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

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Réseaux et Systèmes Distribués (RSD)*

Thème

Grey Wolf Optimization (GWO) pour l'ordonnancement des tâches dans le Cloud computing

Réalisé par :

- *MEBARKI Boumediene*

Présenté le 29 Juin 2022 devant le jury composé de MM.

- *Mr Mohamed Lehsaini* *(Président)*
- *Mr Badr Benmammar* *(Encadrant)*
- *Mr Arslan Nedhir Malti* *(Co-encadrant)*
- *Mr Benmouna Youcef* *(Examinateur)*

Année universitaire: 2021-2022

Remerciement

Je remercie DIEU le tout puissant, maître des cieux et de la terre, qui m'a éclairé le chemin et permis de mener à bien ce travail.

Je tiens à remercie aussi mon encadrant Mr BENMAMMAR Badr ainsi que mon co-encadrant Mr Arslan Nedhir Malti de leurs soutiens durant ce PFE. JE ne saurai jamais oublier leurs conseils, orientations, encouragements tout au long de ce travail.

Aux membres du jury Mr Mohamed Lehsaini et Mr Benmouna Youcef, pour avoir accepté d'évaluer notre travail.

Un merci pour tous mes enseignants qui m'ont permis de travailler dans un environnement serein et paisible.

Un grand merci à toutes les personnes qui m'ont soutenue de près ou de loin au cours de la réalisation de ce modeste travail.

Dédicace

Je dédie ce modeste travail

A mes chers parents qui n'ont jamais cessé de me montrer amour et soutient tout au long de ma vie.

A toute ma famille,

A tous mes amies.

Mebarki Boumediene

Table des matières

Liste des figures…….....…i

Liste des tableaux…….……..……...ii

Liste des abréviations……..…….....iii

Introduction générale……..……..……...……...…….1

CHAPITRE 1 : Cloud computing……...…….……2

1.1 Introduction.....2

1.2 Concepts de base.....2

1.2.1 Caractéristiques principales du Cloud computing 2

1.2.2 Technologies connexes.....4

1.2.3 DataCenter……..……..…….……6

1.3 Modèles de déploiement 7

1.3.1 Cloud public.....7

1.3.2 Cloud privé 7

1.3.3 Cloud communautaire.....7

1.3.4 Cloud hybride 7

1.4 Architecture de Cloud Computing.....8

1.4.1 SaaS (Software as a Services).....9

1.4.2 PaaS (Plateforme as a Services).....9

1.4.3 IAAS (Infrastructure as a Services) 9

1.5 Avantages et inconvénients.....10

1.5.1 Avantages.....10

1.5.2 Inconvénients 11

1.6 Conclusion : 12

CHAPITRE 2: Les métaheuristiques……..…….……..…13

2.1 Introduction.....13

2.2 Méthodes de résolution.....13

2.2.1 Méthodes exactes.....14

2.2.2 Méthodes approchées.....14

2.2.2.1 Heuristiques.....15

2.2.2.2 Métaheuristiques.....15

2.3 Caractéristiques principales des métaheuristiques.....17

2.4 Algorithme basés sur l'intelligence en essaim.....17

2.4.1 Algorithme de colonies d'abeilles.....18

2.4.2 Algorithme de la recherche coucou.....18

2.4.3 Algorithme d'optimisation des chats.....19

2.4.4 Algorithme d'optimisation des loups gris.....19

2.4.4.1 Caractéristique du loup gris.....20

2.4.4.2 Description de l'algorithme.....22

2.5 Conclusion.....26

CHAPITRE 3: Implémentation de l'application et évaluations des résultats……...…27

3.1 Introduction.....27

3.2 Environnement de développement et de simulation.....27

3.2.1 JAVA.....27

3.2.2 Netbeans.....28

3.2.3 Jfreechart.....28

3.2.4 CloudSim.....28

3.2.5 Modélisations des machines virtuelles et des cloudlets.....29

3.3 Techniques d'optimisation multicritères.....30

3.4 Approche et démarche proposées 31
3.4.1 Evaluation multi-objective.....31
3.4.2 Adaptation de la métaheuristique GWO.....32
3.5 IHM développée.....33
3.6 Configuration utilisée.....34
3.6.1 Configuration des machines physiques.....34
3.6.2 Configuration des machines virtuelles.....35
3.6.3 Configuration des Cloudlets 36
3.6.4 La simulation 36
3.7 Résultats obtenus et étude comparative 37
3.7.1 Configuration des différentes simulations 37
3.7.2 Evaluation de l'ordonnancement multi-objectif 38
3.7.2.1 Comparaison des résultats par rapport au makespan.....38
3.7.2.2 Comparaison des résultats par rapport au coût.....39
3.8 Conclusion 42
Conclusion générale……...…….……43
Bibliographie…….……..……....…44

Liste des figures

Figure 1.1 : Convergence des différentes avancées menant à l'avènement du cloud computing [5]..... .5

Figure 1.2 : Une architecture en couches de la technologie de virtualisation [5]..... 6

Figure 1.3: Modèles de déploiement de Cloud computing [9]. 8

Figure 1.4 : Modèles de service en nuage [9]. ..... .10

Figure 2.1 : Principes Généraux d'une métaheuristique à base de: (a) solution unique, (b) solution population [13]..... ..14

Figure 2.2 : Principes Généraux d'une métaheuristique à base de: (a) solution unique, (b) solutionpopulation…..……...…….....….…16

Figure 2.3 : Hiérarchie du loup gris (la dominance diminue de haut en bas) [20] …….20

Figure 2.4 : La mise à jour de la position dans GWO [19] …..……..…….24

Figure 3.5 : Mécanisme de positionnement de l'agent de recherche et de l'effet de ce qui le présent [21] ……..…….……..…..25

Figure 3.1 : Architecture de CloudSim [26]…… ……..…….....…29

Figure 3.2 : Interface principale…….……...…...33

Figure 3.3 : Configuration des machines physiques…….....…34

Figure 3.4 : Configuration des machines virtuelles.....….35

Figure 3.5 : Configuration des cloudlets……..……..…….36

Figure 3.6 : Interface de simulation…….……36

Figure 3.7 : Comparaison en termes de makespan pour 60 VMs……38

Figure 3.8 : Comparaison en termes de makespan pour 80 VMs…….……..…..…38

Figure 3.9 : Comparaison en termes de makespan pour 100 VMs…….…….....39

Figure3.10 : Comparaison en termes de coût pour 60 VMs…….……...39

Figure 3.11 : Comparaison en termes de coût pour 80 VMs……..…….40

Figure 3.12 : Comparaison en termes de coût pour 1000 VMs……...…40

Liste des tables

Tableau 3.1 : Les paramètres de simulation de CloudSim.....37

Liste des Abréviations

Acronyme	Signification
ABC	Artificial Bee Colony
API	Application Program Interface
BA	Bat Algorithm
CPU	Central Processing Unit
CS	Cuckoo Search
CSO	Cat Swarm Optimization
DS	Data Center
GWO	Grey Wolf Optimization
IDE	Integrated Development Environment
IHM	Interactions Homme-Machine
JVM	Java Virtual Machine
LGPL	Lesser General Public Licence
MCDM	Multi-Criteria Decision Making
MIPS	Million Instructions Per Second
PE	Processing Element
QoS	Quality Of Service
RAM	Random Access Memory
SLA	Service Level Agreement
VM	Virtual Machine
WSM	Weight Sum Method

Introduction générale

Le domaine informatique a connu ces dernières années une large évolution du point de vue de nouvelles technologies. En particulier l'apparition du Cloud computing, qui permet actuellement la simplification et l'exécution d'une multitude de services flexibles dans tous les domaines tels que les communications, le champ scientifique, l'espace commercial et d'autres horizons.

L'ordonnancement des tâches est un pari important dans le domaine du Cloud computing qui agit énormément sur les performances de cet environnement. Dans ce cadre, il faut souligner que les opérateurs et les utilisateurs des Cloud computing sont exigeants et ont des buts souvent opposés.

Pour qu'il puisse y avoir un bon ordonnancement, il faut qu'il y ait un accord acceptable entre ses objectifs. En outre, dans le Cloud computing l'ordonnancement des tâches se transforme en un problème d'optimisation multi-objectif. C'est un problème NP-complet lorsqu'il peut être résolu par une classe restreinte d'algorithmes qui demande l'utilisation des métaheuristiques pour sa résolution.

Dans le cadre de ce projet de fin d'études, notre contribution consiste à l'adaptation et à l'application de la métaheuristique GWO (Grey Wolf Optimization) pour l'optimisation d'un certains nombre d'objectifs dans le Cloud computing. Pour cela, nous avons étudié et évalué deux critères de QoS qui sont le makespan et le coût.

Le simulateur CloudSim a été utilisé dans ce travail et les résultats obtenus à travers les différentes simulations sont très satisfaisants.

Le reste de ce manuscrit est constitué de 3 chapitres :

Le premier d'entre eau est dédié à la présentation du Cloud computing qui est le cadre général de notre étude.

Le deuxième chapitre est consacré à la présentation des métaheuristiques en s'intéressant en particulier à l'algorithme GWO que nous avons utilisé pour l'élaboration de ce travail.

Notre contribution dans le cadre de ce PFE est présentée dans le troisième et dernier chapitre de ce manuscrit. Nous présenterons : le simulateur CloudSim, l'IHM réalisée ainsi que les résultats des différentes simulations opérées. Les graphes de comparaison entre le makespan et le coût seront également détaillés et discutés dans ce chapitre.

CHAPITRE 1 :

Cloud computing

1.1 Introduction

Depuis son invention, l'informatique s'est toujours développé d'une manière spectaculaire pour s'imposer comme moyen de communication dans tous les domaines à l'échelle universelle. Durant ces dix dernières année, l'informatique a connu d'autres évolutions à tel point que cet outil est utilisé que ce soit par les institutions ou par le public, pour devenir un outil important dans la vie quotidienne d'où l'influence sur nos modes de vie et de travail est pratiquement présente [1].

A toutes ces technologies de l'information et de la communication qui bouleversent tout le temps, les modalités et les coutumes de la vie, est née ces dernières années un nouveau concept appelé Cloud computing.

L'idée du Cloud computing est apparue dans les années 90 avec l'avènement d'internet et la mise sur le marché public du logiciel CERIV (organisation Européenne pour la recherche nucléaire, un des plus grands centre de recherches scientifiques au monde). C'est à partir de 2000 que le concept du computing a commencé à connaître une véritable énergie avec l'arrivée de Google.

Le Cloud computing est un concept qui permet de déplacer ou de transférer sur des serveurs éloignés des informations, des fichiers ou des bases de données qui se trouvaient antérieurement sur l'ordinateur particulier de l'utilisateur. Il permet par exemple d'avoir accès sur demande à des fichiers informatisés exploitables par différentes personnes [2].

Pour d'amples explications, nous aborderons dans ce chapitre le Cloud computing en particulier particulière sons concept de base, ses modèles de déploiement ainsi que son architecture.

1.2 Concepts de base

1.2.1 Caractéristiques principales du Cloud computing

Le Cloud computing dénommé en français l'informatique en nuage permet d'accéder à divers services informatiques tels que des serveurs, dispositifs de stockage mise en réseau par l'intermédiaire d'internet à partir d'un fournisseur qui peut être rapidement mise en service et libéré avec un minimum d'interdépendance.

Dans ce qui suit nous présentons les principaux services que le Cloud computing propose aux utilisateurs :

- Libre service à la demande : Les utilisateurs qu'ils soient personne individuelle ou entreprise peuvent accéder aux différents services informatiques via le Cloud computing à chaque fois qu'elles en ont besoin. Ces services sont fournis automatiquement sont nécessiter l'intervention humaine.

  Ces services (de Cloud computing) sont proposés entièrement à la demande afin que les utilisateurs puissent avoir du contrôle et de la souplesse nécessaire pour répondre à l'amélioration de leurs besoins [3].
- Scalabilité et élasticité : Nous pouvons considérer que l'élasticité est le niveau auquel un système est apte de s'adapter aux demandes en approvisionnant et en désapprovisionnant de ressources de manière automatique.
- Autonomie : La caractéristique du Cloud computing représente un système autonome qui gère de manière transparente, pour les clients, le matériel, le logiciel. Les données au sein du Cloud computing peuvent être automatiquement reconfigurées et assurées en une seule image qui sera fournie à l'utilisateur [3].
- Payement à l'usage : la demande des ressources dans le Cloud computing s'adapte aux besoins de l'utilisateur. Le prestataire de service est apte de mesurer de façon précise la consommation du client en durée et en quantité, des différents services (CPU, stockage, bande passante,…..), cela lui permettra de facturer l'utilisateur selon sa réelle consommation [4].
- Fiabilité et tolérance aux pannes : La fiabilité constitue une phase indispensable dans toute étude d'un système. Elle désigne, son aptitude à assurer sa mission dans des conditions d'environnement données et pendant une durée donnée. Ainsi la fiabilité détermine la confiance que l'utilisateur peut mettre dans le service rendu par un système. La fiabilité repose sur la mise en place d'un logiciel à accomplir l'ensemble des fonctions spécifiées dans un environnement donné et pour un temps de fonctionnement donné. Quand la fiabilité est assurée, les spécifications floues, les incompréhensions entre les clients et les fournisseurs, perturbations de l'environnement disparaissent. Aujourd'hui, la fiabilité est devenue un paramètre clé de la qualité et d'aide à la décision, dans l'étude de la plupart des composants, ou d'un système.

  Dans le domaine de l'informatique. La tolérance des pannes reste toujours posée. C'est pourquoi, il est nécessaire, voire primordial de renforcer son processus ou son dispositif. La tolérance aux pannes fait référence à la caractéristique qui permet au système de continuer à fonctionner correctement

même en cas de défaillance de certains de ses composants. Autrement dit, la tolérance aux pannes signifie comment un système réagit et permet des dysfonctionnements et des défaillances matérielles ou logicielles. La capacité du système d'exploitation à récupérer et à tolérer les pannes peut être gérée par l'intermédiaire d'un logiciel, d'un matériel ou d'une solution combiné. Certains systèmes informatiques utilisent plusieurs systèmes de tolérance aux pannes pour gérer les pannes avec aisance,

- SLA (Service Level Agreement) (Accord du Niveau de Service) : Le SLA est un document qui définit la qualité de service C'est un genre de prestation admis mutuellement entre le fournisseur de service et l'utilisateur. Autrement dit, il s'agit de clauses basées sur un contrat définissant les objectifs précis attendus et le niveau de service que souhaite obtenir un client de la part du fournisseur en garantissant toutes les responsabilités.

  Le SLA tend à devenir un instrument important aux utilisateurs qui souhaitent bénéficier d'une sécurité évidente sur certains de leurs niveaux de sécurité de stockage ainsi que sur la gestion de leurs données à caractère individuel. De nombreux indicateurs doivent être définis, analysés et contrôlés afin que la performance proposée par le fournisseur puisse être au maximum [5].

1.2.2 Technologies connexes

De nombreuses années se sont écoulées avant que les technologies du Cloud computing ne se vulgarisent et deviennent habituelles à travers les sociétés à l'échelle mondiale. Les méfiants et les sceptiques ont cessé de remettre en question la durabilité du Cloud [5].

La figure 1.1 montre la convergence des technologies qui ont contribué à l'avènement du Cloud computing.

