions de la population pour mettre à jour les pires individus a un impact constructif sur le potentiel d'exploration de l'algorithme.

— La puissance des opérateurs de croisement de l'algorithme génétique est capable d'aider les solutions candidates à exploiter des régions prometteuses dans l'espace de recherche.

5.5 Conclusion

Dans ce dernier chapitre, nous avons présenté deux contributions majeures visant à relever le défi complexe de l'ordonnancement multi-objectif des tâches dans les environnements Cloud hétérogènes. L'objectif est d'orchestrer de manière plus efficace l'attribution des tâches aux machines virtuelles, tout en répondant aux exigences à la fois des utilisateurs et des prestataires de services. Cette optimisation nécessite un équilibre subtil entre l'exploration et l'exploitation de l'espace de recherche des solutions, afin de minimiser les risques de convergence prématurée et d'enlisement dans des solutions locales.

Pour atteindre ces objectifs, nous avons développé deux approches novatrices. La première combine la bio-inspiration tirée du comportement de pollinisation des fleurs avec l'efficacité de l'optimisation par intelligence des loups gris. De plus, nous avons intégré les opérateurs de croisement issus des algorithmes évolutifs pour maintenir la diversité de la population. Cette combinaison unique nous permet de naviguer efficacement dans un espace de recherche complexe et d'atteindre des solutions de haute qualité. Les études expérimentales détaillées et les différents scénarios envisagés, réalisés à l'aide du CloudSim, ont démontré la supériorité de notre algorithme hybride par rapport à d'autres techniques d'optimisation bien établies de la littérature, notamment LPGWO, GGWO, TSMGWO et l'algorithme FPA.

La deuxième contribution constitue une version parallèle de l'algorithme précédemment proposé, ce qui signifie que le processus d'optimisation peut désormais être exécuté sur des machines parallèles. Cette extension implique des modifications substantielles à l'algorithme initial pour tirer parti de la parallélisation. Nous avons introduit la stratégie du "memeplex", inspirée de l'algorithme de saut de grenouilles, au sein de notre algorithme d'optimisation hybride. Chaque memeplex dispose de son propre espace de recherche dédié. L'objectif principal de ces ajustements est d'exploiter de manière encore plus efficace l'approche précédemment présentée pour résoudre des problèmes d'ordonnancement multi-objectif. Les performances de cette seconde approche ont fait l'objet d'une évaluation approfondie et ont clairement démontré son potentiel par rapport à la version initiale de la proposition.

Conclusion générale

Le Cloud Computing a provoqué une révolution inédite dans le domaine de l'informatique, devenant une partie intégrante de la plupart des applications Internet. Cependant, répondre à la demande croissante de ces services nécessite une gestion efficace de ressources massives, composées de milliers, voire de centaines de milliers de serveurs. Cette gestion doit s'adapter à un environnement en constante évolution tout en garantissant la satisfaction des besoins des opérateurs et des utilisateurs, ce qui engendre des défis scientifiques complexes.

Cette thèse se concentre précisément sur la problématique de l'ordonnancement des charges de travail des utilisateurs, c'est-à-dire l'attribution de ces charges aux ressources fournies par les prestataires de services Cloud. L'objectif est d'optimiser la satisfaction des exigences de qualité de service définies dans les contrats de service SLA. De plus, il convient de noter que cette problématique s'inscrit dans le cadre d'un problème d'optimisation multi-objectif, du fait de la nature conflictuelle des métriques de QoS à prendre en considération.

Au cours de cette recherche, nous avons examiné en détail les travaux existants grâce à une étude bibliographique approfondie, soulignant les avantages et les limitations des approches précédentes. L'analyse méticuleuse de ces travaux nous a conduit à formuler trois contributions majeures, basées sur l'utilisation de techniques intelligentes et adaptatives. Ces contributions ouvrent la voie à des perspectives prometteuses pour aborder la complexité inhérente de cette problématique. Elles permettent d'explorer des solutions qui se rapprochent de l'optimalité tout en prenant en compte les contraintes de calcul.

Dans un premier temps, nous avons adapté la métaheuristique de pollinisation des fleurs, qui était sous-exploitée par la communauté de chercheurs. Dans cette optique, un algorithme adapté a été développé, intégrant à la fois le principe d'optimalité de Pareto et la technique TOPSIS pour une sélection judicieuse des ressources, en fonction des préférences des utilisateurs. Nous avons envisagé deux scénarios d'optimisation : l'optimisation mono-objectif et l'optimisation multi-objectif, incluant trois critères : le makespan, le coût et la fiabilité de l'exécution des tâches.

Afin de surmonter les limitations des techniques existantes, notamment la convergence prématurée et le risque de l'optimalité locale, nous avons conçu un nouvel algorithme d'optimisation hybride. Cet algorithme combine le comportement de pollinisation des fleurs avec la capacité d'exploration de la stratégie d'optimisation du loup gris, tout en intégrant des opérateurs de croisement issus des algorithmes évolutionnaires pour maintenir la diversité tout au long du processus d'optimisation.

Cette approche hybride s'attache principalement à orienter la recherche vers les positions optimales des membres de la population, encourageant ainsi l'exploration de l'espace de recherche aux côtés de leurs leaders. Pour aller encore plus loin dans la recherche de solutions efficaces et surmonter cette faiblesse, une troisième contribution a été apportée. Elle implique l'intégration d'une stratégie inspirée de l'algorithme de saut de grenouille, qui divise la population en plusieurs sous-populations de petite taille, appelées "memeplexes". Chaque memeplex possède son propre espace de recherche individuel et ses leaders les plus performants.

Les différentes contributions présentées dans cette thèse démontrent une recherche active axée sur le développement de nouvelles approches algorithmiques pour l'optimisation multi-objectif dans les environnements Cloud hétérogènes. Nous accordons une grande importance à l'équilibre entre les objectifs des fournisseurs de services Cloud et les exigences de qualité de service des utilisateurs. Les résultats obtenus suite à des comparaisons approfondies avec d'autres travaux existants, et à travers des simulations intensives impliquant des charges de travail synthétiques et réelles issues de divers domaines scientifiques, mettent incontestablement en avant la supériorité de nos contributions par rapport à des travaux antérieurs.

De plus, nos algorithmes ont été soumis à des scénarios de plus en plus complexes, impliquant des charges de travail diverses. Ils ont démontré une adaptabilité supérieure par rapport à leurs concurrents, en fournissant des solutions de meilleure qualité en moins de temps de convergence. Dans l'ensemble de nos tests, notre algorithme basé sur la stratégie des "memeplexes" s'est révélé être le plus performant, suivi de près par notre deuxième contribution, qui surpasse les autres algorithmes comparés.

