الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne démocratique et populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
جامعة عين تموشنت بلحاج بوشعيب
Université –Ain Temouchent- Belhadj Bouchaib
Faculté des Sciences et de Technologie
Département des Mathématiques et de L'Informatique

Projet de Fin d'Etudes
Pour l'obtention du diplôme de Master en : Informatique
Domaine : Mathématiques et Informatique
Filière : Informatique
Spécialité : Réseaux et Ingénierie des Données (RID)
Thème

Allocation multiobjectif des workflows aux ressources d'un environnement Cloud Computing

Présenté Par :
1) M. Rahmani Abderrahmen
2) M. Otmani Smain

Devant le jury composé de :

Mr. Mededjel mansour	MCB	UAT.B.B (Ain Temouchent)	Président
Mr. BENDIABDALLAH Mohammed Hakim	MCB	UAT.B.B (Ain Temouchent)	Examinateur
Mr. Bouafia Zouheyr	MAA	UAT.B.B (Ain Temouchent)	Encadrant

Année Universitaire : 2020 / 2021

Dédicace

Je dédié ce modeste travail :
A ma mère et mon père qui ont souffert sans me laisser souffrir, qui n'ont épargné aucun éffort pour me rendre heureus : mon adorable mon frére et ma soeur.

Que Dieu leur offre une longue et joyeus vie.
À mes trés chèrs collègues .

À mes chèrs frère Anis et kada
À tous mes enseignants et tous mes amis et mes collègues et mon directeur Madjdoub mustapha.

Rahmani Abderrahmen

Dédicace

Je dédié ce modeste travail :

À la femme qui a souffert sans me laisser souffrir, qui n'a épargné aucun effort pour me rendre heureuse : mon adorable mère.

À l'homme qui doit mon respect, qui a toujours été là pour moi : mon cher père.
Que Dieu leur offre une longue et joyeuse vie.

À ma tres chèrs epouse et sa famille .

À mes tres chèrs filles meriem aya et bouchra.

À mes chèrs frères ibrahim et kamel, que Dieu lui fasse miséricorde, à ma chère sœur fatima zahra.

À tous mes enseignants et tous mes amiset mes collègues. Merci pour leurs encouragements.

Otmani Smain

Remerciement

nous remercions Allah de nous avoir donne nous le courage et la volonté ainsi que la conscience et la patience et d'avoir pu terminer notre mémoire de master.

nous tiens a exprimer mes vifs remerciements a notre encadreur Mr Zouheyr Bouafia pour m'avoir donne nous l'opportunité de réaliser ce sujet sous sa direction, la confiance faite ainsi que ses conseils fructueux, et son temps consacré tout au long du travail.

Enfin, un merci particulier a tous ce qui m'ont soutenu de prés ou de loin par leurs soutiens et encouragements surtout monsieur Dine Houari .

SOMMAIRE

TABLE DES FIGURES

LISTE DES TABLEAUX

Liste des symboles

CIA : Confidentiality Integrity Availability

CIS : Cloud Information Service

CSA : Cuckoo Search Algorithme

FIFO/FCFS : First In, First Out Ou First Come, First Served

IaaS : Infrastructure as a Service

IBM : International Business Machines

IDE : Integrated Development Environment

MIPS : Million Instructions Per Second

PaaS : Platform as a Service

QOS : Quality Of Service

RR : Round Robin algorithme

SaaS : Software as a Service

SJF : Shortest Job First/ la tâche la Plus courte d'abord

SLA : Service Level Agreement

VM : Machine Virtuelle

SGWf : Système de Gestion de Workflow

NIST : National Institute of Standards and Technology

CPU : Central Processing Unit

RAM : Random-Access Memory

XaaS : XasaService

BPaaS : Business Process as a service

NaaS : Network as a Service

DaaS : Data as a Service

STaaS : Storage as a Service

AWS : Amazon web service

SDN : Software Design Network

INTRODUCTION GENERALE

Le cloud computing envahit le paysage informatique. Poussé par plusieurs facteurs convergents et complémentaires, le cloud computing évolue en tant que modèle de prestation de services informatiques viable à un rythme incroyable.

Plusieurs applications dans de nombreux domaines contiennent généralement de nombreuses workflows. Ces workflows requièrent pour leurs exécutions sur le Cloud un ordonnancement, c'est l'affectation des workflows aux ressources disponibles sur la base des exigences et les caractéristiques des workflows. C'est un processus très important dans le fonctionnement efficace du Cloud.

La plupart des algorithmes d'ordonnancement de base dans les Clouds visent à remplir un ou deux objectifs. Dans le cadre de notre projet de fin d'études nous allons proposer un algorithme d'ordonnancement workflows basé sur la métaheuristique « cuckoo search», Le but est d'optimiser trois objectifs notamment le temps d'exécution , la consommation d'énergie et le coût.

Dans le premier chapitre, nous présenterons quelques notions de base sur le Cloud Computing et les Caractéristiques des cloud computing.

Dans le deuxième chapitre,nous présenterons quelques concepts sur l'ordonnancement des workflows. Nous définissons aussi les algorithmes de base de l'ordonnancement comme FCFS (First Come First Served), Round Robin, Min-Min et SJF (Short Job First) pour les taches et un autre algorithme pour l' ordonancement des workflows.

Dans le troisième chapitre, nous présenterons quelques notions fondamentales sur les métaheuristiques notamment leurs définitions, mode de fonctionnement, caractéristiques et une classification. Ensuite nous présenterons de manière détaillée la métaheuristique « cukoo search » utilisée dans notre projet.

Dans le dernier chapitre, nous expliquerons le choix de simulateur utilisé et l'approche adoptée pour la réalisation de notre travail, nous présenterons ensuite les résultats trouvés.

CHAPITRE 1

CLOUD COMPUTING

SOMMAIRE

1.1 introduction

Le Cloud Computing, traduit le plus souvent en français par « informatique dans les nuages», « informatique dématérialisée » ou encore « info nuagique », est un domaine qui regroupe un ensemble de techniques et de pratiques consistant à accéder, en libre-service, à du matériel ou à des logiciels informatiques, à travers une infrastructure réseau (Internet).

Ce concept rend possible la distribution des ressources informatiques sous forme de services pour lesquels l'utilisateur paie uniquement pour ce qu'il utilise. Ces services peuvent être utilisés pour exécuter des applications scientifiques et commerciales..

Dans ce chapitre, nous allons présenter les notions de base liées au Cloud Computing.

1.2 Historique

Le Cloud Computing n'est pas nouveau, il est exploité depuis les années 2000, les changements qui ont permis l'apparition du Cloud Computing sont nombreux. Ainsi on peut citer l'apparition du SaaS (Software as a Service), le produit délivré par le Cloud. Puis il y a le concept de virtualisation qui permet une mutualisation des serveurs et offre donc une mise en production simplifiée et un meilleur ratio d'utilisation des ressources.
Le Cloud Computing est donc la juxtaposition de ces technologies pour passer à la vitesse supérieure sur l'exploitation de données à travers Internet. Le concept du Cloud Computing a été mis en œuvre en 2002 par Amazon,un leader due business, pour absorber la charge importante des commandes faites.[30]

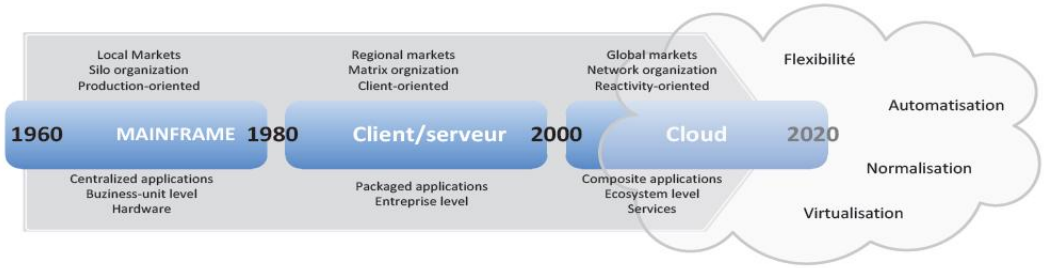


FIGURE 1.1 – historique de cloud computing [30] .