Le Cloud en tant que système adopte de nombreuses technologies entre autres, la virtualisation, l'architecture orientée services et les services Web. Le Cloud est fréquemment confondu avec plusieurs paradigmes tels que :

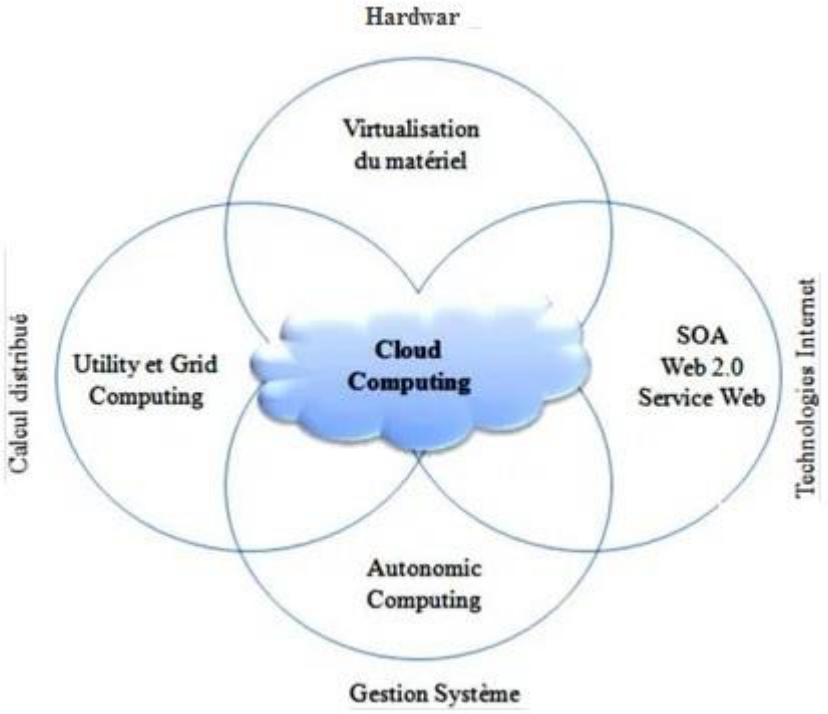


Figure 1.1 : Convergence des différentes avancées menant à l'avènement du cloud computing [5].

- Utility Computing (informatique utilitaire) est un transfert ou délocalisation d'un système de calcul ou de stockage. Il présente une forme de déplacement des ressources à la demande et la facturation des clients selon leur consommation.
- L'autonomie computing qui vise à mettre à jour des systèmes informatiques qui peuvent s'auto-administrer ou s'accommoder à des changements internes et externes sans que l'homme n'intervienne. L'objectif de l'informatique autonome est de surpasser les difficultés dans la conduite des systèmes informatiques.
- La virtualisation est une technologie qui ne tient pas compte des détails de l'équipement physique et fournit des ressources pour les applications de niveau supérieur. La virtualisation rassemble des procédés matériels ou logiciels pour que sur une seule machine physique, puissent fonctionner plusieurs configurations informatiques afin de composer plusieurs machines virtuelles qui recréent le comportement des machines physiques. Voir la Figure 1.2.

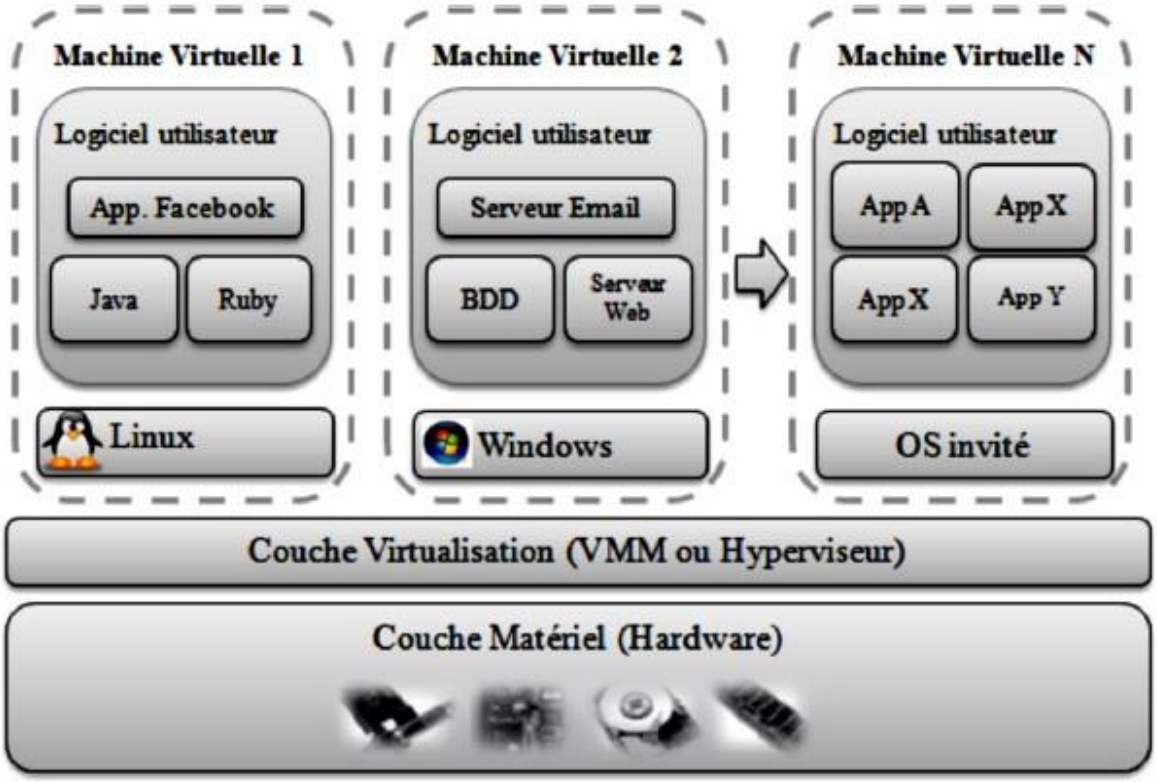


Figure 1.2 : Une architecture en couches de la technologie de virtualisation [5].

1.2.3 DataCenter

Un Datacenter ou centre de données est une infrastructure constituée d'un ensemble d'ordinateurs et d'espaces de stockage. Les entreprises peuvent exploiter cet espace pour l'organisation, le traitement, le stockage et l'entreposage d'une quantité importante de données.

Pour le fonctionnement adéquat, un centre de données doit offrir les conditions nécessaires pour un système de distribution, pour un système d'énergie, un commutateur électrique, des réserves d'énergie consacré au backup (sauvegarde informatique), un système d'aération ou de ventilation et de refroidissement et une forte connexion internet. Un Datacenter doit obligatoirement répondre aux normes et aux règles de sécurité. Celle-ci est garantie par la mise en place de grandes portes d'accès et de corridors qui doivent permettre le transport de larges équipements et le déplacement des employés pour effectuer des réparations à tout moment. Il est aussi impératif de mettre en place un système anti-incendie [6].

1.3 Modèles de déploiement

Dans le cadre du système informatique. Le déploiement renvoie ou fait référence au processus de configuration d'une machine ou d'un système pour le rendre prêt à fonctionner dans un environnement réel. C'est ainsi qu'au fur et à mesure que le Cloud computing se généralisait que plusieurs modèles ont émergé (Voir Figure 1.3).

1.3.1 Cloud public

Le Cloud public constitué de services informatiques dans une infrastructure dans laquelle le prestataire fournit des ressources au public par l'intermédiaire d'Internet. Les ressources peuvent inclure des capacités de stockage, ou des machines virtuelles [7].

1.3.2 Cloud privé

Le Cloud privé est un modèle informatique qui fournit un environnement, à une seule entité commerciale. Il fournit également des ressources informatiques grâce à des composants physiques stockées sur place ou dans des ces centres de données. L'avantage principal du déploiement d'un Cloud privé est la capacité du contrôle important mis à la disposition de l'entité. Celle-ci a la possibilité de configurer et gérer l'environnement en l'adoptant à ses besoins spécifiques [4].

1.3.3 Cloud communautaire

Le Cloud communautaire est une forme de service qui met en évidence une solution de Cloud computing à un nombre réduit de personnes ou d'organisations. Ce genre de Cloud communautaire est souvent créé pour les entreprises et les organisations œuvrant sur des projets, des applications ou des recherches communes qui nécessitent une installation centrale du Cloud computing pour gérer et exécuter de tels projets [8].

1.3.4 Cloud hybride

Le Cloud hybride est un modèle qui rassemble au moins un Cloud privé et un Cloud public qui opèrent ensemble pour livrer une combinaison maniable de services du Cloud computing. Le Cloud hybride offre le meilleur des Clouds public et privé, d'une part la disponibilité des services et des performances convenant en fonction des besoins et d'autre part par l'opportunité pour l'entreprise de maintenir des applications et des données sensibles. Le Cloud hybride permet aux entreprises un meilleur dynamisme et une gestion optimisée. Il faut préciser que Le Cloud hybride garantit une grande protection de données sensibles [8].

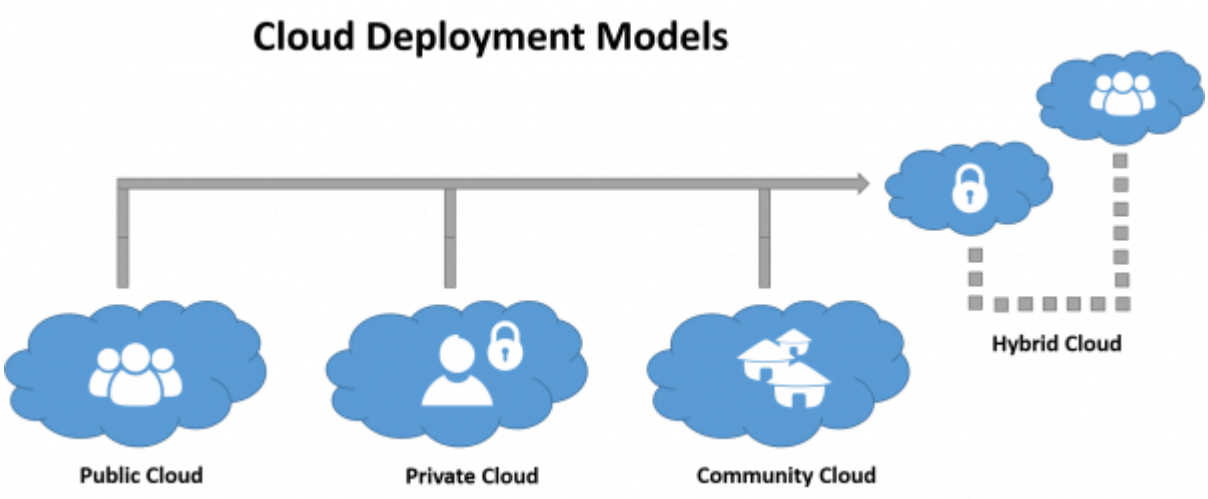


Figure 1.3 : Modèles de déploiement de Cloud computing [9].

1.4 Architecture de Cloud Computing

L'architecture du Cloud computing repose sur un modèle qui permet l'incorporation des technologies afin de créer des Cloud, des environnements qui décomposent, réunissent et partagent des ressources qui évoluent sur un réseau. L'architecture Cloud nous montre la façon dont les composants et les fonctionnalités indispensables pour la concrétisation d'un Cloud sont connectées pour fournir une plateforme en ligne sur laquelle les applications peuvent être existées.

Ainsi, l'architecture Cloud est la forme dont les composants technologiques s'agencent pour réaliser un Cloud dans lequel les ressources sont réunies par l'intermédiaire des technologies de virtualisation.

L'architecture Cloud computing facilite aux entreprises la réduction ou l'élimination de leur dépendance vis-à-vis d'une infrastructure de serveurs, de stockage et de réseau d'un site.

Il existe trois importants modèles d'architecture Cloud qui permettent aux entreprises d'opter pour le Cloud et chaque modèle possède ses propres avantages. Voir la Figure 1.4.

1.4.1 SaaS (Software as a Services)

Le Software as a service dont la traduction littérale est : les logiciels en tant que service, est une forme ou modèle concernant une explication commerciale de logiciels qui sont hébergés sur le serveur d'un fournisseur et qui sont accessibles à distance aux utilisateurs par exemple au travers un navigateur Web [10].

Dans le cas du SaaS, l'utilisateur ne paie pas de licence mais utilise librement le service en ligne et paye en contrepartie un abonnement.

1.4.2 PaaS (Plateforme as a Services)

La plateforme as a Service (PaaS) ou, plateforme en tant que service, est un modèle d'informatique en Cloud dans lequel un fournisseur met à la disposition des clients sur internet des outils matériels et logiciels, leur permettant de développer, d'exécuter et de gérer des applications commerciales sans avoir à créer et à maintenir l'infrastructure que ces processus de développement de logiciels exigent généralement.

La plateforme a as Service propose des solutions qui sont de plus en plus populaire auprès des entreprises qui veulent bénéficier d'une solution d'hébergement légère et flexible. En conséquence, l'utilisateur n'a pas besoin de construire, d'entretenir ou de développer les services et les logiciels auxquels le fournisseur lui donne accès [10].

1.4.3 IAAS (Infrastructure as a Services)

Infrastructure as a Service appelé par abréviation (IAAS) qui est créé pour fournir des ressources, de réseau et de stockage aux utilisateurs à la demande sur Internet en contrepartie d'un paiement à l'utilisation.

Cette forme de Cloud computing donne la possibilité aux consommateurs d'adopter et de réduire les ressources en fonction de leurs besoins, limitant ainsi certaines dépenses d'investissement initiales.

L'infrastructure as a Service est un modèle de Cloud computing particulièrement adapté aux entreprises qui ont déjà de grandes ressources en interne mais souhaitant plus de flexibilité.

L'atout du modèle (IAAS) est de passer d'un mode d'investissement où l'utilisateur doit mobiliser des capacités financières pour se doter en équipements informatiques à un modèle exclusivement locatif. Le payement se fait mensuellement sur la base de

factures de consommation effectives des services utilisés que le fournisseur remet à l'entrepris [11].

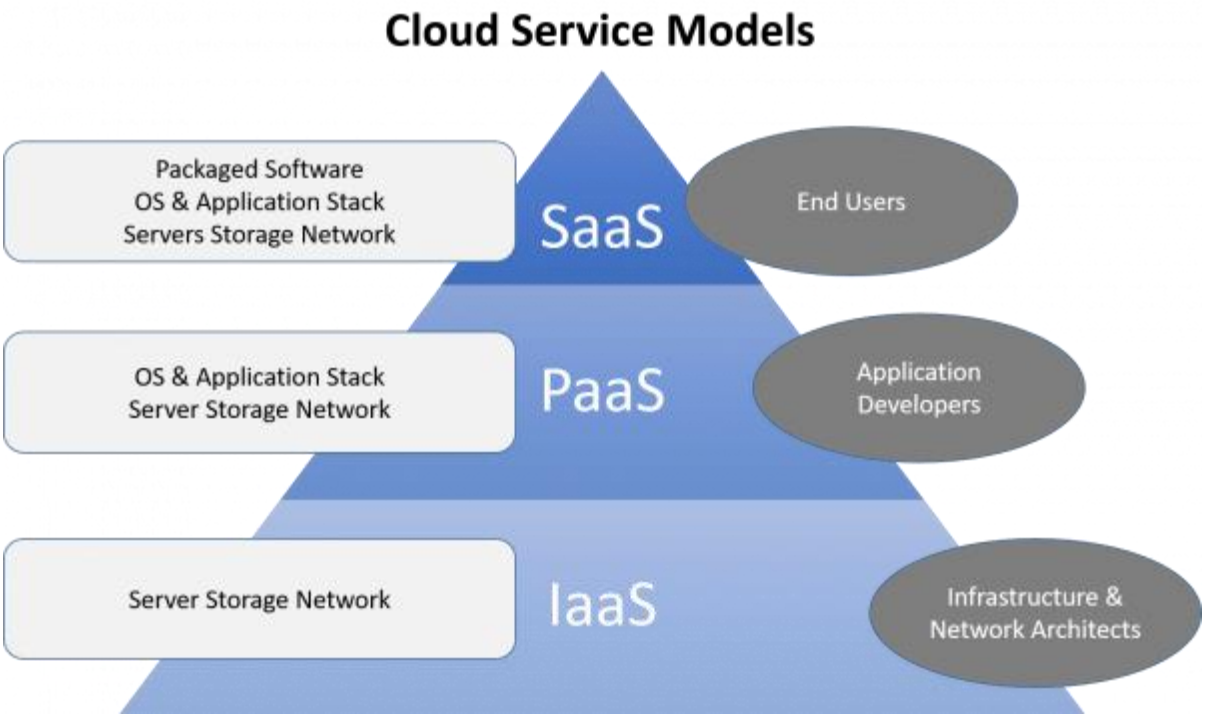


Figure 1.4 : Modèles de service en nuage [9].