Les travaux exposés dans cette thèse, peuvent avoir un impact sur la suite des travaux de recherches à entreprendre dans l'avenir. Les approches et les stratégies que nous avons développées ont produit des résultats prometteurs, qui justifient une poursuite de la recherche, notamment en explorant leurs applications à l'ordonnancement du flux de travail dans des environnements Cloud réels. Nous anticipons des défis complexes liés à la minimisation des coûts de communication et à l'exécution parallèle des tâches, en tenant compte des diverses exigences des utilisateurs finaux et des prestataires de services. Il est également pertinent de poursuivre la recherche sur d'autres objectifs d'optimisation dans les systèmes Cloud, tels que la tolérance aux pannes, la consommation d'énergie et l'équilibrage des charges, pour faire face aux défaillances imprévues et garantir la stabilité du système. Une autre piste de recherche future potentielle consisterait à appliquer les algorithmes hybrides proposés à des contextes d'optimisation différents de ceux abordés dans ce manuscrit, tels que le fog computing, les réseaux de capteurs, le traitement d'images, la radio cognitive, et bien d'autres domaines encore.

Publications (8)

Articles de revues internationales avec comité de lecture et de sélection (3)

1. Arslan Nedhir Malti, Mourad Hakem and Badr Benmammar. "A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems". Cluster Computing, The Journal of Networks Software Tools and Applications, Electronic ISSN : 1573-7543. DOI : 10.1007/s10586-023-04099-3. First published : 12 July 2023.
   Répertoriée DBLP, ACM Digital Library, Scopus | Elsevier et Clarivate Analytics : Impact Factor = 4.4.
2. Arslan Nedhir Malti, Mourad Hakem and Badr Benmammar. "Multi-objective Task Scheduling in Cloud Computing". Concurrency and Computation : Practice and Experience, Wiley InterScience Edition, Online ISSN : 1532-0634. DOI : 10.1002/cpe.7252. Volume 34, Issue 25. First published : 30 July 2022.
   Répertoriée DBLP, ACM Digital Library, Scopus | Elsevier et Clarivate Analytics : Impact Factor = 2.
3. Arslan Nedhir Malti, Mourad Hakem and Badr Benmammar. "An improved hybrid multi-objective optimization algorithm for task scheduling in cloud environments". (submitted).

Articles d'actes de conférences internationales avec comité de lecture et de sélection (4)

1. Arslan Nedhir Malti, Badr Benmammar and Mourad Hakem. "Task Scheduling Optimization in Cloud Computing : A Comparative Study Between Flower Pollination and Butterfly Optimization Algorithms". The 5th international conference on pattern analysis and intelligent systems (PAIS'23). 25-26 October 2023. Faculty of Sciences, University Ferhat Abbas Setif 1, Algeria.
2. Arslan Nedhir Malti, Badr Benmammar and Mourad Hakem. "A Comparative Study of Metaheuristics Based Task Scheduling in Cloud Computing". The 7th International Symposium on Modelling and Implementation of Complex Systems (MISC 2022). 30-31 October 2022. University of Mostaganem – Abdelhamid Ibn Badis, Algeria. In : Modelling and Implementation of Complex Systems. pp 263–278. Lecture Notes in Networks and Systems, vol 593. Springer.Cham. DOI : 10.1007/978-3-031-18516-8_19.

3. Arslan Nedhir Malti, Badr Benmammar and Mourad Hakem. "Task Scheduling in Cloud Computing Based on FPA Metaheuristic Algorithm". Proceedings of the International Conference on Communication and Signal Processing & Information Technology (CSP). The 19th IEEE International Multi-Conference on Systems, Signals & Devices (SSD 2022). May 06-10, 2022. Faculty of Technology, University Ferhat Abbas Setif 1, Algeria.
4. Arslan Nedhir Malti, Badr Benmammar and Mourad Hakem. "QoS based task scheduling algorithm in Cloud computing". The 7th international conference on wireless technologies, embedded and intelligent systems (WITS'22). Faculty of Sciences and Technologies of Tangier, Morocco. 2-4 March 2022.
   Paper published in the ICIES 2022 proceedings (Conference orgonized by EurSED, WITS'22 scientific production partner).

Articles d'actes de conférences nationales avec comité de lecture et de sélection (1)

1. Arslan Nedhir Malti, Badr Benmammar and Mourad Hakem. "QoS-aware Task Scheduling in Cloud Computing". 1st National Conference on Applied Computing and Smart Technologies (ACST'21). July 10, 2021, Ecole Supérieure en Informatique, Sidi Bel Abbès, Algeria.

Bibliographie

[1] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, D. Leaf, *et al.*, "Nist cloud computing reference architecture," *NIST special publication*, vol. 500, no. 2011, pp. 1–28, 2011.

[2] D. Molina, J. Poyatos, J. D. Ser, S. García, A. Hussain, and F. Herrera, "Comprehensive taxonomies of nature-and bio-inspired optimization : Inspiration versus algorithmic behavior, critical analysis recommendations," *Cognitive Computation*, vol. 12, pp. 897–939, 2020.

[3] "Cloud computing market." Accédé le 03/01/2024, https://www.marketsandmarkets.com/Market-Reports/cloud-computing-market-234.html.

[4] Q. Zhang, S. Geng, and X. Cai, "Survey on task scheduling optimization strategy under multi-cloud environment," *CMES-Computer Modeling in Engineering & Sciences*, vol. 135, no. 3, pp. 1863–1900, 2023.

[5] P. Pirozmand, A. Javadpour, H. Nazarian, P. Pinto, S. Mirkamali, and F. Ja'fari, "Gsaga : A hybrid algorithm for task scheduling in cloud infrastructure," *The Journal of Supercomputing*, vol. 78, no. 15, pp. 17423–17449, 2022.

[6] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization : Algorithm and applications," *Future generation computer systems*, vol. 97, pp. 849–872, 2019.

[7] A. Pradhan, S. K. Bisoy, and A. Das, "A survey on pso based meta-heuristic scheduling mechanism in cloud computing environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 4888–4901, 2022.

[8] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud computing : Principles and paradigms*. John Wiley & Sons, 2010.

[9] C. Chaturvedi and B. B. Gupta, "Cloud computing security : Taxonomy of issues, challenges, case studies, and solutions," in *Handbook of Research on Intrusion Detection Systems*, pp. 306–325, IGI Global, 2020.

[10] J. Surbiryala and C. Rong, "Cloud computing : History and overview," in *2019 IEEE Cloud Summit*, pp. 1–7, IEEE, 2019.

[11] A. Archana Lisbon and R. Kavitha, "A study on cloud and fog computing security issues and solutions," *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, no. 03, pp. 17–22, 2017.