1.3 Paradigme du Cloud Computing

1.3.1 Définition

Dans cette section nous allons définir le Cloud Computing et ses différents concepts

D'après Le NIST (National Institute of Standards and Technology) : « Le Cloud Computing est un modèle permettant un accès pratique et omniprésent, sur demande, via un réseau, à un ensemble de ressources informatiques partagées et configurables (par exemple : des réseaux, des serveurs, du stockage, des applications et des services) qui peuvent être réservées et mises à disposition moyennant un effort de gestion réduit au maximum et des interactions minimales avec le fournisseur. »[21]

D'après Numergy :

«Le Cloud Computing (que l'on appelle aussi en France le nuage informatique) fournit des services ou des applications informatiques en ligne, accessibles partout, à tout moment, et de n'importe quel terminal (smartphone, PC de bureau, ordinateur portable et tablette). Pour être plus précis, le Cloud Computing permet de partager, chez un fournisseur d'offres Cloud, une infrastructure, une solution applicative ou encore une plateforme à tout utilisateur qui en fait la demande via un simple site internet (aussi appelé portail) en libre-service. »

D'après Wikipédia :

« Le Cloud Computing, ou l'informatique en nuage ou nuagique ou encore l'infonuagique (au Québec), est l'exploitation de la puissance de

calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement internet . Ces serveurs sont loués à la demande, le plus souvent par tranche d'utilisation selon des critères techniques (puissance, bande passante, etc.) mais également au forfait. »[6]

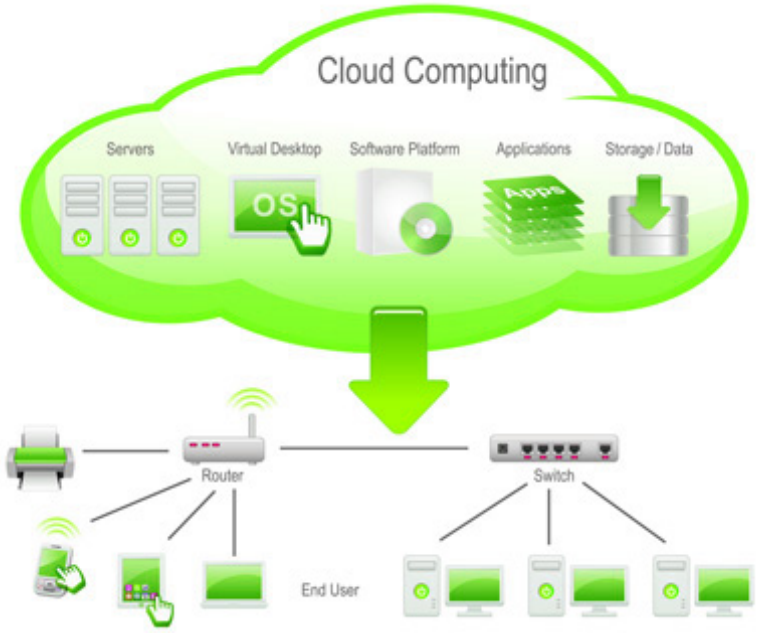


FIGURE 1.2 – Cloud computing[7] .

1.4 Notions de base

Dans cette partie, nous allons présenter quelques notions de base sur le Cloud Computing nécessaires à la compréhension de ce mémoire, notamment les centres de données, machines physiques, machines virtuelles, tâches, fournisseur...

1.Centre de données Un centre de données (Datacenter) est une infrastructure qui regroupe un ensemble d'ordinateurs, d'espace de stockage, des serveurs, des commutateurs, des routeurs, générateur électrique, un système de ventilation et de refroidissement, connexion internet puissante. Cette infrastructure est utilisée comme un outil par les entreprises pour organiser, traiter, stocker leurs grandes quantités de données.[28]

2.Machine physique

Une machine physique (Hôte, Host) est un matériel physique doté d'une

puissance de calcul, capacité de stockage que les machines virtuelles se servent de ses ressources (CPU, RAM, stockage...), pour exécuter leurs tâches.[28]

3.Machine virtuelle

La machine virtuelle (virtual machine) est le résultat de la virtualisation dans laquelle on crée une version logicielle (virtuelle) d'une entité physique, elle s'applique aux serveurs, système d'exploitation, les machines virtuelles ont pour but de réduire les dépenses, augmenter le gain de productivité.[28]

4.Tâches (Cloudlets)

Les tâches indépendants (cloudlets) :
sont l'ensemble des actions que le client souhaite exécuter au sein d'un Cloud Computing, ces actions peuvent être des opérations de stockages, calcul, traitement à distance.

5.workflows

La définition générale de workflow (ou la gestion automatique du flux de travail) désigne un travail coopératif impliquant un nombre limité de personnes devant accomplir, en un temps limité, des tâches articulées autour d'une procédure définie et ayant un objectif global .
Un workflow est composé d'un ensemble de tâches (traitements) organisées selon un ordre logique, afin de réaliser un traitement global, complexe et pertinent sur un ensemble de données sources.
Les workflows scientifiques de grande taille sont souvent de calcul intensif, il est souhaitable de répartir les tâches entre plusieurs ressources, afin d'optimiser les temps d'exécution. En tant que tel, les workflows impliquent souvent des calculs répartis sur des clusters, des grilles, et d'autres infrastructures informatiques. Récemment, les clouds computing sont évalués comme une plateforme d'exécution de workflows . L'exécution d'un workflow est gérée par un Système de Gestion de Workflow (SGWf).

5.1 Systèmes de gestion de workflows pour le clouds :

Un système de workflow cloud SwinDeW-C (Swinburne Decentralised Workflow for cloud) , contient quatre parties principales :

(i) les services de cloud,(ii) les agents d'exécution de workflow cloud,(iii) catalogue des services ,(iiii) interface d'utilisateur.Un autre travail ,qui explore l'exécution des workflows scientifiques sur les clouds,en utilisant une application d'astronomie qui s'appelle Montage.
Un des principaux modules de tous les systèmes de gestion de workflow est l'ordonnancement. Ce dernier est un processus qui "mappe" et gère l'exécution de tâches interdépendantes sur des ressources distribuées. Il alloue des ressources appropriées pour les tâches de workflow de sorte que l'exécution puisse être achevée, tout en satisfaisant les objectifs et contraintes imposés par l'utilisateur. Un ordonnancement adéquat peut avoir un impact significatif sur la performance du système. En général, le problème de mapping de tâches sur les services distribués appartient à une classe de problèmes connus comme NP-complets .

6.Fournisseur

Un fournisseur (Provider) est une société à caractère privé ou bien public offrant des services Cloud Computing tel qu'une plateforme, infrastructure, application ou de stockage, les clients payent que pour le volume de services cloud qu'ils utilisent.[23]

7.Service level agreement (SLA)

C'est un document, un accord, ou bien formellement un contrat établi entre le client et le fournisseur de services Cloud, contenant les services que le fournisseur met à disposition du client, la maintenance, ainsi il permet d'assurer aux clients des niveaux de sécurité en ce qui concerne le stockage et la gestion de leurs données confidentielles.[24]

8.Courtier

Le courtier (Broker) est une entité logicielle qui agit comme un intermédiaire entre les tâches et les datacenters, pour but de distribuer les tâches aux machines virtuelles au sein d'un datacenter. Il joue le rôle d'un négociateur entre les exigences des tâches du client et les ressources disponibles pour trouver une meilleure affectation qui satisfait QOS du client.[26]

9.Consommateur

Le consommateur (client) est un utilisateur bénéficiaire d'un service fourni

par un fournisseur de services Cloud Computing.

10.Virtualisation

La virtualisation constitue le coeur du Cloud Computing. Elle consiste à ajouter une couche logicielle qui permet de faire l'abstraction entre le matériel et le système d'exploitation dans le but de faire fonctionner plusieurs systèmes d'exploitation sur la même machine physique.[24]

1.5 Les Composants du Cloud Computing

Les technologies existantes telles que « grid computing » ou « utility computing » sont des composantes importantes du Cloud Computing.[29]

1.5.1 Applications virtualisées :

Les applications virtualisées rendent compatibles les applications de l'utilisateur avec les hardwares, les systèmes d'exploitations, le réseau et les stockages pour permettre la flexibilité du déploiement.

1.5.2 Infrastructure virtualisée :

Une infrastructure virtualisée fournit l'abstraction nécessaire pour s'assurer qu'une application ou un service ne soit pas directement attaché à l'infrastructure matérielle (serveurs, stockage ou réseaux).Ceci permet aux services de se déplacer dynamiquement à travers les ressources virtualisées d'infrastructure.

1.5.3 Gestion de sécurité et d'identité :

Le système de gestion de sécurité fournit les commandes nécessaires pour assurer les informations sensibles (les protéger) et répondre aux exi-

gences de conformité.

1.5.4 Développement :

Les instruments de développement facilitent non seulement l'orchestration de service, mais permettent également aux processus d'être développés. Ce sont les outils de développement comme le compilateur, SDK (software développement kit) et l'environnement de développement.