1.5 Avantages et inconvénients

Le Cloud computing présente de nombreux avantages pour les entreprises tributaires à des contraintes de rentabilité et de performances de plus en plus fortes, les utilisateurs sont dans l'obligation de chercher de nouvelles solutions informatiques et se retournent vers des outils qui permettent un gain de temps et une réduction des coûts.

1.5.1 Avantages

Parmi les avantages du Cloud computing on peut citer :

- Le Cloud computing facilite le travail nomade et à distance. Cette technologie simplifie les usages et répond aux impératifs d'un nouveau modèle d'entreprise plus mobile.
- Une technologie flexible et évolutive quand les entreprises évoluent et qu'elles ont besoin davantage de bande passante, suite à un trafic plus important sur leurs sites le service Cloud s'adapte immédiatement en fournissant les ressources supplémentaires.

- Une réduction de coût. Grâce au Cloud computing, il n'est plus nécessaire d'acquérir un matériel hardware très coûteux et de gérer soi même ses propres serveurs. Donc cette solution permet de réaliser des économies appréciables.
- L'accès à des logiciels onéreux. Un des aspects positifs du Cloud est la facturation à l'usage. Les entreprises peuvent louer des logiciels et applications qu'elles n'auraient pas la possibilité d'acquérir en raison du prix élevé des licences.
- Une parfaite adaptation aux besoins des entreprises. Les outils collaboratifs et les logiciels de gestion intégrés disponibles en SaaS, sont conçus pour répondre aux exigences de chaque secteur d'activité.

1.5.2 Inconvénients

Parmi les inconvénients du Cloud computing on peut citer :

- Si la connexion internet est perdue, l'accès au Cloud n'est plus possible.
- Plusieurs entreprises n'observent que les frais de stockage mais il faut également prendre en compte les frais de transfert qui peuvent être importants selon l'utilisateur du Cloud par l'entreprise.
- Il faut vérifier que les conditions de service sont conformes aux exigences de l'entreprise.
- Malgré une connexion Internet rapide avec un débit garanti, certains applications web peuvent s'avérer être très lentes.
- L'utilisateur n'a pas de contrôle sur les versions des systèmes et des logiciels déployés.
- Le piratage d'un compte d'entreprise pourrait avoir des conséquences désastreuses pour la réputation de l'entreprise.

1.6 Conclusion :

A travers ce chapitre, nous avons présenté les concepts fondamentaux à connaitre pour comprendre la technologie du Cloud computing en mettant en relief son infrastructure. Nous avons commencée par la présentation du Cloud computing en examinant quelques concepts de base, ensuite nous avons présentée les modèles de déploiement et l'architecture de base du Cloud computing. Le chapitre s'est terminé par introduire les avantages et les inconvénients du Cloud.

Nous avons également mis en relief dans ce chapitre toute l'importance que revêt le Cloud dans la vie des entreprises. Cette technologie permet à tous les secteurs d'éviter les investissements sur l'informatique à l'exception de l'achat des ordinateurs de travail pour les employés.

Le chapitre suivant s'intéresse à la présentation des métaheuristiques.

CHAPITRE 2 :

Les métaheuristiques

2.1 Introduction

La plupart des études liées au processus d'amélioration à l'heure actuelle se concentrent sur la résolution de divers problèmes qui peuvent être discrets ou à variables continues, avec un seul ou plusieurs objectifs (optimisation mono-objectif ou multi-objectif), avec ou sans contraintes.

Le problème d'optimisation peut être défini par un ensemble de variables, une fonction objectif (ou fonction de coût) et un ensemble de contraintes, l'espace de recherche est l'ensemble des solutions possibles du problème. Il possède une dimension pour chaque variable, pour des raisons pratiques et de temps de calcul, l'espace de recherche des méthodes de résolution est en général fini. Cette dernière limitation n'est pas gênante, puisqu'en général le décideur précise exactement le domaine de définition de chaque variable. La fonction objectif définit le but à atteindre, on cherche à minimiser ou à maximiser celle-ci, l'ensemble des contraintes est en général un ensemble d'égalités et d'inégalités que les variables doivent satisfaire. Ces contraintes limitent l'espace de recherche [12].

En effet nous avons assisté ces dernières années à une croissance très rapide des travaux utilisant les méthodes d'optimisation, cette tendance peut être observée dans tous les domaines de la science.

Nous aborderons dans ce chapitre les méthodes exactes et les méthodes approchées qui seront introduites dans ce chapitre en particulier les métaheuristiques, et par la suite nous présenteront l'un de ces algorithmes, il s'agit de l'algorithme d'optimisation du loup gris (Grey Wolf Optimisation), cette algorithme inspiré du comportement de vivre en groupe chez les loups gris.

2.2 Méthodes de résolution

Les méthodes possibles pour trouver des solutions à divers problèmes se répartissent en deux catégories principales : la classe de méthodes exactes et la classe des méthodes approchées. Voir la figure 2.1.

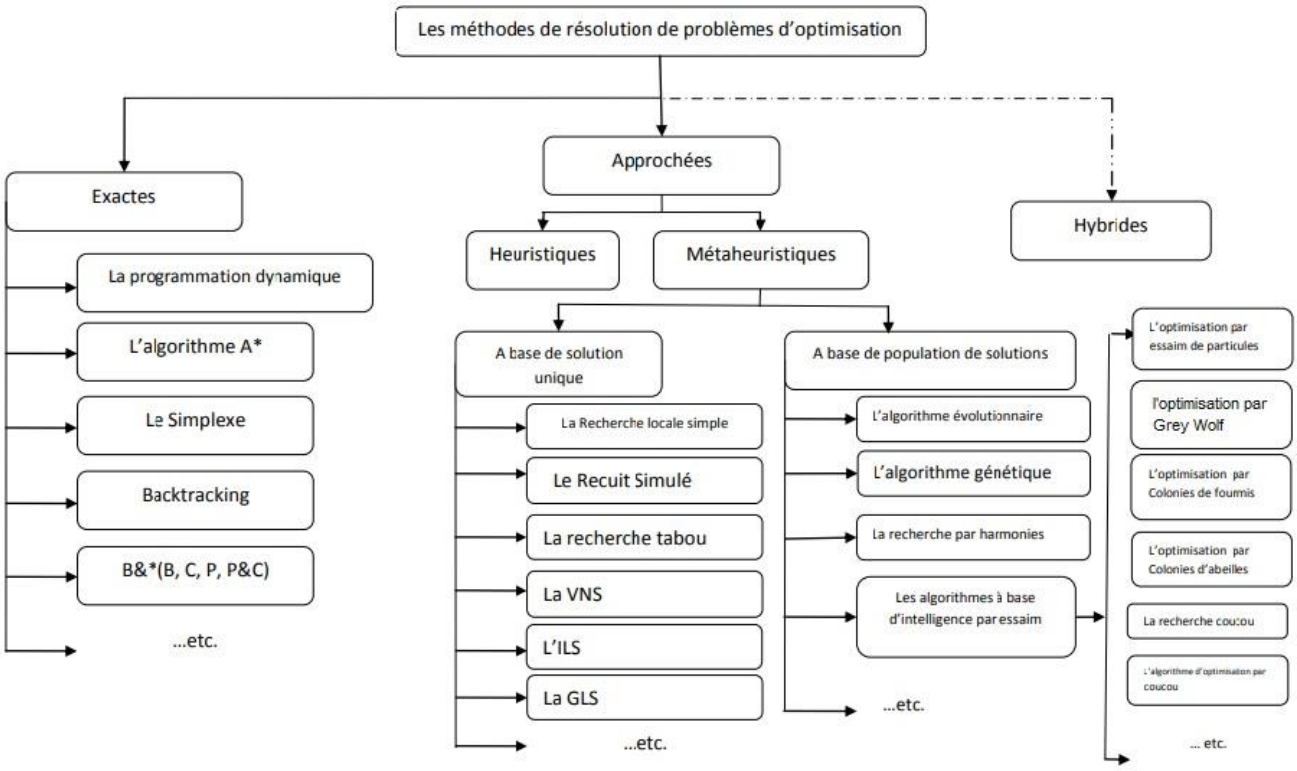


Figure 2.1 : Classification des méthodes de résolution de problème d'optimisation [13].

2.2.1 Méthodes exactes

En fait, des méthodes précises peuvent nous aider à atteindre la solution optimale au problème qui nous est posé, en faisant passer tout l'ensemble de l'espace de recherche avec une méthodologie qui nous amène à obtenir toutes les solutions, qui peuvent être meilleures que la solution optimale obtenue précédemment.

Ainsi, les méthodes exactes sont sues par le fait qu'elle nécessite des coûts de recherche, qui sont généralement onéreux en termes de ressources exigées. En effet, le temps de recherche et l'espace mémoire nécessaire pour l'obtention de la solution optimale par une méthode exacte sont souvent trop grands, notamment avec des problèmes de grandes tailles. Pour cette raison, cette classe d'algorithmes devient de plus en plus complexe avec la taille de la requête à traiter, elle évolue d'une manière importante face à des problèmes qui incluent de nombreuses variables [13].

2.2.2 Méthodes approchées

Quand on possède un temps de calcul limité ou quand on rencontre des problèmes difficiles ou de taille importante, on peut utiliser les méthodes approchées, en se basant

sur une solution de bonne qualité. Dans ce sens, le choix est parfois possible entre deux classes entièrement différentes, une classe heuristique, et une autre métaheuristique [14].

2.2.2.1 Heuristiques

Comme nous le savons, l'heuristique est une méthode ou une technique de guidage, en général elle est empirique ou obtenue par approximation, cela permet la possibilité de choisir la meilleure façon de trouver une solution au problème posé ou d'éliminer les solutions les moins intéressantes, sans qu'on puisse garantir la validité ou la précision de l'information ainsi donnée [15].

Accéder au domaine des heuristiques, c'est renoncer du coup aux schémas classiques. Effectivement, tandis que la démarche classique mathématique est centrée sur l'objet de l'étude, sur la compréhension de sa structure et de sa logique, la méthode heuristique écarte le problème lui-même pour une bonne explication, afin de dégager des schémas de pensée plus généraux et donc originaux.

Dans ce domaine, les heuristiques possèdent une simplicité et donc une rapidité dans leur exécution plus élevée que les algorithmes classiques. Mais une méthode heuristique trop simplifiée ou au contraire trop générale peut conduire à des aspects cognitifs, créant des erreurs de décision.

Les méthodes heuristiques peuvent être classées en deux catégories :

- Des méthodes constructives qui créent des solutions à partir d'une solution initiale en essayant d'en ajouter au fur à mesure des éléments jusqu'à l'obtention complète d'une solution.
- Des méthodes de fouilles locales qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière récurrente permettent d'améliorer cette solution en prospectant son voisinage.

2.2.2.2 Métaheuristiques

Les métaheuristiques ont fait leur apparition grâce aux différents problèmes rencontrés par les méthodes heuristiques pour avoir au préalable une solution réalisable de bonne qualité pour des problèmes d'optimisation difficiles. Ces algorithmes sont plus complets et complexes qu'une simple heuristique, et permettent généralement d'obtenir

une solution de très bonne qualité pour des problèmes nés des domaines de la recherche opérationnelle ou de l'ingénierie dont on n'a pas de méthodes efficaces pour les traiter ou bien quand la solution du problème demande un temps assez conséquent ou une mémoire de stockage plus grande.

La relation entre le temps d'exécution et la qualité de la solution trouvée d'une métaheuristique reste alors dans la plupart des cas très intéressant suite aux différents types d'approches de résolution. La majorité des métaheuristiques utilisent des processus aléatoires et itératifs comme moyens de rassembler de l'information, d'explorer l'espace de recherche et de faire face à divers problèmes. Une métaheuristique peut être utilisée pour différents types de problèmes, alors qu'une heuristique est employée à un seul problème [15].

Beaucoup de métaheuristiques sont souvent inspirées par des systèmes naturels dans plusieurs domaines comme : la physique (recuit simulé), la biologie (algorithmes évolutionnaires et génétiques) et aussi l'éthologie (les algorithmes de colonies de fourmis ou l'optimisation par essaim particulaire). Un des défis de la conception des métaheuristiques est donc de simplifier le choix d'une méthode et le réglage des paramètres pour les adapter à un seul problème [16].

Les méthodes métaheuristiques peuvent être ordonnées en de nombreuses manières. On peut voir celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois. Voir la figure 2.2.

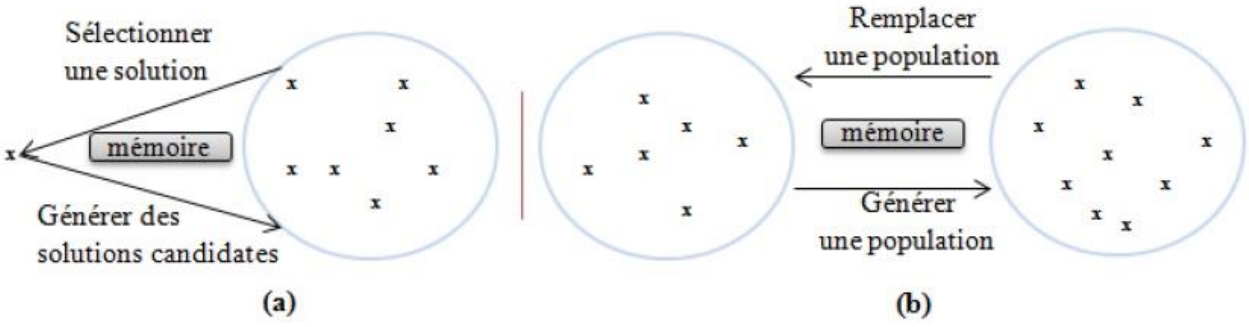


Figure 2.2 : Principes Généraux d'une métaheuristique à base de: (a) solution unique, (b) solution population.

- Les métaheuristiques à base d'une seule solution : débutent avec une seule solution initiale et s'en écartent graduellement, par la construction d'une ligne dans l'espace de recherche. Les méthodes de trajectoire comportent

principalement la méthode de descente, celle du recuit simulé et la recherche tabou.

- Les métaheuristiques à base de population de solutions: contrairement aux algorithmes partant d'une solution particulière, les métaheuristiques à population de solutions rend plus satisfaisant, petit à petit des itérations, une population de solutions. On différencie dans ce groupe, les algorithmes évolutionnaires, qui sont une famille d'algorithmes née de la théorie de l'évolution par la sélection naturelle, et les algorithmes d'intelligence en essaim qui, comme les algorithmes évolutionnaires, émanent de similarité avec des phénomènes biologiques naturels [14].