[12] H. Elazhary, "Internet of things (iot), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms : Disambiguation and research directions," *Journal of network and computer applications*, vol. 128, pp. 105–140, 2019.

[13] "What is cloud computing ?." Accédé le 03/01/2024, https://www.ibm.com/topics/cloud-computing.

[14] G. Oliver and S. Knight, "Storage is a strategic issue : digital preservation in the cloud," *D-Lib Magazine*, vol. 21, no. 3/4, 2015.

[15] Y. Al-Issa, M. A. Ottom, A. Tamrawi, *et al.*, "ehealth cloud security challenges : a survey," *Journal of healthcare engineering*, vol. 2019, pp. 1–15, 2019.

[16] P. Mell, T. Grance, *et al.*, "The nist definition of cloud computing," 2011.

[17] T. Labidi, A. Mtibaa, W. Gaaloul, and F. Gargouri, "Cloud sla negotiation and re-negotiation : An ontology-based context-aware approach," *Concurrency and Computation : Practice and Experience*, vol. 32, no. 15, p. e5315, 2020.

[18] L. Chouhan, P. Bansal, B. Lauhny, and Y. Chaudhary, "A survey on cloud federation architecture and challenges," in *Social Networking and Computational Intelligence : Proceedings of SCI-2018*, pp. 51–65, Springer, 2020.

[19] Z. Zhou, J. Yu, F. Li, and F. Yang, "Virtual machine migration algorithm for energy efficiency optimization in cloud computing," *Concurrency and Computation : Practice and Experience*, vol. 30, no. 24, p. e4942, 2018.

[20] Y. N. Khalid, M. Aleem, U. Ahmed, M. A. Islam, and M. A. Iqbal, "Troodon : A machine-learning based load-balancing application scheduler for cpu–gpu system," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 79–94, 2019.

[21] U. Arshad, M. Aleem, G. Srivastava, and J. C.-W. Lin, "Utilizing power consumption and sla violations using dynamic vm consolidation in cloud data centers," *Renewable and Sustainable Energy Reviews*, vol. 167, p. 112782, 2022.

[22] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers : a survey on software technologies," *Cluster Computing*, vol. 26, no. 3, pp. 1845–1875, 2023.

[23] N. Haile and J. Altmann, "Evaluating investments in portability and interoperability between software service platforms," *Future Generation Computer Systems*, vol. 78, pp. 224–241, 2018.

[24] N. E. H. Bouzerzour, S. Ghazouani, and Y. Slimani, "A survey on the service interoperability in cloud computing : Client-centric and provider-centric perspectives," *Software : Practice and Experience*, vol. 50, no. 7, pp. 1025–1060, 2020.

[25] "Open virtualization format." Accédé le 03/01/2024, https://www.dmtf.org/standards/ovf.

[26] "One interface to rule them all."Accédé le 03/01/2024, http://libcloud.apache.org/.

[27] S. Bharany, K. Kaur, S. Badotra, S. Rani, Kavita, M. Wozniak, J. Shafi, and M. F. Ijaz, "Efficient middleware for the portability of paas services consuming applications among heterogeneous clouds," *Sensors*, vol. 22, no. 13, p. 5013, 2022.

[28] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–16, 2018.

[29] R. Mohamadi Bahram Abadi, A. M. Rahmani, and S. H. Alizadeh, "Server consolidation techniques in virtualized data centers of cloud environments : a systematic literature review," *Software : Practice and Experience*, vol. 48, no. 9, pp. 1688–1726, 2018.

[30] D. Gao, G.-G. Wang, and W. Pedrycz, "Solving fuzzy job-shop scheduling problem using de algorithm improved by a selection mechanism," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 12, pp. 3265–3275, 2020.

[31] X. Han, Y. Han, Q. Chen, J. Li, H. Sang, Y. Liu, Q. Pan, and Y. Nojima, "Distributed flow shop scheduling with sequence-dependent setup times using an improved iterated greedy algorithm," *Complex System Modeling and Simulation*, vol. 1, no. 3, pp. 198–217, 2021.

[32] X. Ma, J. Zhao, and Y. Gong, "Joint scheduling and resource allocation for efficiency-oriented distributed learning over vehicle platooning networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10894–10908, 2021.

[33] S. R. Konda, L. K. Panwar, B. K. Panigrahi, R. Kumar, and V. Gupta, "Binary fireworks algorithm application for optimal schedule of electric vehicle reserve in traditional and restructured electricity markets," *International Journal of Bio-Inspired Computation*, vol. 18, no. 1, pp. 38–48, 2021.

[34] X. Zhao, J. Su, J. Cai, H. Yang, and T. Xi, "Vehicle anomalous trajectory detection algorithm based on road network partition," *Applied Intelligence*, vol. 52, no. 8, pp. 8820–8838, 2022.

[35] J. Wang, Z. Na, and X. Liu, "Collaborative design of multi-uav trajectory and resource scheduling for 6g-enabled internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15096–15106, 2020.

[36] Q. Peng, H. Wu, and R. Xue, "Review of dynamic task allocation methods for uav swarms oriented to ground targets," *Complex System Modeling and Simulation*, vol. 1, no. 3, pp. 163–175, 2021.

[37] B. Tong, L. Chen, and H. Duan, "A path planning method for uavs based on multi-objective pigeon-inspired optimisation and differential evolution," *International Journal of Bio-Inspired Computation*, vol. 17, no. 2, pp. 105–112, 2021.

[38] K. Rajakumari, M. V. Kumar, G. Verma, S. Balu, D. K. Sharma, and S. Sengan, "Fuzzy based ant colony optimization scheduling in cloud computing.," *Computer Systems Science & Engineering*, vol. 40, no. 2, 2022.

[39] J. Xu, Z. Zhang, Z. Hu, L. Du, and X. Cai, "A many-objective optimized task allocation scheduling model in cloud computing," *Applied Intelligence*, vol. 51, pp. 3293–3310, 2021.

[40] M. Xu, M. Zhang, X. Cai, and G. Zhang, "Adaptive neighbourhood size adjustment in moea/d-dra," *International Journal of Bio-Inspired Computation*, vol. 17, no. 1, pp. 14–23, 2021.

[41] M. Li and G.-G. Wang, "A review of green shop scheduling problem," *Information Sciences*, vol. 589, pp. 478–496, 2022.

[42] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A survey of pso-based scheduling algorithms in cloud computing," *Journal of Network and Systems Management*, vol. 25, no. 1, pp. 122–158, 2017.

[43] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics : review, taxonomy, open challenges, and future trends," *Swarm and Evolutionary Computation*, vol. 62, p. 100841, 2021.

[44] F. S. Prity, M. H. Gazi, and K. Uddin, "A review of task scheduling in cloud computing based on nature-inspired optimization algorithm," *Cluster Computing*, pp. 1–31, 2023.