1.5.5 Gestion d'entreprise :

La couche de gestion d'entreprise manipule le cycle de vie des ressources virtualisation et fournit les éléments additionnels d'infrastructure comme pour la gestion du taux de disponibilité, utilisation dosée, gestion de politique, gestion de permis, et recouvrement des pertes.[5]

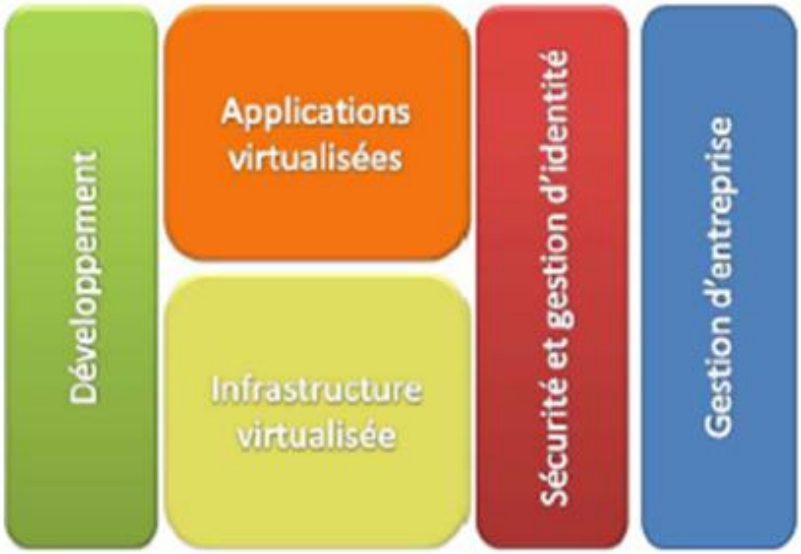


FIGURE 1.3 – Les composants du Cloud Computing.[5]

1.6 Caractéristiques du cloud computing

Le Cloud Computing possède les caractéristiques suivantes :

1.6.1 Accès en libre-service à la demande :

Le Cloud Computing offre des ressources et services aux utilisateurs à la demande. Les services sont fournis de façon automatique, sans nécessiter d'interaction humaine.[18]

1.6.2 Accès réseau universel :

Les services de Cloud Computing sont facilement accessibles au travers du réseau, par le biais de mécanismes standard, qui permettent une utilisation depuis de multiples types de terminaux (par exemple, les ordinateurs portables, tablettes, smartphones).[18]

1.6.3 Mutualisation de ressources (Pooling) :

Les ressources du cloud peuvent être regroupées pour servir des utilisateurs multiples, pour lesquels des ressources physiques et virtuelles sont automatiquement attribuées. En général, les utilisateurs n'ont aucun contrôle ou connaissance sur l'emplacement exact des ressources fournies.[31]

1.6.4 Scalabilité et l'élasticité :

Des ressources supplémentaires peuvent être automatiquement mises à disposition des utilisateurs en cas d'accroissement de la demande, et peuvent être libérées lorsqu'elles ne sont plus nécessaires (selon le besoin). L'utilisateur a l'illusion d'avoir accès à des ressources illimitées à n'importe quel moment, bien que le fournisseur en définisse généralement un seuil (par exemple : 20 instances par zone est le maximum possible pour Amazon EC2).[31]

1.6.5 Autonome :

Le Cloud Computing est un système autonome, géré de façon transparente pour les utilisateurs. Le matériel, le logiciel et les données au sein du Cloud peuvent être automatiquement reconfigurés en une seule image qui sera fournie à l'utilisateur.[16]

1.6.6 Paiement à l'usage :

La consommation des ressources dans le Cloud s'adapte au plus près aux besoins de l'utilisateur. Le fournisseur est capable de mesurer de façon précise la consommation (en durée et en quantité) des différents services (CPU, stockage, bande passante. . .), cela lui permettra de facturer l'utilisateur selon sa réelle consommation.[1]

1.6.7 Fiabilité et tolérance aux pannes :

Les environnements Cloud tirent parti de la redondance intégrée du grand nombre de serveurs qui les composent en permettant des niveaux élevés de disponibilité et de fiabilité pour les applications qui peuvent en bénéficier.[32]

1.6.8 Garantie QoS :

Les environnements du Cloud peuvent garantir la qualité de service pour les utilisateurs, par exemple, la performance du matériel, comme la bande passante du processeur et la taille de la mémoire. [16]

1.6.9 Basé-SLA :

Les Clouds sont gérés dynamiquement en fonction des contrats d'accord de niveau de service (SLA) entre le fournisseur et l'utilisateur. Le SLA dé-

finit des politiques, telles que les paramètres de livraison, les niveaux de disponibilité, la maintenabilité, la performance, l'exploitation, ou autres attributs du service, comme la facturation, et même des sanctions en cas de violation du contrat. Le SLA permet de rassurer les utilisateurs dans leur idée de déplacer leurs activités vers le Cloud, en fournissant des garanties de QoS [31].

1.7 Modèles de déploiement

Les modèles de déploiement qui se réfèrent à l'emplacement des infrastructures et ressources de la solution Cloud Computing et dans quel objectif.Les principaux modèles de déploiement sont : public, privé, communautaire et hybride.

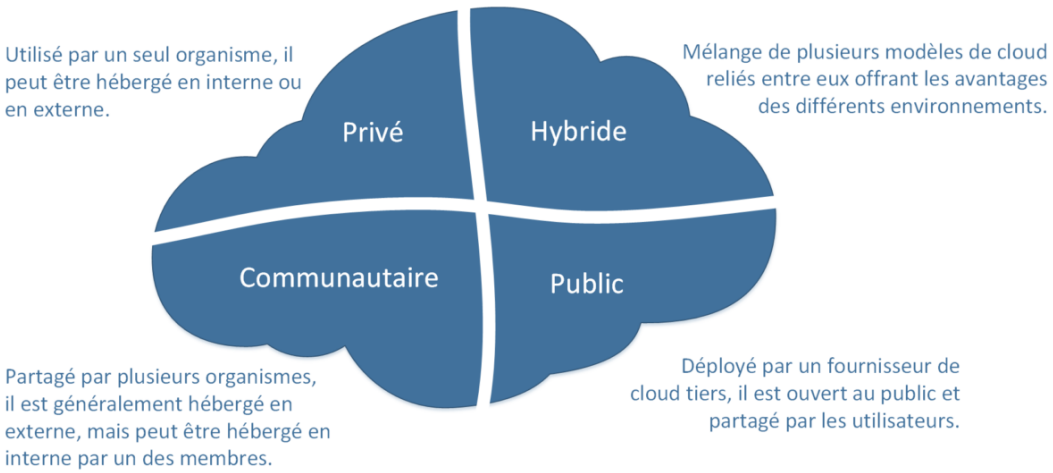


FIGURE 1.4 – Les Modèles de déploiement.[9]

1.7.1 Cloud privé

L'ensemble des ressources d'un Cloud privé est exclusivement mis à disposition d'une entreprise ou organisation unique. Le Cloud privé peut être géré par l'entreprise elle même (Cloud privé interne) ou par une tierce partie (Cloud privé externe). Les ressources d'un Cloud privé se trouvent généralement dans les locaux de l'entreprise ou bien chez un

fournisseur de services. Dans ce dernier, l'infrastructure est entièrement dédiée à l'entreprise et y est accessible via un réseau sécurisé (de type VPN). L'utilisation d'un Cloud privé permet de garantir, par exemple, que les ressources matérielles allouées ne seront jamais partagées par deux clients différents.[31]

1.7.2 Cloud public

L'infrastructure d'un cloud public est accessible à un large public et appartient à un fournisseur de services. Ce dernier facture les utilisateurs selon la consommation et garantit la disponibilité des services via des contrats SLA.[31]

1.7.3 Cloud communautaire

L'infrastructure d'un cloud communautaire est partagée par plusieurs organisations indépendantes ayant des intérêts communs. L'infrastructure peut être gérée par les organisations membres ou par un tiers. L'infrastructure peut être située, soit au sein des dites organisations, soit chez un fournisseur de service.[31]

1.7.4 Cloud hybride

L'infrastructure d'un Cloud hybride est une composition de plusieurs Clouds (privé, communautaire ou public). Les différents Clouds composant l'infrastructure restent des entités uniques, mais sont reliés par une technologie standard ou propriétaire permettant ainsi la portabilité des données ou des applications déployées sur les différents Clouds.[31]