2.3 Caractéristiques principales des métaheuristiques

- Le but est d'explorer réellement l'espace de recherche afin de trouver des solutions qui tendent vers l'optimalité.
- Les métaheuristique sont des méthodes inspirées de processus naturels.
- Les métaheuristiques ne sont pas représentatives à un problème donné.
- Les métaheuristiques peuvent avoir des mécanismes permettant d'éviter le blocage dans des régions de l'espace de recherche.

Dans ce qui suit, nous présentons quelques algorithmes basés sur l'intelligence en essaim, notamment l'algorithme qui sera utilisé dans la partie réalisation de notre projet de fin d'études.

2.4 Algorithme basés sur l'intelligence en essaim

Ces algorithmes sont des méthodes d'optimisation stochastique basée sur une population de solutions, l'intelligence en essaim a été développée par Kennedy et Eberhart en 1995 [Kennedy, 1985]. Elle est inspirée du comportement social des nuées d'oiseaux, des colonies d'insectes, et bien d'autres sociétés animales évoluant en groupe. Effectivement, on peut noter chez ces groupes d'animaux des dynamiques de déplacement plus ou moins complexes, tandis que chaque individu a une intelligence limitée, et ne possède qu'une connaissance locale de sa situation dans l'essaim [17].

Dans ce contexte, nous présentons quelques algorithmes appartenant à cette catégorie : algorithme de colonies d'abeilles (ABC: Artificial Bee Colony), algorithme de la recherche coucou (CS: Cuckoo Search), algorithme d'optimisation des chats

(CSO : Cat Swarm Optimization) et algorithme d'optimisation des loups gris (GWO : Grey Wolf Optimisation).

Nous donnons plus de détails à ce dernier algorithme car, c'est celui-ci qui sera utilisé dans la partie réalisation de notre projet de fin d'études.

2.4.1 Algorithme de colonies d'abeilles

Cette métaheuristique s'inspire de la vie et du comportement de vraies abeilles. Les abeilles font partie des insectes qui vivent en groupe. Elles ont des comportements qui les distinguent des autres insectes. En tant que colonie, elles construisent leur abri dans des arbres ou dans les montagnes.

Une colonie d'abeilles, est composée en général, de mâles et de femelles. Le nombre de femelles est supérieur à celui des mâles, ces derniers représentent moins de mille qui sont appelées faux-bourdons, et les femelles sont de 10 000 à 80 000 environ. Elles s'appellent ouvrières et elles sont stériles.

La colonie est dirigée par une reine qui est responsable de la reproduction par le processus d'accouplement avec les faux-bourdons [16].

2.4.2 Algorithme de la recherche coucou

Cette métaheuristique est inspirée du comportement des coucous dans leur environnement naturel. Dans l'algorithme (CS: Cuckoo Search), une solution donnée est appelée habitat. Un habitat représente une position actuelle du coucou dans l'espace de recherche.

Chaque habitat du coucou produit un certain nombre d'œufs, une partie se transforme en coucou et d'autres se gâchent. Les œufs se forment en fonction de variable d'habitat actuel du coucou et la variable de distance de ponte proportionnelle au nombre total d'œufs. En plus d'autres variables qui déterminent la période à partir de laquelle les valeurs des variables sont extraites [13].

Chaque coucou pond un ensemble d'œufs à chaque itération, une parte d'eux survivent et vont dans le but de rechercher le meilleur habitat. Les autres seront tués par leurs hôtes ou supprimée à cause de leurs mauvaises qualités.

2.4.3 Algorithme d'optimisation des chats

L'algorithme d'optimisation des chats (CSO) est basé sur la considération que chaque chat est une solution dans l'essaim. Dans cet algorithme, il existe deux méthodes : le suivi et la recherche, où les chats sont sélectionnés au hasard dans l'essaim, avec leurs drapeaux placés sur la recherche ou le mode de suivi, et ce processus dépend également du rapport de mélange (MR) dans cet algorithme.

Ces deux méthodes d'opération mathématique ont été conçues afin de trouver des solutions à un problème d'optimisation difficile :

- Mode de recherche (Seeking Mode): dans ce mode, le chat est modélisé au repos, et le chat reste vigilant sur son environnement pour son prochain mouvement.
- Mode de traçage (Tracing Mode) : dans ce mode, on trouve la compatibilité avec la technologie de recherche locale du problème d'optimisation, dans laquelle le chat poursuit la cible en fonction de sa propre vitesse. L'opération de traçage est très similaire au mouvement de particules dans l'optimisation par essaim de particules, en suivant le modèle, où chaque chat a un vecteur de position (X) et un vecteur de vitesse (V) pendant le processus de recherche [18].

2.4.4 Algorithme d'optimisation des loups gris

L'algorithme d'optimisation des loups gris (Grey Wolf Optimisation GWO) est considéré comme une technique intelligente de l'essaim conçue par Mirjalili et Andrew Lewis en 2014. Il simule la hiérarchisation de meneur des loups qui sont bien connus pour leur chasse en groupe [19].

Dans cette forme d'optimisation, l'algorithme simule le chef social du comportement de chasse des loups gris dans la nature. Il faut savoir que la population de cet algorithme est divisée en quatre groupes : alpha (α), bêta (β), delta (δ) et oméga (ω). Les trois premiers loups plus forts sont considérés comme α, β et δ qui dirigent les autres loups (ω) vers des zones encourageantes de l'espace de recherche. Voir la figure 2.3.

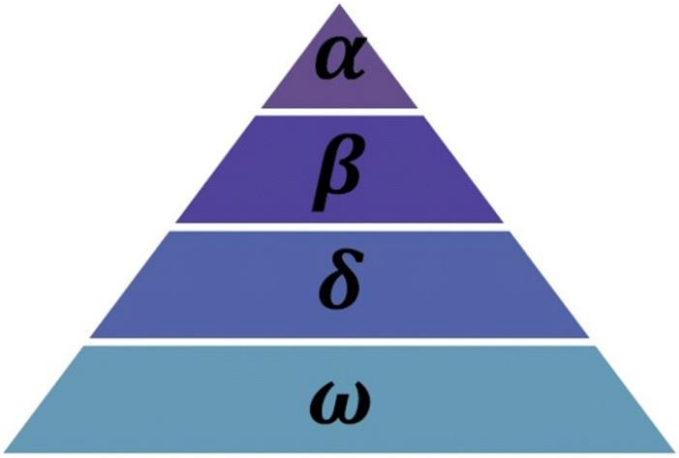


Figure 2.3 : Hiérarchie du loup gris (la dominance diminue de haut en bas) [20].

2.4.4.1 Caractéristique du loup gris

Les loups gris sont des animaux qui entrent dans la classe des prédateurs. Cette catégorie de loups se trouve généralement en amont de la chaîne alimentaire. Ces derniers, vivent dans leur état initial en groupe. En général, le nombre du groupe est composé en moyenne de 5-12 individus. La particularité de ces loups est qu'ils vivent dans une hiérarchie sociale dominante très rigide [19].

- Dans la meute des loups on trouve le premier niveau qui s'appelle alphas(α). Ces loups (alpha) sont les leaders du groupe composé d'un mâle et femelle. Ils ont essentiellement la capacité de prendre la décision sur la chasse, l'endroit de répit, le moment de réveil. Les décisions de l'alpha sont données à la meute. Il a été constaté aussi, un genre d'attitude démocratique, dans lequel un loup alpha suit les autres membres du groupe. Lors de leurs rassemblements, la majorité du groupe reconnaît le loup l'alpha dont sa queue est pendante vers le bas. Il faut savoir que dans le groupe, le loup alpha est considéré comme loup dominant du moment que ses instructions devraient être respectées par la meute. Dans le groupe des loups gris, l'accouplement est uniquement autorisé entre les loups alpha. Dans la vie des loups, alpha n'est toujours pas le membre le plus puissant dans le groupe mais le meilleur dans la gestion quotidienne du pack. La particularité du groupe repose beaucoup plus sur son organisation et sa discipline que sur sa force.
- Le deuxième niveau est appelé bêta (β). Les bêtas représentent les loups dépendants qui assistent l'alpha dans la prise de décision et dans les mouvements et activités quotidiens du pack. Dans le cas ou l'un des loups alpha meurt ou

devient plus âgé, le loup bêta reste très probablement le candidat le plus apte pour remplacer le loup alpha décédé ou devenu plus âgé. Le loup bêta a deux rôles à jouer dans le groupe, d'une part il doit respecter l'alpha, d'autre part, il doit aussi commander l'autre niveau inférieur des loups. Il joue la mission de conseiller de l'alpha et de la discipline pour le groupe. La version bêta solidifie les instructions de l'alpha pendant toute la durée du pack et donne des observations à l'alpha.

- Le troisième niveau se nomme delta (δ). Les loups gris sont composés de loups alpha, de loups beta de loups oméga et de loups delta. Ces derniers doivent se soumettre aux alphas et aux bêtas, mais leur domination est visible sur les loups oméga (le niveau le plus inférieur). Cette catégorie est formée des anciens, des chasseurs, et des gardiens en plus des sentinelles et des scouts. Ces derniers ont la responsabilité de surveiller les limites du territoire et peuvent avertir la meute en cas de danger. Alors que les sentinelles sont tenues de protéger le groupe et lui garantir sa sécurité. Les chasseurs viennent en aide aux alphas et aux betas quand ils sont en période de chasse. Les anciens loups qui ont une grande expérience deviennent loups bêta.
- Le quatrième niveau le plus bas s'appelle oméga(ω). Dans ce niveau. Dans leur nature, les loups oméga ont une soumission totale envers les autres loups dominants. Ces deniers n'ont le droit à manger qu'après le reste du groupe. Il faut souligner que l'oméga parait ne pas être un individu essentiel dans la meute, mais selon des observations, l'ensemble du groupe trouve des difficultés en cas d'absence ou de perte du loup oméga. En cas de perte de celui-ci, l'indiscipline et la violence s'établissent entre les membres du groupe. Cela prouve que la présence des loups oméga est très importante dans la vie de l'ensemble de la meute. Alors on peut dire qu'oméga est choisi pour endosser une responsabilité.

2.4.4.2 Description de l'algorithme

Les étapes générales de l'algorithme GWO sont les suivantes :

Initialiser la population X_i des loups gris (i=1,2,…..,n).

Initialiser a, A et C.

Calculer la valeur de la fonction objective correspondante pour chaque loup.

Choisir les trois premiers meilleurs loups et sauvegarde-les sous α, β et δ.

Tant que (t < Max_itération) faire

pour chaque loup

Actualiser la position courant en utilisant l'équations **2.8**

fin pour

Maitre à jour des paramètres a, A et C.

Calculer la valeur de la fonction objectif correspondante pour chaque loup.

Maitre à jour α, β et δ.

t = t + 1.

Fin tant que

Renvoie la position de α.

Dans ce qui suit, nous présentons la modélisation mathématique de la hiérarchie sociale des loups gris et leur technique de chasse utilisée (poursuite, encerclement et attaque de proies).

- **Hiérarchie sociale**

  Pour la modélisation mathématique de la hiérarchie sociale des loups pendant la création de GWO, nous estimons que l'alpha (α) est la solution la plus apte. En conséquence, les meilleures solutions sont effectivement bêta (β) et delta (δ), respectivement deuxième et troisième.

  Les autres solutions postulantes sont censées être des omégas (ω). Nous signalons que dans l'algorithme GWO, l'optimisation est dirigée par α, β et δ, tandis que les autres membres du groupe suivent ces trois loups [19].

- **Encercler les proies**

GWO modélise la démarche des loups, tout en assimilant la recherche du minimum d'une fonction coût à la stratégie de chasse d'un groupe de loups. La poursuite de la proie est équivalente à l'investigation ou à la recherche globale, qui est représentée par les loups gris entourant leurs proies durant la période de chasse, pour la modélisation mathématique du comportement d'encerclement des équations qui sont proposées :

$$\vec{D} = |\vec{C} \times \overrightarrow{X_p}(t) - \vec{X}(t)| \qquad (2.1)$$

$$\vec{X}(t+1) = \overrightarrow{X_P}(t) - \vec{A}\,\vec{D} \qquad (2.2)$$

Où t indique l'itération en cours, $\vec{A}$ et $\vec{C}$ sont des vecteurs de coefficients, $\overrightarrow{X_P}$ est le vecteur de position de la proie, $\vec{X}$ est le vecteur de position d'un loup gris.

Les vecteurs $\vec{A}$ et $\vec{C}$ sont calculés comme suit:

$$\vec{A} = 2\vec{a} \times \overrightarrow{r_1} - \vec{a} \qquad (2.3)$$

$$\vec{C} = 2 \times \overrightarrow{r_2} \qquad (2.4)$$

$\vec{a}$ est diminué linéairement de 2 à 0, et **r1**, **r2** sont des vecteurs aléatoires dans [0,1].

- **La chasse**

La meute des loups gris possède la faculté de reconnaître l'endroit des proies et les encercler. En général le loup alpha guide la chasse, de temps à autre, les loups bêta et delta peuvent prendre part à la chasse. Par ailleurs, nous ne pouvons avoir aucune idée de l'emplacement de l'optimum (proie) dans un espace de recherche abstraite. Pour toute simulation mathématique du comportement de chasse des loups gris, nous supposons que l'alpha, beta et delta ont une meilleure connaissance à propos de la position éventuelle de la proie. Ainsi, les trois premières meilleures solutions obtenues ont été sauvées jusqu'à maintenant et les autres loups (y compris les omégas) sont obligés de mettre à jour leurs positions selon l'emplacement des meilleurs agents de recherche. (Voir la figure 2.4), comme indiqué dans les équations suivantes :

$$\overrightarrow{D_\alpha} = |\ \overrightarrow{C_1} \times \overrightarrow{X_\alpha} - \overrightarrow{X}\ | \qquad (2.5)$$

$$\overrightarrow{D_\beta} = |\ \overrightarrow{C_2} \times \overrightarrow{X_\beta} - \overrightarrow{X}\ | \qquad (2.6)$$

$$\overrightarrow{\mathbf{D_\delta}} = | \overrightarrow{\mathbf{C_3}} \times \overrightarrow{\mathbf{X_\delta}} - \overrightarrow{X} | \quad \textbf{(2.7)}$$

$$\overrightarrow{\mathbf{X_{(t+1)}}} = \frac{\overrightarrow{\mathbf{X_1}} + \overrightarrow{\mathbf{X_2}} + \overrightarrow{\mathbf{X_3}}}{3} \quad \textbf{(2.8)}$$

Tel que :

- X_α représente la position de alpha.
- X_β indique le La position de bêta.
- X_δ est la position de delta.
- $\overrightarrow{C_1}$, $\overrightarrow{C_2}$, $\overrightarrow{C_3}$ sont des vecteurs aléatoires et $\overrightarrow{X}$ indique la position de la solution.