[45] R. Kaur, V. Laxmi, and Balkrishan, "Performance evaluation of task scheduling algorithms in virtual cloud environment to minimize makespan," *International Journal of Information Technology*, pp. 1–15, 2022.

[46] R. Ghafari, F. H. Kabutarkhani, and N. Mansouri, "Task scheduling algorithms for energy optimization in cloud environment : a comprehensive review," *Cluster Computing*, vol. 25, no. 2, pp. 1035–1093, 2022.

[47] R. Garg, M. Mittal, and L. H. Son, "Reliability and energy efficient workflow scheduling in cloud environment," *Cluster Computing*, vol. 22, no. 4, pp. 1283–1297, 2019.

[48] S. Qin, D. Pi, and Z. Shao, "Ails : A budget-constrained adaptive iterated local search for workflow scheduling in cloud environment," *Expert Systems with Applications*, vol. 198, p. 116824, 2022.

[49] M. A. Alworafi and S. Mallappa, "A collaboration of deadline and budget constraints for task scheduling in cloud computing," *Cluster Computing*, vol. 23, no. 2, pp. 1073–1083, 2020.

[50] M. M. Golchi, S. Saraeian, and M. Heydari, "A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments : Performance evaluation," *Computer Networks*, vol. 162, p. 106860, 2019.

[51] M. K. Elnahary, A. Y. Hamed, and H. El-Sayed, "Task scheduling optimization in cloud computing by cuckoo search algorithm," *Sohag Journal of Sciences*, vol. 7, no. 3, pp. 29–37, 2022.

[52] S. H. H. Madni, M. S. A. Latiff, J. Ali, and S. M. Abdulhamid, "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds," *Arabian Journal for Science and Engineering*, vol. 44, pp. 3585–3602, 2019.

[53] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, "A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing," *IEEE Access*, vol. 7, pp. 160916–160926, 2019.

[54] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, 2019.

[55] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software : Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[56] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst : A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE international conference on advanced information networking and applications*, pp. 446–452, IEEE, 2010.

[57] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "Emusim : an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software : Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013.

[58] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud : a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, pp. 1263–1283, 2012.

[59] S. K. Garg and R. Buyya, "Networkcloudsim : Modelling parallel applications in cloud simulations," in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, pp. 105–113, IEEE, 2011.

[60] E. L. Lawler and D. E. Wood, "Branch-and-bound methods : A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[61] S. Dreyfus, "Richard bellman on the birth of dynamic programming," *Operations Research*, vol. 50, no. 1, pp. 48–51, 2002.

[62] B. S. Stewart and C. C. White III, "Multiobjective a*," *Journal of the ACM (JACM)*, vol. 38, no. 4, pp. 775–814, 1991.

[63] S. W. Golomb and L. D. Baumert, "Backtrack programming," *Journal of the ACM (JACM)*, vol. 12, no. 4, pp. 516–524, 1965.

[64] G. Dantzig, *Linear programming and extensions*. Princeton university press, 1963.

[65] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, vol. 13, no. 5, pp. 533–549, 1986.

[66] K.-F. Man, K.-S. Tang, and S. Kwong, "Genetic algorithms : concepts and applications [in engineering design]," *IEEE transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519–534, 1996.

[67] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, pp. 3–52, 2002.

[68] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341–359, 1997.

[69] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and biological systems : towards a new bionics ?*, pp. 703–712, Springer, 1993.

[70] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, pp. 39–43, Ieee, 1995.

[71] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system : optimization by a colony of cooperating agents," *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[72] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization : artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, pp. 459–471, 2007.

[73] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*, pp. 169–178, Springer, 2009.

[74] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm : theory and application," *Advances in engineering software*, vol. 105, pp. 30–47, 2017.

[75] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Gsa : a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[76] A. Hatamlou, "Black hole : A new heuristic optimization approach for data clustering," *Information sciences*, vol. 222, pp. 175–184, 2013.

[77] H. Shah-Hosseini, "Principal components analysis by the galaxy-based search algorithm : a novel metaheuristic for continuous optimisation," *International journal of computational science and engineering*, vol. 6, no. 1-2, pp. 132–140, 2011.

[78] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization : harmony search theory and practice," *Computer methods in applied mechanics and engineering*, vol. 194, no. 36-38, pp. 3902–3933, 2005.

[79] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm : an algorithm for optimization inspired by imperialistic competition," in *2007 IEEE congress on evolutionary computation*, pp. 4661–4667, Ieee, 2007.

[80] N. Moosavian and B. K. Roodsari, "Soccer league competition algorithm : A novel meta-heuristic algorithm for optimal design of water distribution networks," *Swarm and Evolutionary Computation*, vol. 17, pp. 14–24, 2014.

[81] Y. Shi, "Brain storm optimization algorithm," in *Advances in Swarm Intelligence : Second International Conference, ICSI 2011, Chongqing, China, June 12-15, 2011, Proceedings, Part I 2*, pp. 303–309, Springer, 2011.

[82] D. Oliva, N. Ortega-Sanchez, M. A. Navarro, A. Ramos-Michel, M. El-Abd, S. J. Mousavirad, and M. H. Nadimi-Shahraki, "Segmentation of thermographies from electronic systems by using the global-best brain storm optimization algorithm," *Multimedia Tools and Applications*, pp. 1–31, 2023.

[83] M. Ghaemi and M.-R. Feizi-Derakhshi, "Forest optimization algorithm," *Expert Systems with Applications*, vol. 41, no. 15, pp. 6676–6687, 2014.

[84] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment," *PloS one*, vol. 12, no. 5, p. e0176321, 2017.

[85] A. Brandwajn and T. Begin, "First-come-first-served queues with multiple servers and customer classes," *Performance Evaluation*, vol. 130, pp. 51–63, 2019.

[86] M. Waheed, N. Javaid, A. Fatima, T. Nazar, K. Tehreem, and K. Ansar, "Shortest job first load balancing algorithm for efficient resource management in cloud," in *Advances on Broadband and Wireless Computing, Communication and Applications : Proceedings of the 13th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA-2018)*, pp. 49–62, Springer, 2019.

[87] T. Balharith and F. Alhaidari, "Round robin scheduling algorithm in cpu and cloud computing : a review," in *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, pp. 1–7, IEEE, 2019.

[88] Y. Samadi, M. Zbakh, and C. Tadonki, "E-heft : enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing," in *2018 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 601–609, IEEE, 2018.

[89] B. Li, L. Niu, X. Huang, H. Wu, and Y. Pei, "Minimum completion time offloading algorithm for mobile edge computing," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pp. 1929–1933, IEEE, 2018.