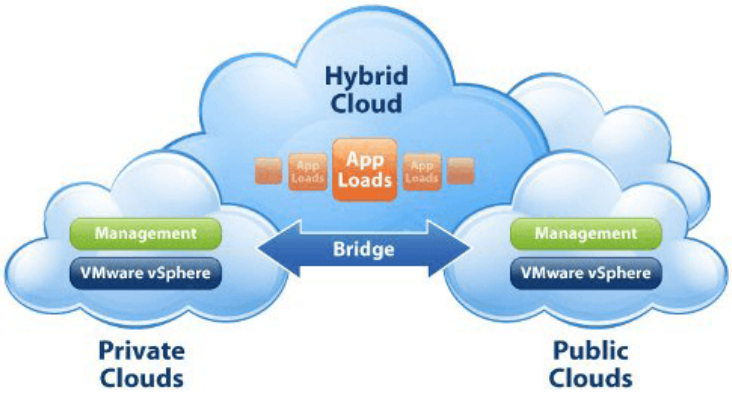


FIGURE 1.5 – Cloud hybride[33]

1.8 Les différents niveaux de Cloud Computing :

Il existe généralement trois types de services dans le Cloud computing. Ces services s'organisent en trois niveaux successifs : le niveau infrastructure (IaaS), le niveau plateforme (PaaS) et le niveau application (SaaS).[30]

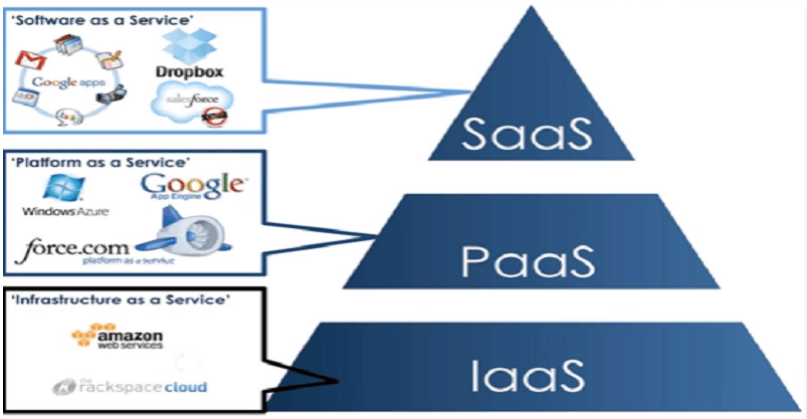


FIGURE 1.6 – Les differents niveaux du Cloud Computing[34]

1.8.1 Infrastructure as a Service (IaaS) :

Les services IaaS du Cloud computing permettent de mettre à la disposition des utilisateurs des ressources matérielles (réseau, stockage, systèmes d'exploitation) accessibles sous forme virtuelle.[30]

Avantage de (IaaS) : Grande flexibilité, contrôle total des systèmes (administration à distance par SSH ou Remote Desktop, RDP), qui permet d'installer tout type de logiciel métier.[30]

Inconvénient de (IaaS) : Besoin d'administrateurs système comme pour les solutions de serveurs classiques sur site. Les cibles sont les responsables d'infrastructures informatiques. Amazon EC2 est le principal qui propose ce genre d'infrastructures. Eucalyptus est un exemple d'infrastructure.[3

1.8.2 Platform as a Service (PaaS) :

Les services du type PaaS disposent des environnements spécialisés au développement d'applications comprenant les outils et les modules nécessaires pour ce type de travail. Selon Etchevers et al (2011), il s'agit des environnements d'exécution qui permettent de gérer le cycle de vie des applications. Ce cycle de vie comprend notamment les phases de conception, de déploiement et plus généralement d'administration des applications.[30]

Avantage de (PaaS) : Le déploiement est automatisé, pas de logiciel supplémentaire à acheter ou à installer.[30]

Inconvénient de (PaaS) : Limitation à une ou deux technologies (ex. : Python ou Java pour Google AppEngine, .NET pour Microsoft Azure, propriétaire pour force.com). Pas de contrôle des machines virtuelles sous jacentes.[30]

1.8.3 Software as a Service (SaaS) :

Les services SaaS du Cloud computing permettent de mettre à la disposition des utilisateurs des applications prêtes à l'emploi. A la différence des applications Web ordinaires, les services SaaS du Cloud computing sont caractérisés par un haut niveau d'abstraction qui permet d'adapter l'application à un cas particulier d'usage.[30]

Avantage de (SaaS) : Plus d'installation, plus de mise à jour (elles sont continues chezle fournisseur), plus de migration de données etc. Paiement à l'usage. Test de nouveaux logiciels avec facilité.[30]

Inconvénient de (SaaS) : Limitation par définition au logiciel proposé. Pas de contrôle sur le stockage et la sécurisation des données associées au logiciel. Réactivité des applications Web pas toujours idéale.[30]

1.9 Apports et risques

1.9.1 Apports

L'arrivée du Cloud Computing est considérée comme une révolution dans le secteur informatique et économique, nous citons quelques apports de Cloud Computing :[14]

1.Réduction des coûts

Le Cloud Computing donne accès à des services coûteux à moindre prix et de manière évolutive.[14]

2.L'accessibilité et Mobilité

Grâce au Cloud, l'utilisateur peut utiliser ces services à distance, à n'importe quel moment et de n'importe quel appareil.[14]

3.Disponibilité

Le client peut profiter d'un service quand il veut et de manière dynamique (évolutive) et transparente.[14]

1.9.2 Risques

Les fournisseurs font face à de nombreux risques qui sont définis comme des événements incertains avec une probabilité de se produire et d'avoir un impact négative (menace). La démarche Cloud étant récente, les fournisseurs doivent mettre en place un système pour la gestion des risques du moment que le risque zéro n'existe pas.[14]

CIA

La confidentialité, l'intégrité et la disponibilité des données sont les principales préoccupations des clients lors d'une migration vers une solution Cloud.[14]

a. La confidentialité : Il est primordial pour les clients de prévenir le risque de perdre des données confidentielles en imposant à leurs fournisseurs des normes de sécurité.

b. Intégrité : L'intégrité de données qui peut-être associée à la non-répudiation des données : le récepteur d'un message a la possibilité de savoir qu'une donnée a été modifiée ou bien consultée a l'aide des mécanismes de sécurité.

c. Disponibilité : L'accès aux ressources doit être permanente et évolu-

tive.

La continuité du métier

Le Cloud Computing étant récent, de nombreux fournisseurs sont tous récents aussi et apprennent en même temps que leurs clients. Que se passeraitil si le fournisseur disparaissait ?, la continuité du métier du client est quasiment importante.[14]

Les coûts cachés

Une entreprise doit mener une étude avant de prendre la décision de migrer ses centres de données vers une solution cloud. Une étude de cas proposée par McKinsey indiquait que les coûts augmenteraient de 144 %.[14]

La perte des données

La perte des données est le risque majeur le plus significatif. Il faut donc s'assurer de la bonne sauvegarde des données.[14]

1.10 Conclusion

Dans ce chapitre, nous avons présenté des notions fondamentales et les caractéristiques sur le Cloud Computing. Dans le chapitre suivant nous allons présenter quelques concepts sur l'ordonnancement des workflows. Nous avons aussi mis l'accent sur quelques algorithmes de base de l'ordonnancement comme Min-Min, Round Robin, FIFO et SJF, ensuite un algorithme (Priority-based Job Scheduling Algorithm) d' ordonnancement des tâches dépendants (workflows).

CHAPITRE 2

ORDONNANCEMENT DES TÂCHES ET WORKFLOWS

SOMMAIRE

2.1 Introduction

L'informatique dans le nuage ou le Cloud computing est un nouveau modèle de prestation de service informatique utilisant de nombreuses technologies existantes.

L'ordonnancement des workflows et l'allocation de ressources sont souvent considérés comme des vrais challenges pour les gestionnaires dans ce type de technologies.

C'est ainsi que de nombreux travaux ont été consacrés à la recherche des solutions pour remédier a ces problèmes.

Dans ce chapitre, nous allons présenter quelques concepts sur l'ordonnancement des workflows et introduire les algorithmes de base dans ce domaine.

2.2 Ordonnancement des workflows

L'ordonnancement des workflows (les taches dépendantes) dans le Cloud Computing, permet de définir la manière d'allouer des ressources aux utilisateurs afin d'exécuter leurs workflows. La démarche Cloud Computing étant récente, L'ordonnancement des workflows et l'allocation des ressources dans le Cloud Computing sont considérés comme l'un des enjeux essentiels, plusieurs recherches ont été consacrées afin d'atteindre une gestion optimale des ressources au sein d'un Cloud Computing.

À travers cette partie nous allons présenter, quelques définitions, les principaux algorithmes d'ordonnancement des workflows dans le Cloud Computing.