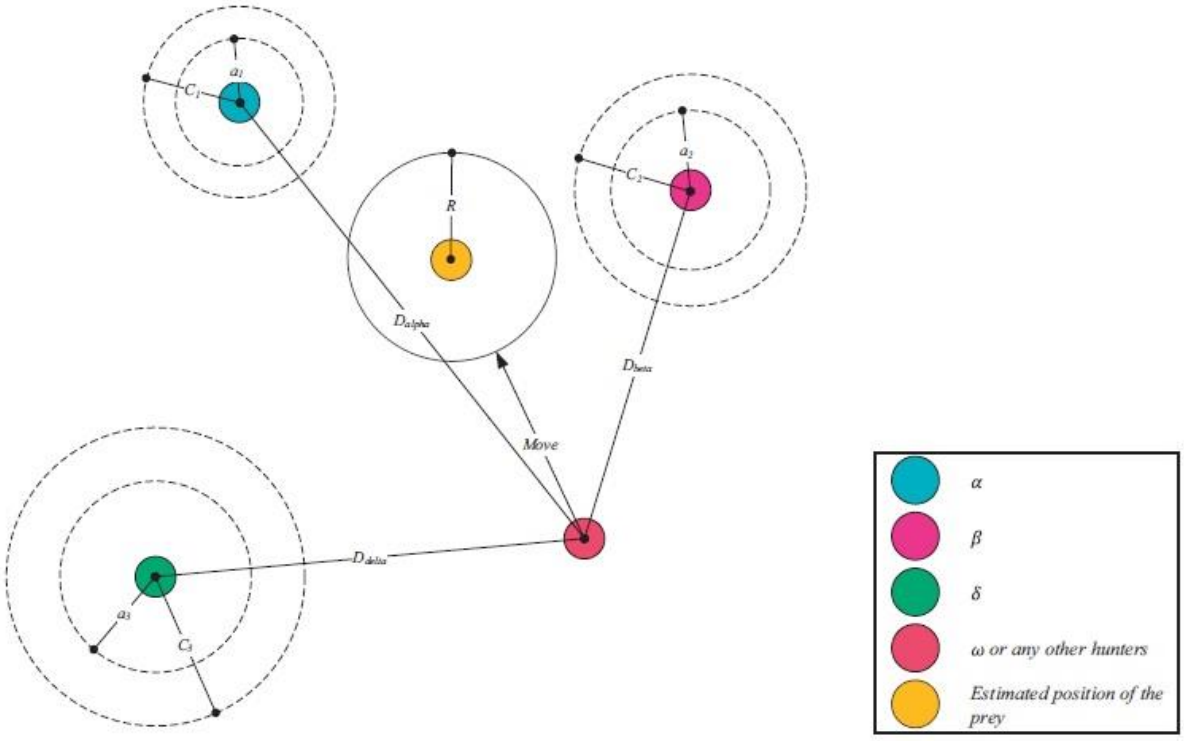


Figure 2.4 : La mise à jour de la position dans GWO [19].

- **Attaquer la proie**

Comme il a été indiqué plus haut, les loups gris achèvent la chasse en attaquant la proie quand elle est immobilisée et afin de procéder à la modélisation mathématique de l'approche de la proie, nous effectuons à la diminution de la valeur de $\overrightarrow{a}$. La plage de fluctuation de $\overrightarrow{A}$ est également diminuée. Autrement dit, $\overrightarrow{A}$ est une valeur aléatoire dans l'intervalle [-2a, 2a] où $\overrightarrow{a}$ est diminué de 2 à 0 au cours des itérations, quand les valeurs aléatoires de $\overrightarrow{A}$ sont dans [-1,1],

l'emplacement suivant d'un individu de recherche peut se trouver dans n'importe quelle position entre cette dernière et la position de la proie, une valeur de |A| < 1 force les loups à attaquer la proie.

Suite à ces opérateurs proposés jusqu'à présent, l'algorithme GWO autorise ses individus de recherche pour mettre à jour leur position en fonction de l'emplacement de l'alpha, bêta et delta, et attaque la proie. Ainsi l'algorithme GWO est sujet à l'inertie dans les solutions locales avec ces opérateurs. Il est clair que le mécanisme d'encerclement présenté indique l'exploration dans une certaine mesure, mais GWO a besoin de plus d'opérateurs.

Alors que les sept (07) emplacements possibles ont été représentés sur la figure 2.5, les données aléatoires $\overrightarrow{A}$ et $\overrightarrow{C}$ permettent aux loups de se déplacer dans toutes les positions dans l'espace continu autour de la proie [19].

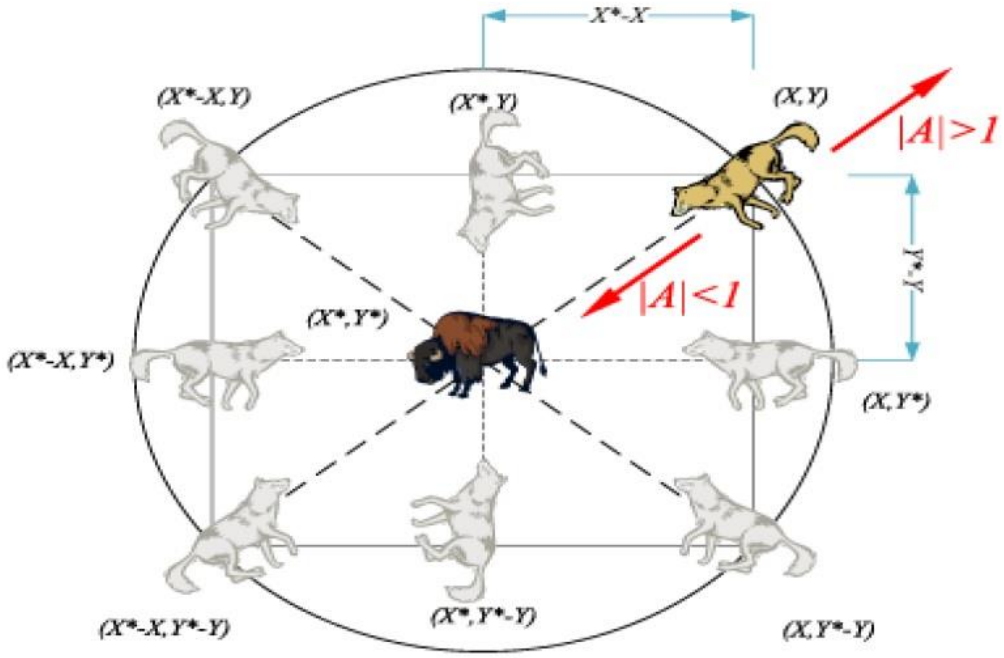


Figure 2.5 : Mécanisme de positionnement de l'agent de recherche et de l'effet de ce qui le présent [21].

2.5 Conclusion

Dans ce chapitre, nous avons présenté les méthodes de résolution de problèmes, à savoir les méthodes exactes et les méthodes approchées en se concentrant sur les métaheuristiques. Une attention particulière a été donnée à la métaheuristique GWO ou l'algorithme des loups gris en présentant ses caractéristiques ainsi que sa description.

Dans le cadre de notre PFE, nous nous intéressons dans le chapitre suivant, à l'application de l'algorithme GWO pour l'ordonnancement de tâches multi-objectives dans le domaine du Cloud computing.

CHAPITRE 3 :

Implémentation de l’application et évaluations des résultats

3.1 Introduction

L'objectif de ce chapitre est de présenter notre contribution dans le cadre de ce PFE. Cette contribution se base sur l'application de l'algorithme Grey Wolf afin de faire l'ordonnancement des tâches multi-objectif dans le Cloud computing par rapport à deux critères qui sont : le *makespan* et le *coût*. Le chapitre présente une évaluation multi-objective de l'algorithme Grey Wolf pour ces deux critères.

Nous présentons également les outils de simulations utilisés dans le cadre de ce PFE, à savoir, le langage Java, l'IDE Netbeans, l'outil Jfreechart et le simulateur CloudSim.

Nous nous intéressons en particulier à l'architecture de ce simulateur, ses classes fondamentales ainsi que les classes que nous avons incorporées pondant la réalisation de notre projet.

Nous présentons aussi les principes de base de l'optimisation multi-objective, notamment, quelques méthodes pour mesurer la qualité d'un ensemble de solutions non-dominées. Nous présentons également l'IHM que nous avons développée dans le cadre de ce PFE ainsi une étude comparative par rapport aux résultats obtenus.

3.2 Environnement de développement et de simulation

L'environnement de développement et de simulation est composé des éléments suivants :

3.2.1 JAVA

Java est un langage de programmation orienté objet dont la syntaxe est similaire au langage C. Ce langage est beaucoup plus utilisé pour le développement d'applications des sociétés et mobiles [22].

Nous mettons en relief quelques chiffres et faits relatifs au langage Java en 2020 :

- 40% des machines d'entreprises ont une JVM installée.
- Il y a plus de 9.4 millions de développeurs Java dans le monde.

Nous savons que java est un langage compilé et interprété puisque le compilateur java produit un code intermédiaire (Byte-code) qui sera interprété par une JVM. Celle-ci assure la portabilité du code qui a permis la réussite de java.

3.2.2 Netbeans

Netbeans est un environnement de développement intégré open source qui est développé par Oracle et disponible en 23 langues et en plusieurs versions qui sont adaptées aux environnements Windows, Linux, Solaris et MacOS.

Il nous permet de développer des applications dans plusieurs langages dont Java, C, C++, Javascript, PHP, HTML, Python et Ruby [23].

3.2.3 Jfreechart

JFreeChart est une API Java permet aux développeurs de créer des graphiques et des diagrammes de très bonne qualité professionnelle dans leurs applications. Cette API est open source et sous licence LGPL (Lesser General Public Licence).

3.2.4 CloudSim

Plusieurs outils de simulation de systèmes distribués existent dans la littérature. Les plus connus sont GridSim [24], CloudSim, Simgrid [25].

Dans ce qui suit, on va présenter le Framework CloudSim, qui permet la simulation de l'environnement du Cloud computing.

La structure de CloudSim est composée de différentes couches ainsi que ses éléments architecturaux. Le moteur de simulation aux événements discrets SimJava se trouve au niveau le plus bas, qui implémente les fonctionnalités de base nécessaires pour le traitement des événements, la communication entre les composants, la gestion de l'horloge, les files d'attente, la création de composants du système (services, hôte, Datacenter, Broker, les machines virtuelles).

CloudSim supporte la modélisation et la simulation de l'environnement de Datacenter basé sur Cloud, comme par exemple la mémoire, les interfaces de gestion dédiées aux VMs ainsi que le stockage et la bande passante. L'instanciation et l'exécution des entités de base (VM, hôtes, Datacenters, applications) sont gérées par la couche CloudSim pendant l'étape de simulation. Nous trouvons dans la couche supérieure de la pile de simulation le code de l'utilisateur qui expose les politiques d'ordonnancement de Broker, VM, applications (nombre d'utilisateurs, nombre de tâches) ainsi que la configuration des fonctionnalités liées aux hôtes [24]. Voir la figure 3.1.

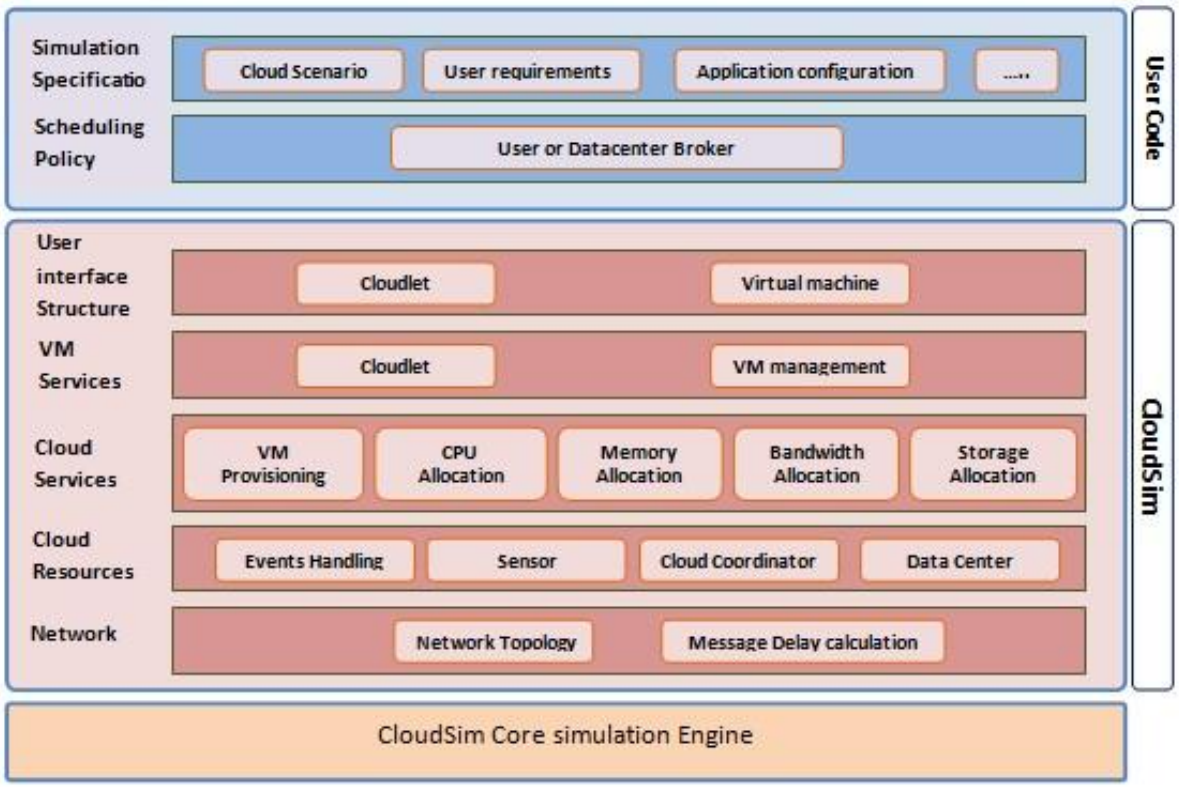


Figure 3.1 : Architecture de CloudSim [26].

Ce simulateur définit deux politiques de planification des tâches:

- Space shared **:** une seule tâche s'exécute sur la *VM* jusqu'à la fin et après sa terminaison, une autre tâche sera lancée sur cette *VM*.
- Time shared **:** plus d'une tâche peut être exécutée sur la même *VM* dans une tranche de temps différente.

3.2.5 Modélisations des machines virtuelles et des cloudlets

La couche de virtualisation gère l'exécution des *VMs* sur les machines physiques et la manière dont les cloudlets sont exécutés. En effet, CloudSim permet deux formes d'exécution : le temps partagé et l'espace partagé pour l'exécution *VMs* sur les machines physiques. L'instanciation de brokers gère aussi l'exécution des *VMs*. La fin du cycle de vie d'un brocker est liée a la cessation de vie de chacune des *VMs* qu'il contient. Un brocker n'achève son exécution qu'au moment où les *VMs* contenues dans celui-ci finissent leur exécution [27].

3.3 Techniques d'optimisation multicritères

L'optimisation multi-objective simplifie la recherche des valeurs de variables qui permettent de maximiser ou de minimiser une ou plusieurs fonctions objectifs. Elle est conduite avant tout par la définition des critères de qualité de la solution du problème, ensuite, par l'algorithme d'optimisation qui va solutionner le problème en se basant sur la recherche de la meilleure solution en fonction de ces critères mentionnés.

L'optimisation multi-objective permet de définir des problèmes incorporant plusieurs critères qui sont parfois opposés. Dans ce cadre, la solution optimale que nous recherchons n'est nullement un simple point, mais au contraire une somme pondérées qui puissent satisfaire l'ensemble des contraintes [28].

Dans ce PFE, nous nous sommes intéressés aux méthodes de prise de décision multicritères (Multi-Criteria Decision Making ou MCDM). Ces méthodes nous aident à faire les meilleurs choix dans le cas où plusieurs critères sont présents [29].

Il existe plusieurs méthodes multicritères dont les plus répandues sont : la somme pondérée (Weight Sum Method), la méthode de produit pondéré (Weighted Product Model), le processus de hiérarchie analytique (Analytic Hierarchie Process) et ELECTRE (Outranking Method).

Afin de donner d'amples explications, nous allons détailler dans ce qui suit la méthode de la somme pondérée (WSM) qui a été utilisée dans le cadre de ce PFE.