[90] H. Krishnaveni and V. Sinthu Janita Prakash, "Execution time based sufferage algorithm for static task scheduling in cloud," in *Advances in Big Data and Cloud Computing : Proceedings of ICBDCC18*, pp. 61–70, Springer, 2019.

[91] D. George Amalarethinam and S. Kavitha, "Rescheduling enhanced min-min (remm) algorithm for meta-task scheduling in cloud computing," in *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*, pp. 895–902, Springer, 2019.

[92] A. Sandana Karuppan, S. Meena Kumari, and S. Sruthi, "A priority-based max-min scheduling algorithm for cloud environment using fuzzy approach," in *International Conference on Computer Networks and Communication Technologies : ICCNCT 2018*, pp. 819–828, Springer, 2019.

[93] H. Chen, Q. Liu, and Q. Ai, "A new heuristic scheduling strategy lbmm in cloud computing," in *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 1, pp. 314–317, IEEE, 2016.

[94] M. Ibrahim, S. Nabi, R. Hussain, M. S. Raza, M. Imran, S. A. Kazmi, A. Oracevic, and F. Hussain, "A comparative analysis of task scheduling approaches in cloud computing," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 681–684, IEEE, 2020.

[95] K. Dubey, M. Kumar, and S. C. Sharma, "Modified heft algorithm for task scheduling in cloud environment," *Procedia Computer Science*, vol. 125, pp. 725–732, 2018.

[96] S. Seth and N. Singh, "Dynamic heterogeneous shortest job first (dhsjf) : a task scheduling approach for heterogeneous cloud computing systems," *International Journal of Information Technology*, vol. 11, no. 4, pp. 653–657, 2019.

[97] N. Venkataraman, "Threshold based multi-objective memetic optimized round robin scheduling for resource efficient load balancing in cloud," *Mobile Networks and Applications*, vol. 24, pp. 1214–1225, 2019.

[98] H. Krishnaveni and V. S. Janita, "Completion time based sufferage algorithm for static task scheduling in cloud environment," *Int. J. Pure Appl. Math.*, vol. 119, no. 12, pp. 13793–13797, 2018.

[99] S. Elmougy, S. Sarhan, and M. Joundy, "A novel hybrid of shortest job first and round robin with dynamic variable quantum time task scheduling technique," *Journal of Cloud computing*, vol. 6, no. 1, pp. 1–12, 2017.

[100] M. A. Alworafi, A. Dhari, S. A. El-Booz, A. A. Nasr, A. Arpitha, and S. Mallappa, "An enhanced task scheduling in cloud computing based on hybrid approach," in *Data Analytics and Learning : Proceedings of DAL 2018*, pp. 11–25, Springer, 2019.

[101] H. S. Caranto, W. C. L. Olivete, J. V. D. Fernandez, C. A. R. Cabiara, R. B. M. Baquirin, E. F. G. Bayani, and R. J. D. Fronda, "Integrating user-defined priority tasks in a shortest job first round robin (sjfrr) scheduling algorithm," in *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, pp. 9–13, 2020.

[102] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing : A literature survey," *Future Generation Computer Systems*, vol. 91, pp. 407–415, 2019.

[103] I. M. Ibrahim *et al.*, "Task scheduling algorithms in cloud computing : A review," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 4, pp. 1041–1053, 2021.

[104] A. Khalili and S. M. Babamir, "Makespan improvement of pso-based dynamic scheduling in cloud environment," in *2015 23rd Iranian Conference on Electrical Engineering*, pp. 613–618, IEEE, 2015.

[105] D. Gabi, A. S. Ismail, and N. M. Dankolo, "Minimized makespan based improved cat swarm optimization for efficient task scheduling in cloud datacenter," in *Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference*, pp. 16–20, 2019.

[106] M. Abd Elaziz, S. Xiong, K. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Systems*, vol. 169, pp. 39–52, 2019.

[107] C. Malik, S. Jain, and S. Randhawa, "Resource scheduling in cloud using harmony search," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 2, pp. 1–6, IEEE, 2016.

[108] J. G. D. Matos, C. K. D. M. Marques, and C. H. Liberalino, "Genetic and static algorithm for task scheduling in cloud computing," *International Journal of Cloud Computing*, vol. 8, no. 1, pp. 1–19, 2019.

[109] L. Abualigah and M. Alkhrabsheh, "Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 740–765, 2022.

[110] I. Attiya, M. Abd Elaziz, S. Xiong, *et al.*, "Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm," *Computational intelligence and neuroscience*, vol. 2020, 2020.

[111] X. Huang, Y. Lin, Z. Zhang, X. Guo, and S. Su, "A gradient-based optimization approach for task scheduling problem in cloud computing," *Cluster Computing*, vol. 25, no. 5, pp. 3481–3497, 2022.

[112] P. Rekha and M. Dakshayini, "Efficient task allocation approach using genetic algorithm for cloud environment," *Cluster Computing*, vol. 22, no. 4, pp. 1241–1251, 2019.

[113] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," *Neural Computing and Applications*, vol. 32, pp. 1531–1541, 2020.

[114] A. M. Manasrah and H. Ba Ali, "Workflow scheduling using hybrid ga-pso algorithm in cloud computing," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–16, 2018.

[115] Z. Zhou, J. Chang, Z. Hu, J. Yu, and F. Li, "A modified pso algorithm for task scheduling optimization in cloud computing," *Concurrency and Computation : Practice and Experience*, vol. 30, no. 24, p. e4970, 2018.

[116] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, and A. El-Sayed, "Cloudlet scheduling based load balancing on virtual machines in cloud computing environment," *Journal of Internet Technology*, vol. 20, no. 5, pp. 1371–1378, 2019.

[117] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, and A. El-Sayed, "Cost-effective algorithm for workflow scheduling in cloud computing under deadline constraint," *Arabian Journal for Science and Engineering*, vol. 44, pp. 3765–3780, 2019.

[118] S. Rani and P. Suri, "An efficient and scalable hybrid task scheduling approach for cloud environment," *International Journal of Information Technology*, vol. 12, pp. 1451–1457, 2020.

[119] D. Chaudhary and B. Kumar, "Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing," *Applied Soft Computing*, vol. 83, p. 105627, 2019.

[120] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, and J. Murphy, "A woa-based optimization approach for task scheduling in cloud computing systems," *IEEE Systems journal*, vol. 14, no. 3, pp. 3117–3128, 2020.

[121] P. Han, C. Du, and J. Chen, "A dea based hybrid algorithm for bi-objective task scheduling in cloud computing," in *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 63–67, IEEE, 2018.

[122] W. H. El-Ashmawi and A. F. Ali, "A modified salp swarm algorithm for task assignment problem," *Applied Soft Computing*, vol. 94, p. 106445, 2020.