2.2.1 Définition

L'ordonnancement des workflows dans le Cloud Computing est le processus d'affectation des workflows aux machines machines virtuelles. Cette attribution joue un rôle très important sur les performances du datacenter, ainsi que d'autres exigences de l'utilisateur décrites dans le SLA.[3]

2.2.2 Problème d'ordonnancement

Le problème d'ordonnancement consiste à la manière d'organiser dans un temps déterministe, la réalisation des workflows, en prenant en considération des contraintes temporelles (contraintes de délai, contraintes d'enchaînement, ...), ainsi que des contraintes liées à l'utilisation et la disponibilité des ressources.[15]

2.2.3 Rôle d'ordonnanceur

Le rôle d'un ordonnanceur est de trouver la machine virtuelle appropriée pour traiter les taches qui lui sont soumises. Les ordonnanceurs se varient de plus simple (Allocation Round Robin) au plus compliqué (selon des contraintes).[25]

2.2.4 Algorithmes de base sur l'ordonnancement des taches

Nous allons présenter dans ce qui suit, les principaux algorithmes de base d'ordonnancement des taches :

Algorithme Min-min

L'algorithme procède comme suit : il calcule le temps d'exécution minimale pour toutes les workflows afin de choisir la valeur minimale entre ces temps minimum, qui représente le temps minimum d'exécution parmi toutes les taches sur les ressources. Ensuite, en fonction de ce temps minimum, le workflows est ordonnancée sur la machine correspondante. Puis le temps d'exécution pour toutes les autres workflows est mis à jour sur cette machine en ajoutant le temps d'exécution de la tâche assignée a des temps d'exécution des autres tâches sur cette machine/ressource et la tache assignée est supprimée de la liste des taches. la même procédure est répétée jusqu'à ce que toutes les taches soient assignées sur les ressources.[30]

Algorithme Round Robin

Le principe de l'algorithme est simple qui consiste à distribuer de façon équitable les taches sur les machines virtuelles disponibles, autrement dit le nombre de tâches pour chaque machine virtuelle est le même. L'algorithme Round Robin [22]est implémenté par défaut dans le simulateur

CloudSim.[30]

Algorithme FIFO/FCFS

L'algorithme FIFO (First In First Out) ou FCFS (First Come First Served) est l'un des algorithmes les plus simples. L'idée de base est d'ajouter chaque tache et ressource disponible dans une file et exécuter chaque tâche et ressource par leur ordre d'arrivée.[30]

Algorithme Shortest Job First/Plus court d'abord

Le principe de l'algorithme SJF ressemble au FIFO, on choisit d'exécuter la tache qui sera plus courte, au lieu d'exécuter dans l'ordre d'arrivée. Le problème se pose dans la détermination du temps d'exécution d'une tache avant de l'exécuter et pour cela il faut se baser sur une estimation.[30]

2.2.5 les algorithmes pour l'ordonnancement des workflows

Priority-based Job Scheduling Algorithm

En Cloud computing, une approche innovante pour traiter le travail de programmation en utilisant des mesures mathématiques .[2]

L'importance du travail pour la programmation est considérée par cet algorithme et s'appelle l'algorithme pour la priorité basée algorithme de planification des workflows « PJSC ». Il est centré comme le modèle de prise de décision des normes multiplicatives. Il y a trois niveaux de priorités dans cet algorithme qui sont niveau de programmation, niveau de ressources et niveau de travail qui est illustré à la figure 2.1.[2]

Dans cet algorithme, les travaux sont pris comme J= J1, J2, J3, J,. . . ,Jm qui demande des actifs dans une atmosphère cloud et les ressources définies sont prises comme C= C1, C2, C3, C4,. . . .,Cd c'est-à-dire présenté dans l'atmosphère nuageuse comme entrée où (d«m). Chaque ensemble de travaux demande une ressource avec la priorité requise. Priorité de différents ensembles de tâches est comparé indépendamment . Chaque travailse voit allouer une ressource avec la priorité spécifiée. D'où le les réseaux de corrélation de chaque activité/ensemble de tâches sont calculés selon les perspectives de récupération des ressources, et une matrice de comparaison des ressources est également calculée. Maintenant la matrice normale

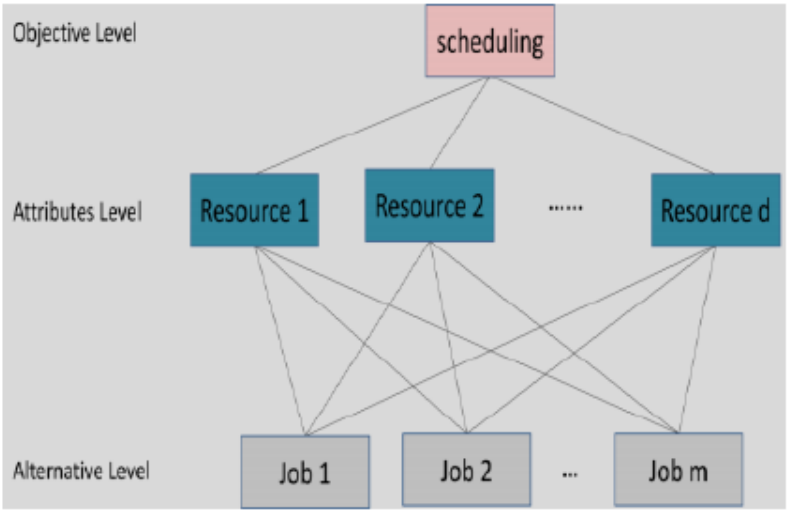


FIGURE 2.1 – Ordonnancement basé sur trois niveaux de priorité .[2]

de tous les emplois portant le nom est calculée par comparer chacune des matrices d'ensembles de workflowss et des chemins prioritaires sont également calculés . Alors la matrice des ressources normales est calculé, et le nom de la matrice est donné comme Gamma . [2]

Maintenant, le VPS (vecteur prioritaire de s) est calculé dans cet algorithme et S est indiqué comme un ensemble d'emplois. La matrice est multipliée par la matrice qui a entraîné des VPS. Maintenant, le travail le mieux classé est choisi et la ressource est attribuée à cet emploi. La liste du travail est mise à niveau avec l'heure et la procédure de programmation procède jusqu'au point que tous les emplois sont planifiés dans une ressource appropriée . Les essais / expériences se présentent de montrer que le calcul de l'algorithme a la complexité rationnelle. Il existe également quelques problèmes liés à ce calcul, par exemple, le temps d'achèvement, la cohérence et la complexité.[2]

Autre algorithmes pour l'ordonnancement des workflows

algorithme	Resumé	Parametre	Outil	Avantage	inconvénients.
Compromise time-cost scheduling algorithm (CTC) [41]	Un algorithme d'ordonnancement CTC pour l'exécution de workflow dans le Cloud computing.Il est centré sur des contraintes d'une relation interactive entre le temps et le cout comme un compromis qui est basé sur des caractéristiques du Cloud	Makespan, cout monétaire	SwinDeWC	L'algorithme a une pré-étape pour découvrir et réordonner les taches echouées. Il exploite l'effet interactif entre le cout et le deadline qui agit sur la performance du workflow. De plus, il permet a l'utilisateur de redéfinir leurs deadline et leurs couts dans chaque cycle de l'ordonnancement	Il ne considére pas simultanément les deux contraintes dans le workflow pour minimiser la performance totale
Learning architecture for scheduling (LA)	L'article propose une nouvelle approche pour l'ordonnancement du workflow dans le Cloud computing, c'est l'architecture d'apprentissage qui utilise un processus de décision pour diriger optimalement le processus d'exécution du workflow selon l'état de l'environnement	Makespan, cout monétaire	Cloudsim	Il s'adapte automatiquement au changement d'environnement des ressources par l'apprentissage. De plus, il garantit l'exécution réussie du workflow	L'espace d'état pour faire des taches est grande en incluant non-utilisation des ressources selon le temps. Il ne considére pas les types de VMs. Il répéte l'évaluation de la fonction fitness

Multiple QoS constrained scheduling strategy of multiple workflows (MQMW)	La stratégie peut faire l'ordonnancement pour multiple workflows qui sont démarrés tous en méme temps et les exigences de QoS sont prises en compte. Il considére 4 facteurs qui affectent grandement le makespan, le cout et le taux de réussite du workflow	Taux de réussite, cout, temps, makespan	CloudSim	Il s'accorde avec les multiples workflows et le multiple-objectifs optimal en méme-temps.De plus, il considére la performance totale par 3 contraintes. Une tache est toujours terminée	Il ne fait pas un ré-ordonnancement quand une tache n'est pas terminée
Scheduling Scientific Workflows Elastically SSWE 40	SSWE fait l'ordonnancement d'un workflow élastique sur le Cloud computing pour optimiser le temps d'exécution du workflow et met a échelle elastique des ressource lors de l'exécution	Le temps d'exécution, capacité	CloudSim	Il considére les changements élastiques des ressources quand le workflow s'exécute.De plus , les ressources peuvent etre assignées seulement quand elles sont nécessaire	Il groupe des ressources qui sont de meme capacité de calcul dans un cluster. Il ne considére pas d'autres caractéristiques de VMs comme : le prix,le stockage ,la bande passante, etc