Il s'agit d'une méthode de prise de décision multicritère dans laquelle il y aura plusieurs choix et nous avons l'obligation de déterminer le meilleur choix en fonction de plusieurs critères.

La démarche essentielle que nous retenons de cette méthode, repose sur la fusion des problèmes de manière qu'il ne reste en fin de compte qu'une seule valeur à optimiser. Autrement dit, cette approche permet de réécrire plusieurs objectifs d'un problème multi-objectif en un problème d'optimisation mono-objective en faisant en sorte de multiplier par une valeur de poids (w_i) prédéfinie relativement pour chaque fonction objectif pour que les optimiseurs aient une concentration sur certains objectifs.

Ainsi, à chaque fois que le problème et l'importance de chaque critère soit défini, une seule solution est générée comme le montre l'équation suivante :

$$f(x) = \sum_{n=1}^{n} w_i f_i \qquad (3.1)$$

Les poids $w_1, w_2, \ldots, w_n$ doivent satisfaire les deux contraintes suivantes :

$$\sum_{n=1}^{n} w_i = 1 , \qquad 0 \leq w_i \leq 1 \qquad (3.2)$$

Normalisation de tous les f_i afin de conserver la proportionnalité entre les valeurs :

$$\overline{f_i} = \frac{f_i}{\sum_{i=1}^{n} f_i} \qquad (3.3)$$

3.4 Approche et démarche proposées

Dans ce qui suit, nous faisons une description de notre approche utilisée dans le cadre de ce PFE. Dans ce contexte, l'objectif de notre ordonnancement consiste à affecter des tâches aux *VMs* de façon à optimiser deux métriques de QoS à savoir : le makespan et le coût.

Un ordonnancement multi-objectif en fonction de deux critères en se basant sur l'algorithme GWO.

3.4.1 Evaluation multi-objective

Comme nous l'avons souligné précédemment, nous faisons une évaluons des deux critères de QoS qui sont : le makespan et le coût.

Makespan : il est le moyen le plus utilisé pour évaluer les algorithmes d'ordonnancement. Le makespan incarne la différence entre le temps de soumission de la première tâche et le temps de réception des résultats. En d'autres termes c'est le temps de fin d'exécution de la dernière tâche.

Coût : la majorité des prestataires de Cloud propose des prix pour bénéficier de leurs services. En ce qui concerne notre projet, le coût global de l'exécution correspond à la somme des coûts liés à l'utilisation des *VMs* pour toutes les tâches.

Pour connaître le coût d'exécution d'une tâche donnée, il faut avoir à la foi la durée d'exécution sur la *VM* et le prix unitaire de l'exécution des tâches sur la dite machine.

Le makespan et le coût sont deux métriques classés aux catégories à minimiser. Pour une optimisation multi-objective, la fonction objective résultante est définie comme suit :

$$\boldsymbol{F = \ w_1 * \ \text{makespan} + \ \ w_2 \ * \ \text{coût}} \qquad (\mathbf{3.4})$$

Où $w = \{w_1, w_2\}$ représente le vecteur des poids qui traduit l'importance ou l'exigence de l'utilisateur par rapport à chaque critère. Dans ce PFE, nous avons utilisé trois vecteurs différents de poids {0.8, 0.2}, {0.5, 0.5} et {0.2, 0.8} qui correspondent respectivement aux deux critères {makespan, coût}. Le premier vecteur favorise le makespan, le second vecteur donne le même poids pour les deux critères et le troisième vecteur favorise le coût.

3.4.2 Adaptation de la métaheuristique GWO

Le pseudo code de la métaheuristique GWO qui a été adapté dans le cadre de ce PFE pour l'affectation des tâches dans les différentes ***VMs*** est donné comme suit :

Initialiser la population initiale d'une manière aléatoire.

Trouver les trois meilleurs individus (ordonnancements) I_α, I_β, I_δ.

Tant que critère d'arrêt (nombre d'itération) **faire**

Définir une probabilité de commutation (***a***) diminué linéairement ϵ [2 à 0].

Pour chaque individu de la population **faire**

Générer un nouvel individu selon l'équation (**2.8)**

Evaluer le nouvel individu

Si le nouvel individu est meilleur, le mettre à jour dans la population

Fin pour

Evaluer à nouveau la population

Mettre à jour les trois meilleurs individus I_α, I_β, I_δ.

Fin tant que

Retourner I_α

Il faut souligner qu'une phase de pré-simulation (en utilisant du GWO adapté), est essentielle affin de trouver les meilleures affectations des cloudlets aux *VMs* avant de les transmettre au broker.

3.5 IHM développée

CloudSim utilise seulement en le mode console, il n'a pas d'interface graphique, donc le cadre de ce PFE, nous avons développée notre IHM pour faciliter l'accès et la manipulation de la simulation.

Aussi, devrons tout d'abord passer par l'étape de configuration des différents paramètres de la simulation, afin de vérifier le comportement de notre algorithme.

Les interfaces suivantes représentent les différentes étapes pour paramétrer la simulation.

Interface principale :

Dès le lancement de notre l'application, la fenêtre d'accueil clarifiée dans la figure 3.2 contient les informations de notre PFE.

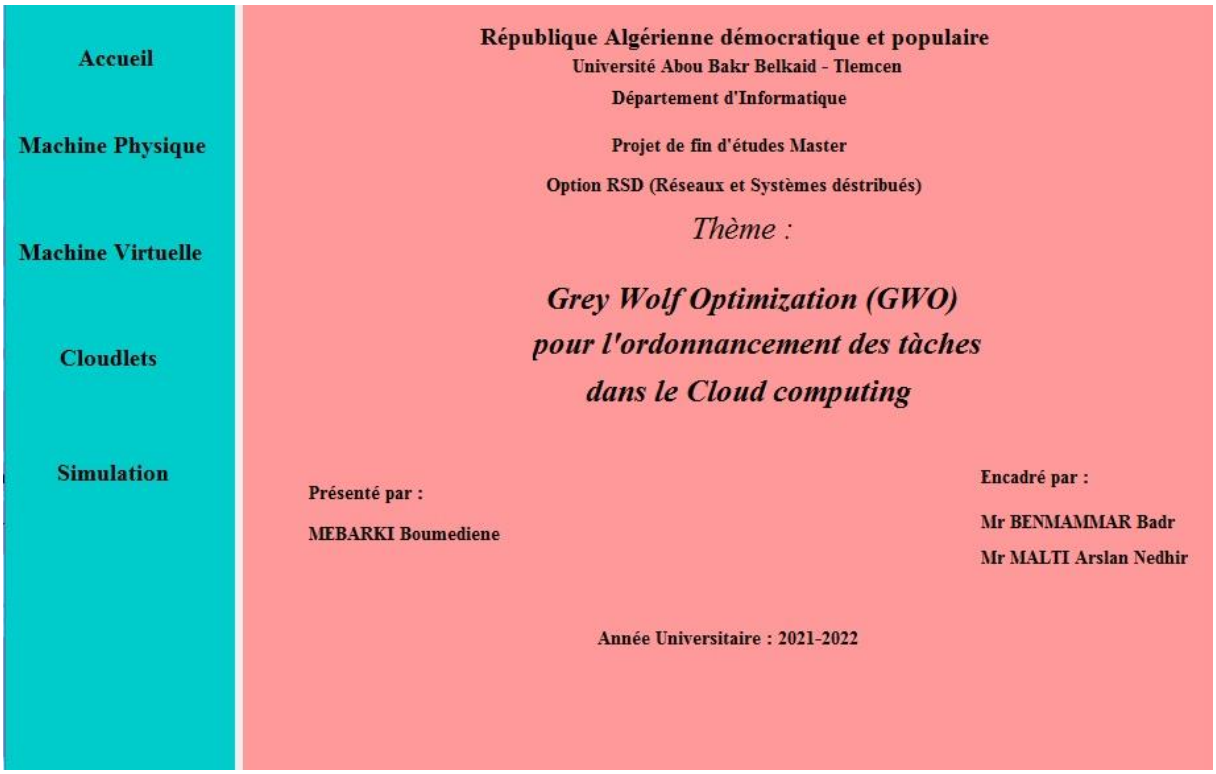


Figure 3.2 : Interface principale.

3.6 Configuration utilisée

Pour l'évaluation de notre algorithme, il est nécessaire d'abord de passer par l'étape de configuration pour déterminer les paramètres de simulation.

3.6.1 Configuration des machines physiques

La configuration de l'environnement Cloud passe par la saisie des paramètres nécessaires aux machines physiques comme le nombre des DataCenters, le nombre des hôtes qui se trouvent dans chaque DataCenter, le nombre de processeur (PE) dans chaque hôte, la vitesse de chaque processeur (MIPS), la RAM, la bande passante et la capacité de stockage.

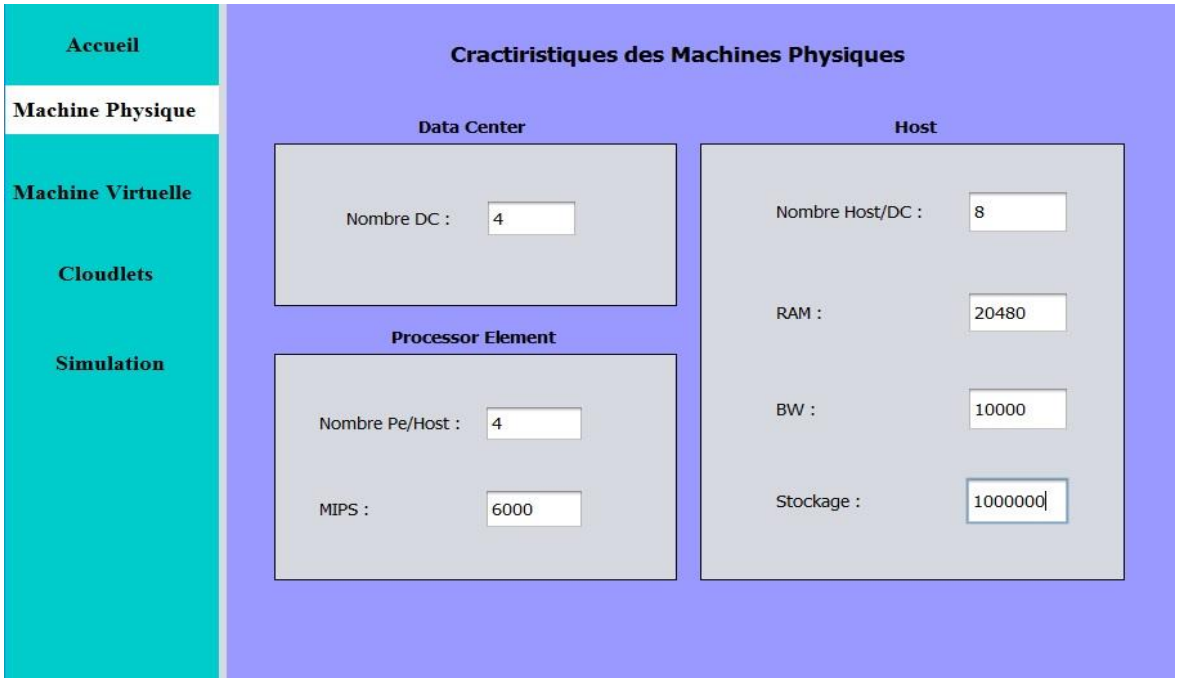


Figure 3.3 : Configuration des machines physiques.

3.6.2 Configuration des machines virtuelles

Après la configuration des différentes caractéristiques des machines physiques, nous devrons configurer les machines virtuelles *VMs*.

Les caractéristiques d'une *VM* sont : sa vitesse de traitement (MIPS), son coût d'utilisation, son taux d'échec, sa capacité de RAM, sa bande passante ainsi que son stockage.

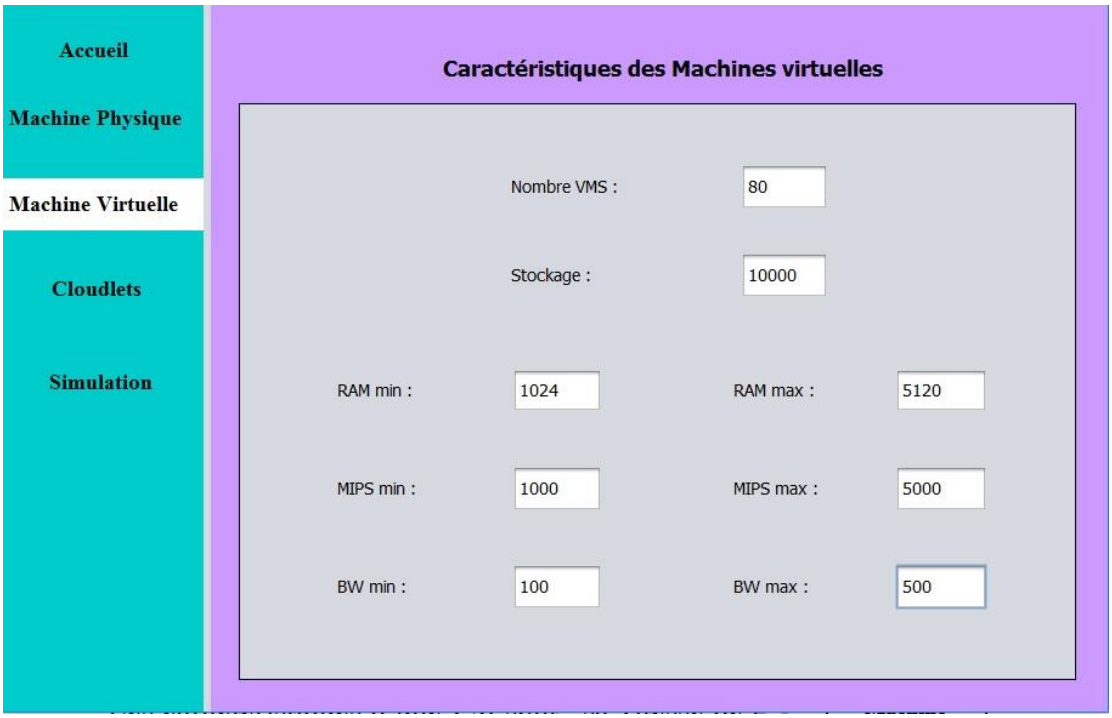


Figure 3.4 : Configuration des machines virtuelles.

3.6.3 Configuration des Cloudlets

Pour la configuration des cloudlets, il suffit d'une simple saisie qui représente les caractéristiques des cloudlets tels que leurs nombres et leurs longueurs.

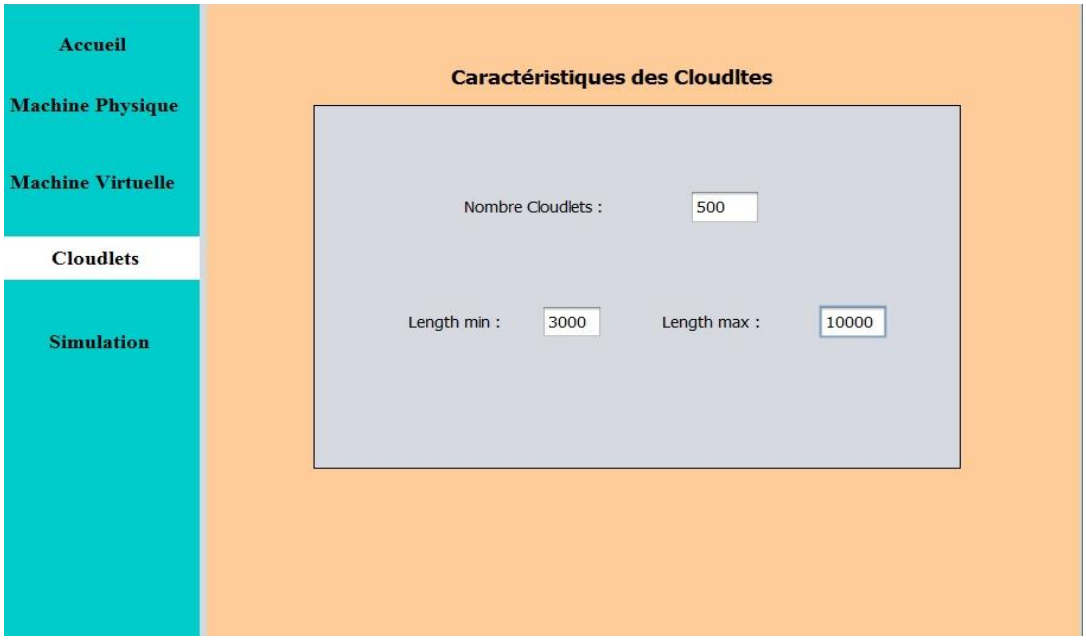


Figure 3.5 : Configuration des cloudlets.