[123] G. Natesan and A. Chokkalingam, "An improved grey wolf optimization algorithm based task scheduling in cloud computing environment.," *Int. Arab J. Inf. Technol.*, vol. 17, no. 1, pp. 73–81, 2020.

[124] S. M. G. Kashikolaei, A. A. R. Hosseinabadi, B. Saemi, M. B. Shareh, A. K. Sangaiah, and G.-B. Bian, "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," *The Journal of Supercomputing*, vol. 76, pp. 6302–6329, 2020.

[125] S. T. Milan, L. Rajabion, A. Darwesh, M. Hosseinzadeh, and N. J. Navimipour, "Priority-based task scheduling method over cloudlet using a swarm intelligence algorithm," *Cluster Computing*, vol. 23, pp. 663–671, 2020.

[126] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Computing*, vol. 26, no. 23, pp. 13069–13079, 2022.

[127] A. N. Malti, M. Hakem, and B. Benmammar, "Multi-objective task scheduling in cloud computing," *Concurrency and Computation : Practice and Experience*, vol. 34, no. 25, p. e7252, 2022.

[128] T. Bezdan, M. Zivkovic, E. Tuba, I. Strumberger, N. Bacanin, and M. Tuba, "Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm," in *International Conference on Intelligent and Fuzzy Systems*, pp. 718–725, Springer, 2020.

[129] M. Sardaraz and M. Tahir, "A hybrid algorithm for scheduling scientific workflows in cloud computing," *IEEE Access*, vol. 7, pp. 186137–186146, 2019.

[130] I. Casas, J. Taheri, R. Ranjan, and A. Y. Zomaya, "Pso-ds : a scheduling engine for scientific workflow managers," *The Journal of Supercomputing*, vol. 73, no. 9, pp. 3924–3947, 2017.

[131] M. Adhikari, T. Amgoth, and S. N. Srirama, "Multi-objective scheduling strategy for scientific workflows in cloud environment : A firefly-based approach," *Applied Soft Computing*, vol. 93, p. 106411, 2020.

[132] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Computing and Applications*, pp. 1–14, 2021.

[133] M. Imdoukh, I. Ahmad, and M. Alfailakawi, "Optimizing scheduling decisions of container management tool using many-objective genetic algorithm," *Concurrency and Computation : Practice and Experience*, vol. 32, no. 5, p. e5536, 2020.

[134] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i : solving problems with box constraints," *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577–601, 2013.

[135] K. K. Chakravarthi and L. Shyamala, "Topsis inspired budget and deadline aware multi-workflow scheduling for cloud computing," *Journal of Systems Architecture*, vol. 114, p. 101916, 2021.

[136] R. Medara and R. S. Singh, "Energy efficient and reliability aware workflow task scheduling in cloud environment," *Wireless Personal Communications*, pp. 1–20, 2021.

[137] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[138] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 781–786, IEEE, 2012.

[139] P. Han, C. Du, J. Chen, F. Ling, and X. Du, "Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique," *Journal of Systems Architecture*, vol. 112, p. 101837, 2021.

[140] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft," *Future Generation Computer Systems*, vol. 93, pp. 278–289, 2019.

[141] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm : Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[142] A. K. A. Talukder, M. Kirley, and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global grids," *Concurrency and Computation : Practice and Experience*, vol. 21, no. 13, pp. 1742–1756, 2009.

[143] S. R. Thennarasu, M. Selvam, and K. Srihari, "A new whale optimizer for workflow scheduling in cloud computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 3807–3814, 2021.

[144] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Computing*, vol. 24, no. 1, pp. 205–223, 2021.

[145] T. Menouer and P. Darmon, "New scheduling strategy based on multi-criteria decision algorithm," in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 101–107, IEEE, 2019.

[146] T. P. Jacob and K. Pradeep, "A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization," *Wireless Personal Communications*, vol. 109, no. 1, pp. 315–331, 2019.

[147] J. K. Samriya and N. Kumar, "An optimal sla based task scheduling aid of hybrid fuzzy topsis-pso algorithm in cloud environment," *Materials Today : Proceedings*, vol. ., p. ., 2020.

[148] M. N. Kabir, J. Ali, A. A. Alsewari, and K. Z. Zamli, "An adaptive flower pollination algorithm for software test suite minimization," in *2017 3rd international conference on electrical information and communication technology (EICT)*, pp. 1–5, IEEE, 2017.

[149] S. Ulusoy, S. M. Nigdeli, and G. Bekdaş, "Novel metaheuristic-based tuning of pid controllers for seismic structures and verification of robustness," *Journal of Building Engineering*, vol. 33, p. 101647, 2021.

[150] H. R. Patel and V. A. Shah, "Application of metaheuristic algorithms in interval type-2 fractional order fuzzy tid controller for nonlinear level control process under actuator and system component faults," *International Journal of Intelligent Computing and Cybernetics*, vol. 14, no. 1, pp. 33–53, 2021.

[151] Z. A. A. Alyasseri, A. T. Khader, M. A. Al-Betar, A. K. Abasi, and S. N. Makhadmeh, "Eeg signals denoising using optimal wavelet transform hybridized with efficient metaheuristic methods," *IEEE Access*, vol. 8, pp. 10584–10605, 2019.

[152] I. Gupta, A. Kaswan, and P. K. Jana, "A flower pollination algorithm based task scheduling in cloud computing," in *International Conference on Computational Intelligence, Communications, and Business Analytics*, pp. 97–107, Springer, 2017.

[153] T. Bezdan, M. Zivkovic, M. Antonijevic, T. Zivkovic, and N. Bacanin, "Enhanced flower pollination algorithm for task scheduling in cloud computing environment," in *Machine Learning for Predictive Analysis*, pp. 163–171, Springer, 2021.

[154] N. K. Walia, N. Kaur, M. Alowaidi, K. S. Bhatia, S. Mishra, N. K. Sharma, S. K. Sharma, and H. Kaur, "An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments," *IEEE Access*, pp. 117325–117337, 2021.

[155] A. N. Malti, B. Benmammar, and M. Hakem, "Task scheduling in cloud computing based on fpa metaheuristic algorithm," in *2022 19th International Multi-Conference on Systems, Signals & Devices (SSD)*, pp. 41–46, IEEE, 2022.

[156] A. N. Malti, B. Benmammar, and M. Hakem, "Qos based task scheduling algorithm in cloud computing," in *E3S Web of Conferences*, vol. 351, p. 01014, EDP Sciences, 2022.

[157] X.-S. Yang, "Flower pollination algorithm for global optimization," in *International conference on unconventional computing and natural computation*, pp. 240–249, Springer, 2012.