TABLE 2.1 – Autrs algorithmes pour l' ordonnancement des workflows.[15]

2.3 Comparaison entre les algorithme d'ordonnancement basique.

Algorithmes	Méthodologie	Paramètres	Avantages	Inconvénient
Genetic Algorithm	L'algorithme génétique représente domaine de solution et une fonction appropriée pour estimer le domaine de solution.	1- La taille de population. 2- probabilité de Crossover. 3- probabilité de mutation.	1- Il peut résoudre les problèmes mathématiques et financiers avec plus de précision. 2- Facile à comprendre les concepts. 3- Certaines demandes ont nécessité moins de temps pour le traitement.	1- Cet algorithme fonctionne très lentement. 2- Cet algorithme ne peut pas trouver les solutions exactes. 3- La méthode de sélection devrait être appropriée.
Greedy Algorithm	L'algorithme essaie de trouver l'optimum global en suivant l'approche heuristique de résolution des problèmes.	1- Paramètre µ. 2- Domaine D. 3- Population n.	1- Facile à mettre en œuvre. 2- Cet algorithme nécessite moins de ressources. 3- L'exécution est très rapide. 4- La planification est très rapide.	1- La solution d'optimisation globale n'est pas atteinte par cet algorithme. 2- Très difficile d'apporter des modifications aux paramètres.
Priority-Based Job Scheduling Algorithm	1- Utilisé dans le cas des tâches dépendent workflow.	1- Priorité à chaque file du d'attente 1. La priorité du processus augmente avec l'augmentation du temps. 2- Facile à utiliser. 3- Idéal pour les applications qui nécessitent du temps et ressources.	1. La priorité processus augmente avec l'augmentation du temps. 2. Facile à utiliser . 3. Idéal pour les applications qui nécessitent du temps et resources. 4. temps d'!!!	1- Les travaux ayant la priorité la plus basse seront perdus lorsque le système se bloque. 2- La famine en ressources.

Round Robin	L'algorithme fonctionne sur une approche cyclique dans laquelle chaque tâche a une chance égale d'être choisie et a une unité de temps tout aussi petite pour exécution	1- Temps d'arrivée. 2- Time slice.	1- Un bon temps de réponse. 2- La charge est équilibrée. 3- Moins complexe.	1- La préemption provoque l'arrêt du processus une fois la tranche de temps est termine.
Particle Swarm Optimization	L'algorithme utilise la population pour trouver les valeurs minimales optimales qui aident à créer un ordre correct des tâches et planifier la tâche vers une ressource appropriée	1- Inertie. 2- Constantes C1, C2.	1- utilisation maximale des ressources, trouver la solution optimale, minimiser le temps de traitement.	1- Vitesse de convergence lente si l'espace de recherche est grand.
Min-Min Algorithm	L'algorithme sélectionne la tâche avec temps d'exécution minimale	1- Makespan.	1- Meilleur makespan.	1- Déséquilibre de charge. 2- Mauvaise qualité de service.

TABLE 2.2 – Comparaison entre les algorithme d'ordonnancement basique.[11]

.

2.4 Conclusion

Dans ce chapitre, nous avons présenté quelques concepts sur l'ordonnancement des workflows.

Nous avons aussi présenté algorithmes de base de l'ordonnancement comme Min-Min, Round Robin, FIFO, SJF et on a présenté un algorithme pour les workflows Priority-based Job Scheduling Algorithm et une comparaison entre les algorithmes d'ordonnancementdes tâches.

Étant donné que notre mission est de proposer un algorithme d'ordonnancement des workflows multiobjectifs a travers une métaheuristique « cuckoo search » , Dans le chapitre suivant nous allons présenter quelques notions fondamentales sur les métaheuristiques.

CHAPITRE 3

LES MÉTAHEURISTIQUES

SOMMAIRE

3.1 Introduction

Dans la vie pratique, on se retrouve souvent confronté à des problèmes de différentes complexités, pour lesquelles on cherche des solutions qui satisfont deux notions antagonistes : la rapidité et la qualité. Devant le coût de recherche prohibitif des méthodes exactes (particulièrement avec des problèmes de grande taille) et la spécificité des heuristiques au problème donné, les métaheuristiques construisent une solution moins exigeante. En fait, elles sont applicables sur une grande variété de problèmes d'optimisation de différentes complexités. En outre, elles permettent de fournir des solutions de très bonne qualité (pas nécessairement optimales) en temps de calcul raisonnable.

Dans ce chapitre, nous allons présenter quelques notions fondamentales sur les métaheuristiques notamment leurs définitions, mode de fonctionnement, caractéristiques et une classification. Ensuite nous allons présenter de manière détaillée l'algorithme Cuckoo search.

3.2 Définition

Une métaheuristique est un algorithme d'optimisation qui s'applique pour la résolution des problèmes d'optimisation difficiles et complexes souvent dans les domaines de la recherche opérationnelle ou de l'intelligence artificielle. Les métaheuristiques sont généralement des algorithmes stochastiques, qui cherchent à faire progresser vers un optimum global, autrement dit l'extremum global d'une fonction par échantillonnage d'une fonction objectif.[4]

L'idée de base est d'avoir une flexibilité en échange de réduire les exigences vise à vis la qualité de la solution trouvée, leur but est de trouver des solutions bonnes en temps acceptable mais aucune garantie de la solution trouvée, il peut exister d'autres solutions optimales dans l'espace de recherche. Parmi les métaheuristique existants, nous citons : la recherche taboue, le recuits simulé, les colonies de fourmis et les algorithmes révolutionnaires.[4]

3.3 Caractéristiques

Les caractéristiques essentielles des métaheuristiques sont les suivantes :
1. Les métaheuristiques sont basées sur des paramètres qui guident l'exploration et affectent la qualité de la solution.[4]
2. Les métaheuristiques nécessitent la spécification d'un point de départ pour l'exploration, qui est choisie aléatoirement.[4]
3. Les métaheuristiques nécessitent la spécification d'une condition d'arrêt, par exemple un temps de CPU maximum, nombre d'itération.
4. Les métaheuristiques parcourt un espace de recherche en se basent sur la combinaison de deux actions :

-Intensification (exploitation) :

Cette action consiste à procéder à une recherche intensifiée autour d'une solution déjà existante dite prometteuse.

-La diversification (exploration) :

Cette action consiste à explorer de nouvelles régions de l'espace de recherche (souvent par randomisation) qui peut être pas encore visitée.
5. Les métaheuristiques ne font généralement aucune garantie de la qualité de la solution trouvée, leur traitement est souvent stochastique et peuvent donner des différents résultats sur une même instance du problème, plus on augmente le nombre des itérations plus on a de chance d'obtenir une meilleure solution.

3.4 Principe de base

Les métaheuristiques sont basées généralement sur des principes communs qui sont les suivants :
1.On a un espace de recherche nommé S et une fonction fitness f : S $\rightarrow$ R.
2.On définit un voisinage V (x) pour chaque $X \in S$, qui est l'ensemble des points $Y \in S$ que X peut atteindre, ce voisinage est spécifie par un ensemble de transformation Ti, appelée aussi mouvement qui permet d'obtenir Y à partir de X, donc Si $Y \in V(x)$ alors il existe un i tel que Y = Ti(x).[4]
3.On spécifie un opérateur dit d'exploration U qui a pour but de choisir à

partir d'un X0 le prochain point $X1 \in V(X0)$ de la trajectoire de la recherche, l'opérateur va se baser sur l'utilisation des valeurs de la fonction fitness (objectif) dans le voisinage de V (x0). Ce processus d'exploration : $X0 \rightarrow X1 \in V(X0) \rightarrow X2 \in V(X1). \ldots$, continue jusqu'à l'obtention d'un critère d'arrêt (nombre d'itération, temps CPU).[4]

3.5 Classification des métaheuristiques

Les métaheuristiques peuvent être classés selon plusieurs paramètres. La figure ci-dessous, présente la classification des métaheuristiques selon le paramètre suivantes :

3.5.1 Fonction objectif

C'est-à-dire la manière d'utiliser la fonction objectif par une métaheuristique.
Cette catégorie comprend deux types de métaheuristiques statique et dynamique.
Supposons qu'on a un problème d'optimisation afin de maximiser ou bien de minimiser une fonction f sur un espace de recherche de solutions nommé S, alors nous avons :

- Les métaheuristiques qui travaillent directement sur f dites statiques.
- Les métaheuristiques dynamiques qui travaillent sur une fonction g obtenue à partir de la fonction f en ajoutant quelques composants à fin de changer la topologie l'espace de recherche S.[4]

3.5.2 Nombre de solutions

Nous distinguons deux familles de métaheuristiques : à base de solution unique (S-métaheuristique) et les métaheuristiques à base de population de solutions (Pmétaheuristique).