3.6.4 La simulation

Une fois les deux phases de configuration et de pré-simulation achevées, la planification des tâches dans les différentes *VMs* est transmise au broker pour lancer la simulation et procéder à l'affichage des résultats obtenus.

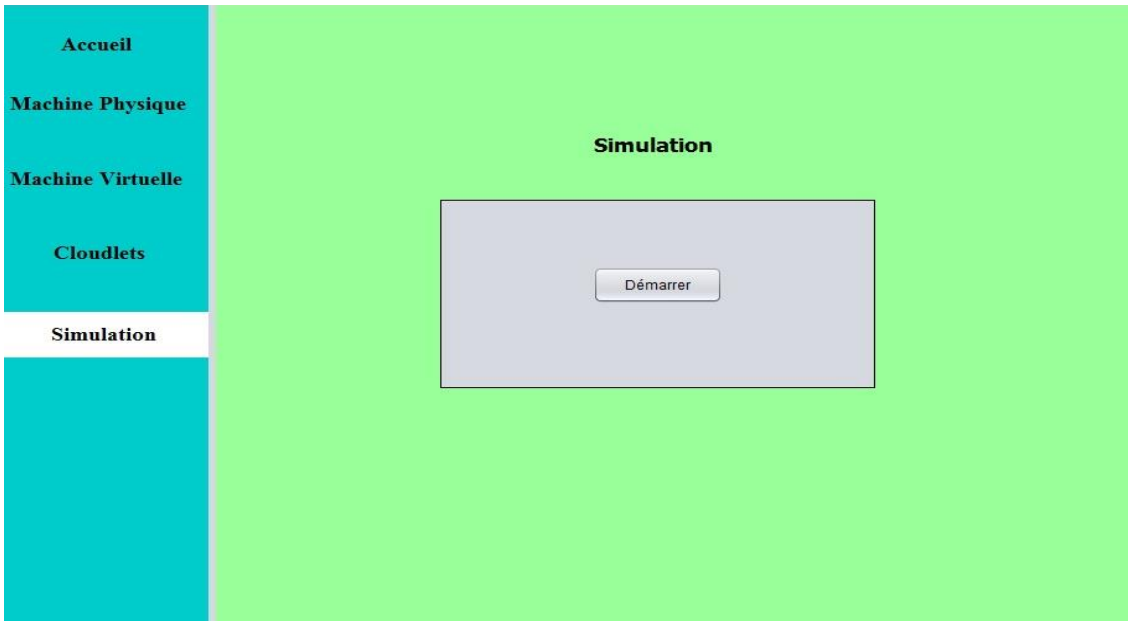


Figure 3.6 : Interface de simulation.

3.7 Résultats obtenus et étude comparative

Cette section présente la configuration des différentes simulations ainsi que les résultats obtenus après l'exécution de l'algorithme GWO pour l'ordonnancement des tâches.

3.7.1 Configuration des différentes simulations

Les différentes simulations ont été réalisées sous une machine sous Windows 10 de 64 bits, avec un processeur Intel(R) Core (TM) i5-9400M CPU @ 2.90GHz, une vitesse de 2.90 Ghz et une capacité mémoire de 4 GB.

En ce qui concerne GWO, les paramètres à définir sont les suivants: la taille de la population et le nombre d'itération pour toutes les simulations.

Il faut se référer au tableau 3.1 qui nous montre les paramètres des simulations réalisées dans un environnement hétérogène, définissant la valeur associée à chaque paramètre.

Data center	Nombre	4
Host	Nombre	8
	PES	4 (Quad core)
	MIPS	6 000
	RAM	20 GB
	Bande passante	10 GB
	Stockage	1 TB
Machine virtuelle	Nombre	60 – 80 - 100
	MIPS	1000 to 5000
	RAM	1 GB to 5GB
	Bande passante	100 MB to 500 MB
	Stockage	10 GB
	Type de politique	Time Shared
Cloudlet	Nombre	500 – 1000 - 1500
	Longueur	3000 to 10000

Tableau 3.1 : Les paramètres de simulation de CloudSim.

Puisque les algorithmes utilisés ne sont pas déterministes, une seule exécution ne peut pas suffire pour que nous ayons des conclusions. Pour cela, nous avons pris comme base le résultat obtenu après une moyenne de 10 simulations pour chaque type d'ordonnancements, afin que les résultats puissent être plus convaincants.

3.7.2 Evaluation de l'ordonnancement multi-objectif

3.7.2.1 Comparaison des résultats par rapport au makespan

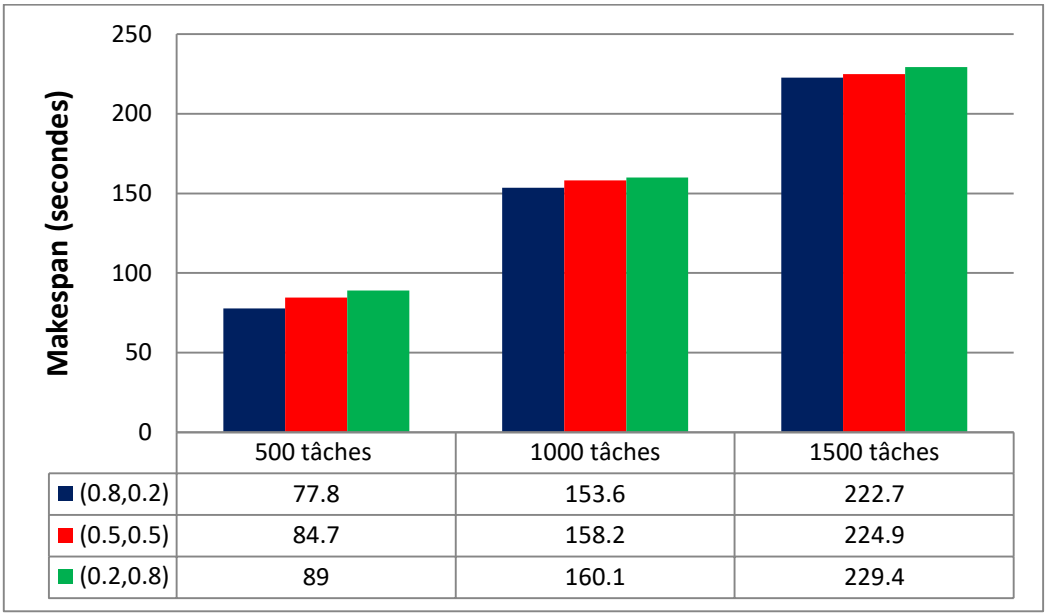


Figure 3.7 : Comparaison en termes de makespan pour 60 VMs.

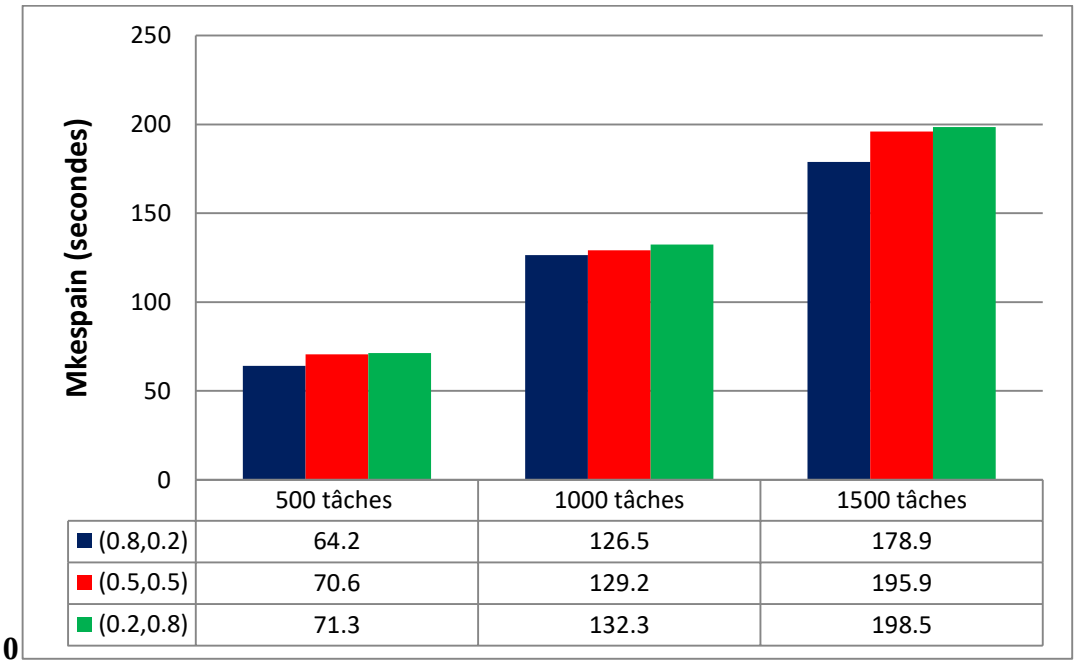
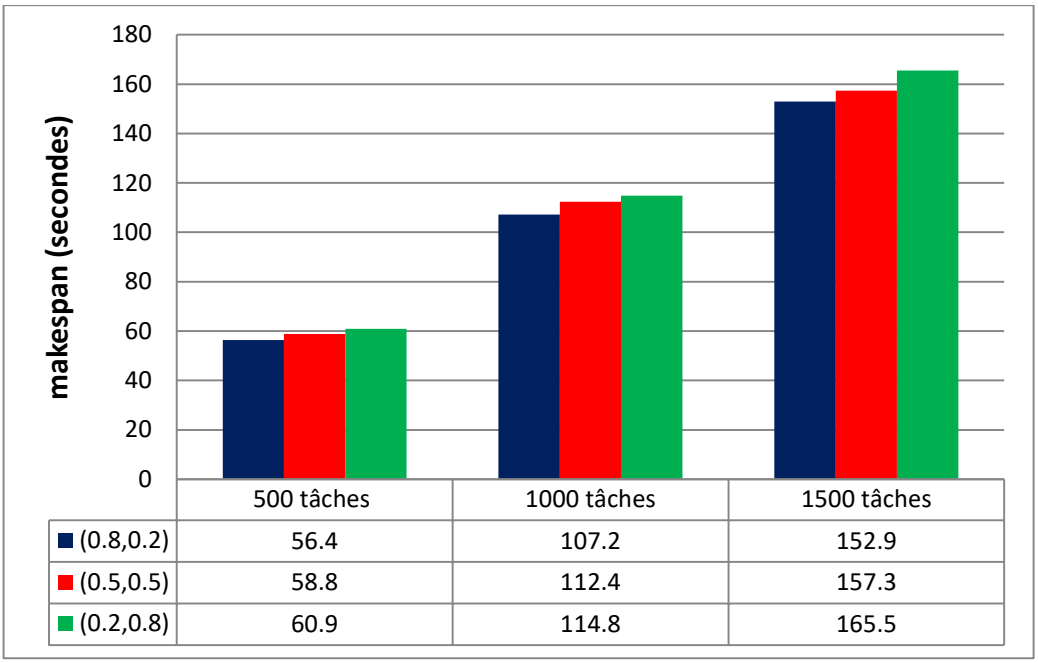


Figure 3.8 : Comparaison en termes de makespan pour 80 VMs.

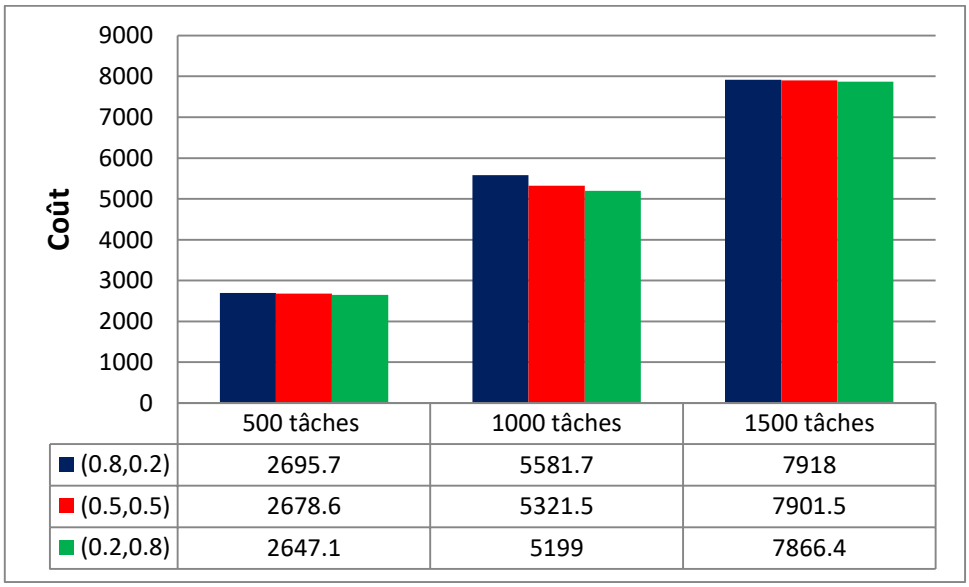


	500 tâches	1000 tâches	1500 tâches
(0.8,0.2)	56.4	107.2	152.9
(0.5,0.5)	58.8	112.4	157.3
(0.2,0.8)	60.9	114.8	165.5

Figure 3.9 : Comparaison en termes de makespan pour 100 VMs.