[158] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, 2007.

[159] B. Benmammar, Y. Benmouna, and F. Krief, "A pareto optimal multi-objective optimisation for parallel dynamic programming algorithm applied in cognitive radio ad hoc networks," *International Journal of Computer Applications in Technology*, vol. 59, no. 2, pp. 152–164, 2019.

[160] C.-L. Hwang and K. Yoon, "Methods for multiple attribute decision making," in *Multiple attribute decision making*, pp. 58–191, Springer, 1981.

[161] X. Wang, C. S. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1124–1134, 2011.

[162] M. N. Aktan and H. Bulut, "Metaheuristic task scheduling algorithms for cloud computing environments," *Concurrency and Computation : Practice and Experience*, vol. 34, no. 9, p. e6513, 2022.

[163] K. K. Chakravarthi, P. Neelakantan, L. Shyamala, and V. Vaidehi, "Reliable budget aware workflow scheduling strategy on multi-cloud environment," *Cluster Computing*, vol. 25, no. 2, pp. 1189–1205, 2022.

[164] T. R. Newman, B. A. Barker, A. M. Wyglinski, A. Agah, J. B. Evans, and G. J. Minden, "Cognitive engine implementation for wireless multicarrier transceivers," *Wireless communications and mobile computing*, vol. 7, no. 9, pp. 1129–1142, 2007.

[165] H. S. Narman, M. S. Hossain, and M. Atiquzzaman, "h-ddss : Heterogeneous dynamic dedicated servers scheduling in cloud computing," in *2014 IEEE International Conference on Communications (ICC)*, pp. 3475–3480, IEEE, 2014.

[166] G. da Cunha Rodrigues, R. N. Calheiros, V. T. Guimaraes, G. L. d. Santos, M. B. De Carvalho, L. Z. Granville, L. M. R. Tarouco, and R. Buyya, "Monitoring of cloud computing environments : concepts, solutions, trends, and future directions," in *Proceedings of the 31st annual ACM symposium on applied computing*, pp. 378–383, 2016.

[167] R. Y. Shtykh, Y. Zhu, and Q. Jin, "A context-aware framework for flowable services," in *2009 Third International Conference on Multimedia and Ubiquitous Engineering*, pp. 251–256, IEEE, 2009.

[168] N. Miglani, G. Sharma, and S. Khurana, "Multi-objective reliability-based workflow scheduler : An elastic and persuasive task scheduler based upon modified-flower pollination algorithm in cloud environment," *Concurrency and Computation : Practice and Experience*, vol. 34, no. 22, p. e7150, 2022.

[169] M. Gokuldhev, G. Singaravel, and N. Ram Mohan, "Multi-objective local pollination-based gray wolf optimizer for task scheduling heterogeneous cloud environment," *Journal of Circuits, Systems and Computers*, vol. 29, no. 07, p. 2050100, 2020.

[170] M. Gokuldhev and G. Singaravel, "Local pollination-based moth search algorithm for task-scheduling heterogeneous cloud environment," *The Computer Journal*, vol. 65, no. 2, pp. 382–395, 2022.

[171] D. Alsadie, "Tsmgwo : Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers," *IEEE Access*, vol. 9, pp. 37707–37725, 2021.

[172] G. Natesan and A. Chokkalingam, "Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm," *ICT Express*, vol. 5, no. 2, pp. 110–114, 2019.

[173] A. Khalili and S. M. Babamir, "Optimal scheduling workflows in cloud computing environment using pareto-based grey wolf optimizer," *Concurrency and Computation : Practice and Experience*, vol. 29, no. 11, p. e4044, 2017.

[174] N. Gobalakrishnan and C. Arun, "A new multi-objective optimal programming model for task scheduling using genetic gray wolf optimization in cloud computing," *The Computer Journal*, vol. 61, no. 10, pp. 1523–1536, 2018.

[175] S. Khurana and R. Singh, "Workflow scheduling and reliability improvement by hybrid intelligence optimization approach with task ranking," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 7, no. 24, pp. 1–10, 2019.

[176] D. A. Amer, G. Attiya, I. Zeidan, and A. A. Nasr, "Elite learning harris hawks optimizer for multi-objective task scheduling in cloud computing," *The Journal of Supercomputing*, pp. 1–26, 2022.

[177] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[178] K. Meidani, A. Hemmasian, S. Mirjalili, and A. Barati Farimani, "Adaptive grey wolf optimizer," *Neural Computing and Applications*, vol. 34, no. 10, pp. 7711–7731, 2022.

[179] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *J. Water Resour. Plann Manag*, vol. 129, no. 3, pp. 210–225, 2003.

[180] M. Karpagam, K. Geetha, and C. Rajan, "A modified shuffled frog leaping algorithm for scientific workflow scheduling using clustering techniques," *Soft Computing*, vol. 24, no. 1, pp. 637–646, 2020.

[181] B. Benmammar, "Quality of service optimization in orthogonal frequency division multiplexing-based cognitive radio systems based on shuffled frog leaping algorithm," *Concurr. Comput. Pract. Exp.*, vol. 34, no. 1, p. e6530, 2022.

[182] A. N. Malti, M. Hakem, and B. Benmammar, "A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems," *Cluster Computing*, pp. 1–24, 2023.

[183] X. Chen, W. Du, and F. Qian, "Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization," *Chemometrics and Intelligent Laboratory Systems*, vol. 136, pp. 85–96, 2014.

[184] A. Özkış and A. Babalık, "A novel metaheuristic for multi-objective optimization problems : The multi-objective vortex search algorithm," *Information Sciences*, vol. 402, pp. 124–148, 2017.

[185] A. N. Malti, M. Hakem, and B. Benmammar, "An improved hybrid multi-objective optimization algorithm for task scheduling in cloud environments," (submitted).

[186] M. Zeedan, G. Attiya, and N. El-Fishawy, "Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing," *Computing*, vol. 105, no. 1, pp. 217–247, 2023.

[187] A. N. Malti, B. Benmammar, and M. Hakem, "A comparative study of metaheuristics based task scheduling in cloud computing," in *International Symposium on Modelling and Implementation of Complex Systems*, pp. 263–278, Springer, 2022.

[188] A. Hussain, M. Aleem, M. A. Iqbal, and M. A. Islam, "Investigation of cloud scheduling algorithms for resource utilization using cloudsim.," *Computing & Informatics*, vol. 38, no. 3, 2019.

[189] T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment," in *2014 International conference on advances in computing, communications and informatics (ICACCI)*, pp. 658–664, IEEE, 2014.

[190] Z. Zhong, K. Chen, X. Zhai, and S. Zhou, "Virtual machine-based task scheduling algorithm in a cloud computing environment," *Tsinghua Science and Technology*, vol. 21, no. 6, pp. 660–667, 2016.