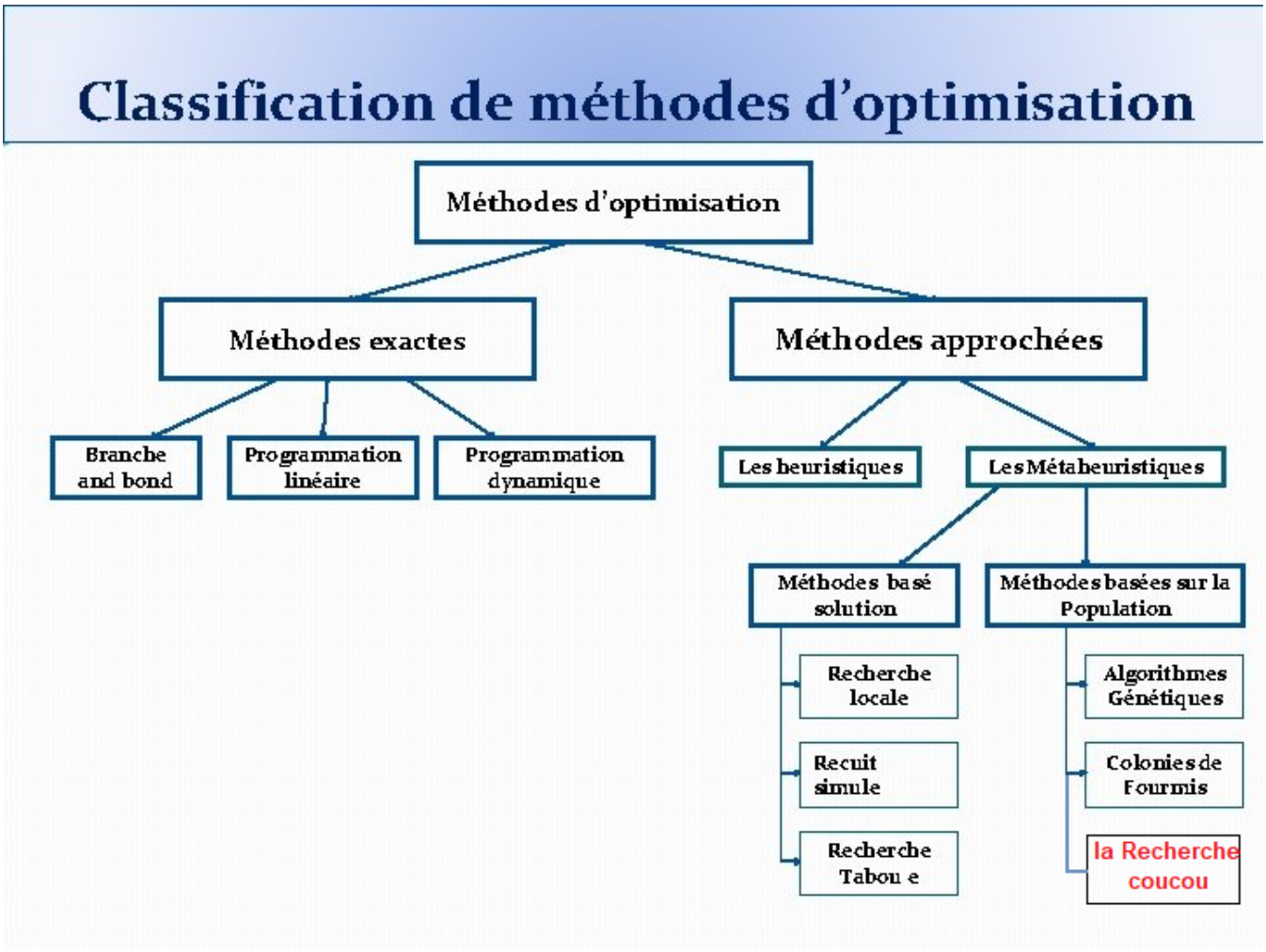


FIGURE 3.1 – Classification de méthodes de résolution de problèmes d'optimisation. [8] .

a) Métaheuristiques à base de solution unique

Les métaheuristiques à base de solution unique débutent la recherche avec une seule solution initiale. Elles se basent sur la notion du voisinage pour améliorer la qualité de la solution courante. En fait, la solution initiale subit une série de modifications en fonction de son voisinage. Le but de ces modifications locales est d'explorer le voisinage de la solution actuelle afin d'améliorer progressivement sa qualité au cours des différentes itérations. Le voisinage de la solution s englobe l'ensemble des modifications qui peuvent être effectuées sur la solution elle-même. La qualité de la solution finale dépend particulièrement des modifications effectuées par les opérateurs de voisinages. En effet, les mauvaises transformations de la solution initiale mènent la recherche vers la vallée de l'optimum local d'un voisinage donné (peut être un mauvais voisinage) ce qui bloque la re-

cherche en fournissant une solution de qualité insuffisante. De nombreuses méthodes à base de solution unique ont été proposées dans la littérature. Parmi lesquelles : la descente, le recuit simulé, la recherche tabou, la recherche à voisinage variable (VNS : Variable Neighbourhood Search), la recherche locale réitérée (ILS : Iterated Local Search), la recherche locale guidée (GLS : Guided Local Search)... etc.

La recherche locale simple (la descente) La recherche locale simple ou la descente est un algorithme d'amélioration très ancien. Son principe consiste à explorer le voisinage de la solution courante afin d'améliorer sa qualité progressivement, comme le montre la figure 3.1 qui représente un schéma d'évolution d'une recherche locale simple. A chaque itération du processus d'amélioration, l'algorithme modifie un ensemble de composantes de la solution courante pour permettre le déplacement vers une solution voisine de meilleure qualité. Le processus est répété itérativement jusqu'à la satisfaction du critère d'arrêt.

Il est à noter qu'il existe trois types de la descente : la descente déterministe, la descente stochastique et la descente vers le premier meilleur voisin.

L'algorithme 3.1 résume les étapes de l'algorithme général de la descente. L'algorithme entame la recherche par la construction d'une solution initiale s et l'évaluation de sa qualité f(s). Ensuite il commence l'ensemble des étapes du processus d'amélioration suivantes :

• Modifier s pour obtenir une nouvelle solution s' de meilleure qualité que s et qui appartient à son voisinage. Le choix de la fonction de voisinage est important. Il dépend de l'objectif à atteindre. En fait, si l'objectif est de minimiser le coût on doit suivre la direction de la vallée. Sinon (dans le cas de maximisation), on doit suivre la direction du sommet.[13]

• Evaluer la qualité f (s') de la nouvelle solution s' .

• Remplacer s par s' , si s' est de meilleure qualité

Le processus d'amélioration sera répété jusqu'à arriver au cas où toutes les voisines candidates sont de piètre qualité que la solution courante. Autrement dit, la recherche s'arrête lorsqu'un optimum local est atteint.