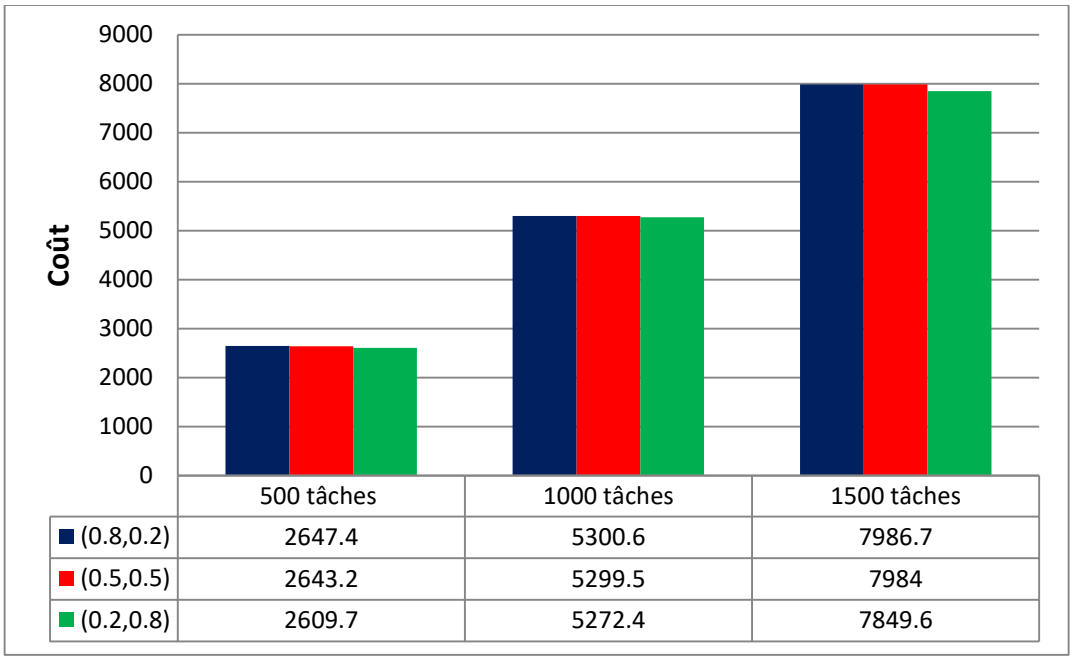
Résultats : à travers les résultats obtenus après, il a été constaté qu'il existe une proportionnalité entre le nombre des tâches et le nombre des *VMs* d'une part, la valeur du makespan de l'autre part. Nous avons remarqué que la valeur du makespan diminue à mesure que le nombre de *VMs* augmente. Nous avons également remarqué que la meilleure valeur optimisée du makespan est liée au vecteur de poids est de {0.8, 0.2} qui donne la pondération maximale au paramètre makespan.

3.7.2.2 Comparaison des résultats par rapport au coût



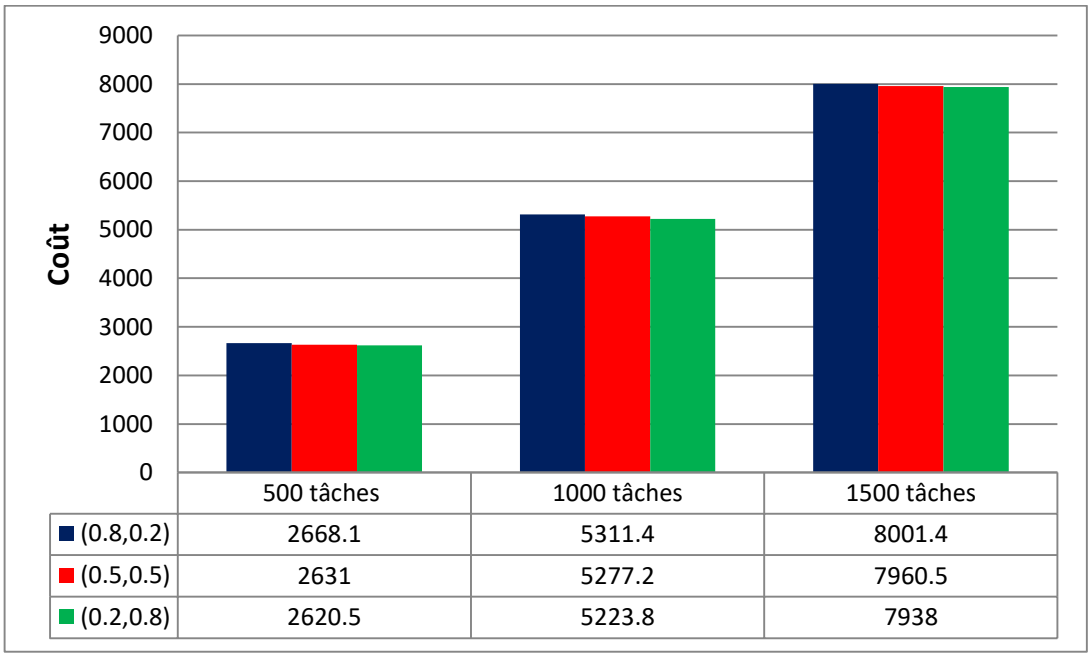
	500 tâches	1000 tâches	1500 tâches
(0.8,0.2)	2695.7	5581.7	7918
(0.5,0.5)	2678.6	5321.5	7901.5
(0.2,0.8)	2647.1	5199	7866.4

Figure 3.10 : Comparaison en termes de coût pour 60 VMs.



	500 tâches	1000 tâches	1500 tâches
(0.8,0.2)	2647.4	5300.6	7986.7
(0.5,0.5)	2643.2	5299.5	7984
(0.2,0.8)	2609.7	5272.4	7849.6

Figure 3.11 : Comparaison en termes de coût pour 80 VMs.



	500 tâches	1000 tâches	1500 tâches
(0.8,0.2)	2668.1	5311.4	8001.4
(0.5,0.5)	2631	5277.2	7960.5
(0.2,0.8)	2620.5	5223.8	7938

Figure 3.12 : Comparaison en termes de coût pour 100 VMs.

Résultats : en ce qui concerne le coût, les mêmes résultats ont été observés comme pour le makespan, puisque la valeur du coût augmente à mesure que le nombre des tâches augmentent. Sauf que l'augmentation dans le nombre des *VMs* ne s'appuie pas sur la valeur du coût. Nous avons aussi remarqué que la valeur de ce dernier atteint la

meilleure optimisation possible quand nous donnons la valeur {0.2, 0.8} au vecteur de poids qui représente la pondération maximale au paramètre coût.

Résultat global : nous remarquons qu'il existe une proportionnalité entre le makespan et le coût. Dans le cas où on veut trouver une meilleure optimisation du makespan, le coût devient forcément plus élevé. Contrairement à cela, quand on veut avoir une valeur basse du coût, la valeur du makespan augmente.

3.8 Conclusion

Nous avons consacré ce dernier chapitre du manuscrit à la présentation de notre contribution dans le cadre de ce PFE. L'objectif visé était de réaliser un ordonnancement de tâches multi-objectif dans le Cloud computing, en utilisant la métaheuristique GWO. Pour cela, nous avons utilisé des outils tels que le langage JAVA, l'IDE, NetBeans et le simulateur CloudSim. Les résultats obtenus sont très satisfaisants et l'adaptation de l'algorithme GWO pour l'ordonnancement des tâches multi-objectif dans le Cloud computing a donnée des résultats cohérents par rapport à deux critères de QoS qui sont le makespan et le coût.

Conclusion générale

Le développement du Cloud computing a connu ces dernières années une avancée importante qui est marquée par l'utilisation d'une large association de ressources informatiques dont on peut y accéder à la demande par l'intermédiaire d'internet. La majorité des services des Cloud computing sont sous la forme de tâches liées les unes aux autres. En fin de compte, ceci demande des procédés d'ordonnancement plus conceptualisés pour ordonnancer les tâches et de les faire coïncider sur les machines virtuelles adaptées, dans le but de répondre aux impératifs du QoS définies dans les contrats de service. Dans ce contexte, les clients rencontrent le problème du choix des meilleures ressources à adopter pour répondre à leurs demandes. Ce problème de choix est évalué comme étant un problème d'optimisation multi-objectif avec la présence de métriques de QoS conflictuelles, qu'il faut prendre évidemment en considération.

Nous avons utilisé dans ce projet de fin d'études une métaheuristique pour l'optimisation multi-objective afin de résoudre le problème de l'ordonnancement des tâches dans le Cloud computing. Il s'agit de la métaheuristique Grey Wolf Optimization, qui permet de trouver la solution optimale en un délai acceptable évaluant deux métriques de QoS qui sont : le makespan et le coût.

L'approche par agrégation a été utilisée dans ce travail afin de transformer, le problème de l'ordonnancement des tâches multi-objectif en un problème mono-objectif.

Les simulations que nous avons effectuées en utilisant CloudSim, ont donné des résultats très satisfaisants. Plusieurs vecteurs de poids ont été évalués afin de favoriser le makespan ou le coût selon le choix de l'utilisateur.

Au regard de notre travail accompli dans le cadre de ce PFE, il serait intéressant d'élargir le problème d'optimisation à d'autres paramètres tels que l'énergie, la fiabilité, la tolérance aux pannes...

Tester Il d'autre métaheuristique dans ce domaine comme par exemple les algorithmes génétiques, l'optimisation par essaims serait aussi intéressant.

Bibliographie

[1] Caire, Jean-Louis, and Willy Munsch. "Objectif Cloud: une démarche pratique orientée services". Éd. ENI, 2014.

[2] Reguieg Hichem. "Cloud Computing : services informatiques dynamique basés sur le Web-Concepts et notions de base.". Cours de Master "Système d'information et Donnée ". Université des Sciences et de la Technologie d'Oran. 2020.

[3] Wang, Lizhe, Jie Tao, Marcel Kunze, Alvaro Canales Castellanos, David Kramer, and Wolfgang Karl. "Scientific cloud computing: Early definition and experience." In 2008 10th ieee international conference on high performance computing and communications, pp. 825-830. Ieee, 2008.

[4] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." 2011.

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

[6] Guediri khelil, Naceri Redouane. Réalisation et conception d'un cloud computing pour l'Université d'el-Oued. Mémoire de Master en informatique. Université d'el-Oued 2015.

[7] Mansouri, Dou El Kifel. La synchronisation et les risques dans les clouds computing. Thèse de doctorat. Université Mohamed Boudiaf des Sciences et de la Technologie Mohamed Boudiaf d'Oran. 2016.

[8] Jim Dowling. Introduction to Cloud Computing. Royal Institute of Technology. Disponible sur https://www.kth.se/social/files/554fa451f276544829be2e5e/9-cloud-computing.pdf. Consulté le 24/04/2022.

[9] Différents types de structures de calcule en nuage que devriez sa voir. Disponible sur https://www.uniprint.net/fr/7-types-cloud-computing-structures. Consulté le 24/04/2022.

[10] Khadidjatou Iman Bamba. Cours Sur Les Fondamentaux des cloud computing. Disponible sur https://itandsi.files.wordpress.com/2014/09/les-fondamentaux-du-cloud-computing.pdf. Consulté le 24/04/2022.

[11] Femmam, Manel. Une approche formelle pour la planification des tâches pour la QoS dans le cloud-computing. Thèse de doctorat. Université Mohamed Khider–BISKRA, 2018.

[12] El Dor, Abbas. Perfectionnement des algorithmes d'optimisation par essaim particulaire: applications en segmentation d'images et en électronique. Thèse de doctorat. Université Paris-Est, 2012.

[13] Gherboudj, Amira. "Méthode de Résolution de Problème Difficiles Académiques." Thèse de Doctorat. Université de Constantine 2 (2013).

[14] Boussaid, Ilhem. Perfectionnement de métaheuristiques pour l'optimisation continue. Thèse de doctorat. Paris Est, 2013.

[15] Douiri, M., Souad Elbernoussi, and Halima Lakhbab. "Cours des méthodes de résolution exactes heuristiques et métaheuristiques." Université Mohamed V, Faculté des sciences de Rabat (2009): 5-87.

[16] Mouassa, Souhil. Optimisation de l'écoulement de puissance par une méthode métaheuristique (technique des abeilles) en présence d'une source renouvelable (éolienne) et des dispositifs FACTS. Thèse de doctorat. Université Ferhat ABBAS - Sétif 1. 2018.

[17] Hachimi, Hanaa. Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications. Thèse de doctorat. INSA de Rouen; École Mohammadia d'ingénieurs (Rabat, Maroc), 2013.

[18] Amara, Mohamed. Conception Optimale Des Systèmes Electriques Configurable Par colonies De Fourmis. Thèse de doctorat. 2021.

[19] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." Advances in engineering software 69 (2014): 46-61.

[20] Hiérarchie du loup gris. Disponible sur :

https://www.researchgate.net/figure/Dominance-hierarchy-of-grey-wolf-increasing-from-bottom-to-top-29_fig2_325976785. Consulté le 04/05/2022.

[21] Multi-objective grey wolf optimizer. Disponible sur : https://www.semanticscholar.org/paper/Multi-objective-grey-wolf-optimizer%3A-A-novel-for-Mirjalili-Saremi/1fb44ce4381febc34a756bfe0f2867eb73e43f35/figure/0. Consulté le 05/05/2022.

[22] Jean-Michel Doudoux. Développons en Java Version 2.20 du 28/01/2019. Disponible sur https://jmdoudoux.fr/java/dej/dej_2_20.pdf. Consulté le 01/06/2022.

[23] Netbeans. Disponible sur :

https://ma-petite-encyclopedie.org/accueil?lex_item=NetBeans. Consulté le 01/06/2022.

[24] SimGrid. Disponible sur https://simgrid.org. Consulté le 01/06/2022.

[25] GridSim. Disponible sur https://gridsim.hevs.ch. Consulté le 01/06/2022.

[26] CloudSim. Disponible sur :

https://graal.ens-lyon.fr/~ecaron/m2rts/2015/blogbazm. Consulté le 01/06/2022.

[27] Guérout, Tom. Ordonnancement sous contraintes de qualité de service dans les clouds. Thèse de doctorat. INSA de Toulouse, 2014.

[28] Barichard, Vincent, and Jin-Kao Hao. "Une approche hybride pour l'optimisation multi-objectif sous contraintes." JFPLC. 2003. pp. 33-46

[29] Benmammar, Badr, Youcef Benmouna, and Francine 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 59.2 (2019): 152-164.

Résumé :

Les performances de l'environnement de Cloud computing sont considérablement influencées par l'ordonnancement des tâches. Les deux parties, notamment, les fournisseurs et les utilisateurs des services Cloud ont des objectifs souvent opposés. Un bon ordonnanceur doit fournir une adaptation acceptable par rapport à ces objectifs. Par conséquent l'ordonnancement des tâches dans le Cloud computing devient un problème d'optimisation multi objectif. Dans ce projet de fin d'étude, nous avons travaillé sur l'ordonnancement des tâches dans le Cloud computing en utilisant la métaheuristique Grey Wolf Optimization (GWO) par rapport à deux métriques de QoS qui sont : le makespan et le coût. Les résultats obtenus des différentes simulations réalisées à l'aide de CloudSim, sont très satisfaisants.

Mots clés: Cloud computing, Optimisation multi-objectif, QoS, GWO, CloudSim.

Abstract:

The performance of the cloud computing environment is significantly influenced by the tasks scheduling. Both parties, namely, providers and users of Cloud services have often conflicting goals. A good scheduler must provide acceptable adaptation with respect to these objectives. Therefore, task scheduling in cloud computing becomes a multi-objective optimization problem. In this final year project, we worked on task scheduling in Cloud computing using the Grey Wolf Optimization (GWO) metaheuristic with respect to two QoS metrics which are makespan and cost. The results obtained from the different simulations performed using CloudSim are very satisfactory.

Key words: Cloud computing, Multi-objective optimization, QoS, GWO, CloudSim.

ملخص:

يتأثر آداء بيئة الحوسبة السحابية بشكل كبير بجدولة المهام. غالبًا ما يكون للطرفين ، لا سيما مقدمي ومستخدمي الخدمات السحابية ، أهدافًا متعارضة. يجب أن يوفر المجدول الجيد تكييفًا مقبولًا لهذه الأهداف. لذلك ، تصبح جدولة المهام في الحوسبة السحابية مشكلة تحسين متعددة الأهداف. في مشروع نهاية الدراسة هذا ، عملنا على جدولة المهام في الحوسبة السحابية بإستخدام métaheuristique تحسين الذئب الرمادي (GWO) من خلال مقياسين لجودة الخدمة وهما : وقت التنفيذ والتكلفة. النتائج المتحصل عليها من عمليات المحاكاة المختلفة التي أجريت بإستخدام CloudSim جد مرضية.

الكلمات المفتاحية: الحوسبة السحابية، التحسين متعدد الأهداف، مقاييس جودة الخدمة، GWO، CloudSim.