[191] X. Zhu, C. Chen, L. T. Yang, and Y. Xiang, "Angel : Agent-based scheduling for real-time tasks in virtualized clouds," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3389–3403, 2015.

[192] A. Hussain and M. Aleem, "Gocj : Google cloud jobs dataset for distributed and cloud computing infrastructures," *Data*, vol. 3, no. 4, p. 38, 2018.

[193] "The hpc2n seth log." Accédé le 03/01/2024, https://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/index.html.

[194] S. Behzad, R. Fotohi, and M. Effatparvar, "Queue based job scheduling algorithm for cloud computing," *International Research Journal of Applied and Basic Sciences ISSN*, vol. 37853790, 2013.

[195] N. A. Mehdi, A. Mamat, H. Ibrahim, and S. K. Subramaniam, "Impatient task mapping in elastic cloud using genetic algorithm," *Journal of Computer Science*, vol. 7, no. 6, pp. 877–883, 2011.

[196] "Gocj : Google cloud jobs dataset." Accédé le 03/01/2024, https://data.mendeley.com/datasets/b7bp6xhrcd/1.

[197] "Amazon ec2 on-demand pricing." Accédé le 03/01/2024, https://aws.amazon.com/fr/ec2/pricing/on-demand/.

[198] A. Belgacem and K. Beghdad-Bey, "Multi-objective workflow scheduling in cloud computing : trade-off between makespan and cost," *Cluster Computing*, vol. 25, no. 1, pp. 579–595, 2022.

[199] P. J. Gaidhane and M. J. Nigam, "A hybrid grey wolf optimizer and artificial bee colony algorithm for enhancing the performance of complex systems," *Journal of computational science*, vol. 27, pp. 284–302, 2018.

Résumé

De nos jours, le Cloud Computing est largement adopté dans divers domaines et son expansion s'accompagne des défis et des contraintes dont les organisations doivent prendre en compte pour en tirer pleinement parti. L'ordonnancement des tâches émerge comme un enjeu clé pour améliorer les performances. Malheureusement, il est bien connu que ce problème est NP-difficile, nécessitant des approches judicieuses. La plupart des recherches antérieures se concentrent sur un seul objectif, et celles qui considèrent plusieurs objectifs utilisent souvent des fonctions de compromis simples, négligeant l'influence mutuelle des paramètres. De plus, le phénomène de stagnation des solutions locales et la convergence prématurée entravent le processus de recherche. Pour surmonter ces limites, cette thèse propose de nouveaux algorithmes d'optimisation basés sur des approches bio-inspirées, notamment le comportement de pollinisation des fleurs, la capacité d'exploration de l'optimiseur du loup gris, les opérateurs de croisement des algorithmes évolutionnaires, et la stratégie employée dans l'algorithme de saut de grenouille. Ces approches sont appliquées à l'ordonnancement des tâches dans les systèmes de Cloud hétérogènes, formulé sous forme d'un problème d'optimisation multi-objectif. Les résultats obtenus à travers une série de tests et de simulations, avec des charges de travail synthétiques et standard, démontrent la supériorité de nos algorithmes en termes de qualité des solutions et de vitesse de convergence par rapport à d'autres techniques récemment proposées dans la littérature.

Mots clés : *algorithmes distribués*, *cloud computing*, *optimisation multi-objectif*, *ordonnancement des tâches*, *méta-heuristiques*, *optimisation combinatoire*.

Abstract

Nowadays, Cloud Computing is widely adopted in various domains and its expansion is accompanied by challenges and constraints that organizations must to be aware of and address to fully harness its power. Task scheduling has emerged as a key concern for enhancing performances. Unfortunately, it is well-known that this problem is NP-hard, requiring judicious approaches. Most previous research focuses on a single objective, and those dealing multiple objectives often use a simple compromise function and do not consider how each of the parameters might influence the others. Additionally, the stagnation phenomenon of local solutions and the premature convergence hinder the research process. To overcome these limitations, this thesis proposes new optimization algorithms based on bio-inspired approaches, including the flower pollination behavior, the exploration capability of the gray wolf optimizer, evolutionary algorithms crossover operators, and the strategy employed in the frog algorithm. These approaches are applied to task scheduling in heterogeneous Cloud systems, formulated as a multi-objective optimization problem. The results obtained through a series of tests and simulations, with synthetic and standard workloads, demonstrate the superiority of our algorithms in terms of solution quality and convergence speed compared to other techniques recently proposed in the literature.

Keywords : *distributed algorithms*, *cloud computing*, *multi-objective optimization*, *task scheduling*, *metaheuristics*, *combinatorial optimization*.

ملخص

في الوقت الحاضر، يتم اعتماد الحوسبة السحابية على نطاق واسع في مجالات مختلفة، ويصاحب توسعها تحديات وقيود يجب على المؤسسات أن تكون على دراية بها ومعالجتها لتسخير قوتها بشكل كامل. لقد برزت جدولة المهام كمشكلة رئيسية لتحسين أداء أنظمة الحوسبة السحابية. لسوء الحظ، فمن المعروف جيدًا أن هذه المشكلة صعبة، وحلها يتطلب أساليب حكيمة. تركز معظم الأبحاث السابقة على هدف واحد، وغالبًا ما يستخدم أولئك الذين يتعاملون مع أهداف متعددة وظيفة تسوية بسيطة دون مراعاة التأثير المتبادل للمعلمات. إضافة إلى ذلك، فإن ظاهرة ركود الحلول المحلية والتقارب المبكر يعيقان عملية البحث. للتغلب على هذه القيود، تقترح هذه الأطروحة خوارزميات تحسين جديدة تعتمد على أساليب مستوحاة من العوامل البيولوجية، بما في ذلك سلوك تلقيح الزهور، وقدرة الاستكشاف لمُحسِنْ الذئب الرمادي، وعمليات التهجين في الخوارزميات التطورية، والاستراتيجية المستخدمة في خوارزمية الضفدع. يتم تطبيق هذه الأساليب على جدولة المهام في الأنظمة السحابية غير المتجانسة، والتي تمت صياغتها كمشكلة تحسين متعددة الأهداف. تُظهر النتائج التي تم الحصول عليها من خلال سلسلة من الاختبارات والمحاكاة، مع أعباء العمل التركيبية والقياسية، تفوق خوارزمياتنا من حيث جودة الحلول وسرعة التقارب مقارنة بالتقنيات المقترحة مؤخرًا في الأدبيات.

الكلمات المفتاحية: الخوارزميات الموزعة ، حوسبة سحابية ، التحسين متعدد الأهداف ، جدولة المهام ، الاستدلال الفوقية ، تحسين تركيبي .