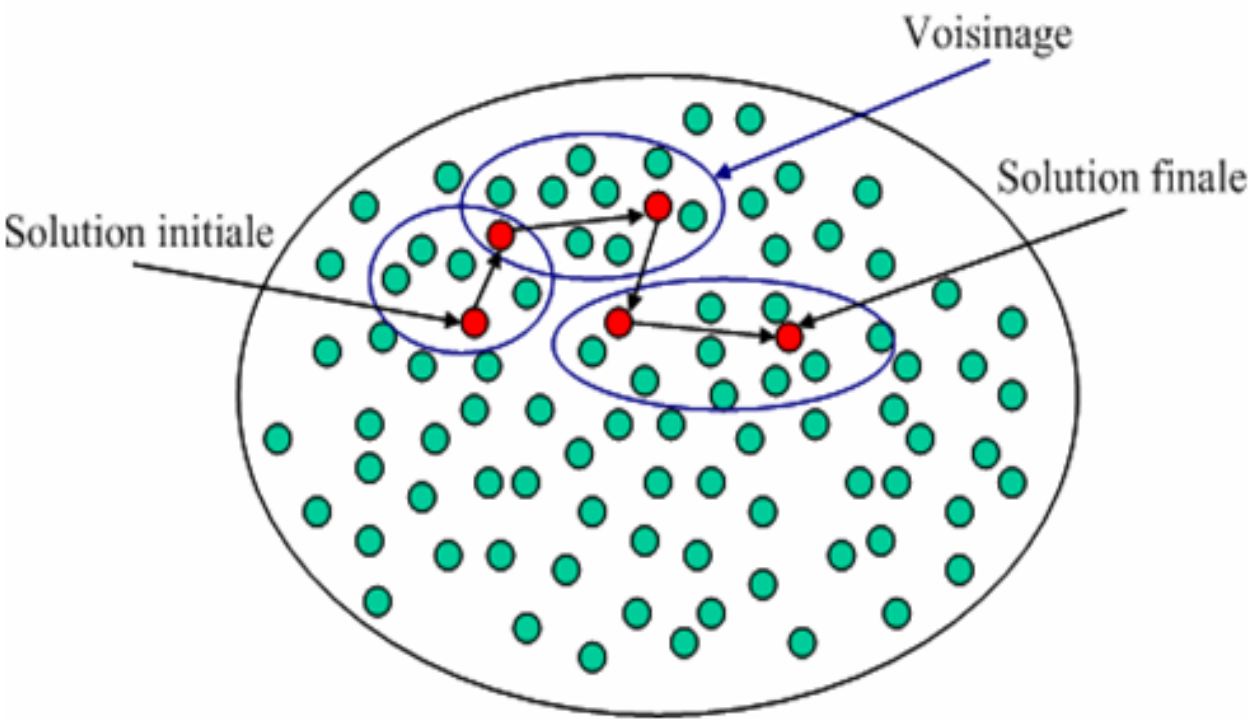


FIGURE 3.2 – Un schéma d'évolution d'une recherche locale simple [13].

L'avantage de la recherche locale simple revient à sa simplicité et à sa rapidité. Cependant, son problème réside dans le fait d'être bloquée par le premier optimum local rencontré. Ce dernier n'est pas forcement l'optimum global. Il peut être très loin de l'optimum global.[13]

Algorithme 3.1 : La recherche locale simple (la descente)

Début
Construire une solution initiale s ;
Calculer la fitness f (s) de s ;
Tant que la condition d'arrêt n'est pas vérifiée **faire**
Modifier s pour obtenir une nouvelle solution voisine s' ;
Calculer f (s') ;
Si f (s') est meilleure que f (s) **alors**
Remplacer s par s' ;
Fin Si
Fin Tant que
Retourner s ;
Fin

La recherche tabou

La recherche tabou (RT) est une métaheuristique à base d'une solution unique. Elle a été proposée en 1986 par Glover [Glover, 1986]. RT est une méthode de recherche locale avancée, elle fait appel à un ensemble de règles et de mécanismes généraux pour guider la recherche de manière intelligente [Glover et Laguna, 1997]. L'optimisation de la solution avec la recherche tabou se base sur deux astuces : l'utilisation de la notion du voisinage et l'utilisation d'une mémoire permettant le guidage intelligent du processus de la recherche. En parcourant le voisinage de la solution courante s, la recherche tabou ne s'arrête pas au premier optimum local rencontré. Elle examine un échantillonnage de solution du voisinage de s et retient toujours la meilleure solution voisine s', même si celleci est de piètre qualité que la solution courante s, afin d'échapper de la vallée de l'optimum local et donner au processus de la recherche d'autres possibi-

lités d'exploration de l'espace de recherche afin de rencontrer l'optimum global. En fait, les solutions de mauvaise qualité peuvent avoir de bons voisinages et donc guider la recherche vers de meilleures solutions. Cependant, cette stratégie peut créer un phénomène de cyclage (i.e. on peut revisiter des solutions déjà parcourues plusieurs fois). Afin de pallier à ce problème, la recherche tabou propose l'utilisation d'une mémoire permettant le stockage des dernières solutions rencontrées pour ne pas les visiter dans les prochaines itérations et tomber dans le problème du cyclage répétitif. Cette mémoire est appelée « la liste tabou », d'où le nom de la métaheuristique tabou. La taille de la liste tabou est limitée, ce qui empêche l'enregistrement de toutes les solutions rencontrées. C'est la raison pour laquelle la liste tabou procède comme une pile FIFO, où la plus ancienne solution sera écartée pour laisser place à la dernière solution rencontrée. Le but de faciliter la gestion de la liste tabou en termes de temps de calcul ou d'espace mémoire nécessaires à pousser les chercheurs à proposer de ne conserver dans la liste que des attributs (i.e. caractéristiques) des solutions au lieu des solutions complètes. Particulièrement, dans le cas où le nombre de variables est très élevé. Toutefois, l'utilisation de la liste tabou peut mener à un blocage dans certains cas. En fait, les nouvelles solutions qui possèdent des attributs des solutions tabou, seront considérées tabou aussi même si elles ne sont pas encore été visitées. Pour remédier à ce problème, on a défini une technique appelée « critère d'aspiration» permettant de sortir du blocage causé par la ressemblance des attributs des solutions tabou. Le critère d'aspiration permet d'omettre le statut tabou sur une solution lorsque certaines circonstances sont respectées. Un critère d'aspiration très classique consiste à autoriser une solution tabou si celle-ci est l'une des meilleures solutions rencontrées depuis le démarrage de la recherche.[13]

Algorithme 3.2 : La recherche tabou

Début
Construire une solution initiale s ;
Calculer la fitness f(s) de s ;
Initialiser une liste tabou vide ;
sbest=s ;
Tant que le critère d'arrêt n'est pas vérifié **faire**
Trouver la meilleure solution s' dans le voisinage de s qui ne soit pas tabou ou qui vérifie le critère d'aspiration ;
Calculer f(s') ;
Si fitness de (s') est meilleure que fitness de (sbest) alors sbest = s' ;
Fin Si
Mettre à jour la liste tabou ; s= s' ;
Fin Tant que
Retourner sbest ;
Fin

La recherche tabou a été largement utilisée pour la résolution de problèmes d'optimisation difficiles comme : le problème du voyageur de commerce, les problèmes du sac à dos, les problèmes de routage, d'ordonnancement, de coloration de graphes, d'exploitation géologique, etc. C'est une méthode facile à mettre en œuvre, rapide, donne souvent de bons résultats et permet de se sauver du premier optimum local rencontré contrairement à la méthode de la recherche locale simple. En revanche, la liste tabou demande de ressources importantes si elle est de grande taille. En fait, elle demande une gestion soigneuse de sa taille et de ce qu'elle doit contenir comme informations sur les solutions trouvées, de manière à ne pas être très exigeante en temps de parcours et d'espace mémoire requis et aussi pour éviter les confusions causées par la ressemblance des attributs des

solutions qui bloquent la recherche. Il est à noter qu'il existe plusieurs variantes de la recherche tabou. Ces variantes dépendent essentiellement du choix du voisinage et de la manière de gérer la liste tabou. L'algorithme 2.3 représente un schéma général de l'algorithme de la recherche tabou.[13]

b) Métaheuristiques à base de population de solutions

Dans cette famille de métaheuristique, le processus de recherche débute avec un ensemble de solutions dit populations, et durant les itérations du processus de recherche ils essayent progressivement d'améliorer leurs qualités afin d'aboutir à des solutions avec de meilleures performances. Ils sont parfois nommés des méthodes évolutives parce qu'elles font évoluer une population d'individus selon des règles bien précises, l'intérêt de cette famille de métaheuristiques est d'utiliser la population comme facteur de diversité pour augmenter la possibilité d'apparition de bonnes solutions en matière de qualité, parmi les métaheuristiques célèbres de cette famille, nous citons : L'algorithme génétique,les algorithmes de colonies de fourmis[5], la recherche coucou (cuckoo search).[13]

L'algorithme génétique

L'algorithme génétique représente une célèbre métaheuristique évolutionnaire. Il a été proposé par Jhon Holland en 1975 [Holland, 1975]. L'algorithme génétique s'inspire des mécanismes biologiques tels que les lois de Mendel et la théorie de l'évolution proposée par Charles Darwin [Darwin, 1859]. Son processus de recherche de solutions à un problème donné imite celui des êtres vivants dans leur évolution. Il utilise le même vocabulaire que celui de la biologie et la génétique classique, on parle donc de : gène, chromosome, individu, population et génération.

-Un gène : est un ensemble de symboles représentant la valeur d'une variable. Dans la plupart des cas, un gène est représenté par un seul symbole (un bit, un entier, un réel ou un caractère).

-Un chromosome : est un ensemble de gènes, présentés dans un ordre donné de manière qui prend en considération les contraintes du problème à traiter.
-Un individu : est composé d'un ou de plusieurs chromosomes. Il représente une solution possible au problème traité.
-Une population : est représentée par un ensemble d'individus (i.e. l'ensemble des solutions du problème).
-Une génération : est une succession d'itérations composées d'un ensemble d'opérations permettant le passage d'une population à une autre.[13]

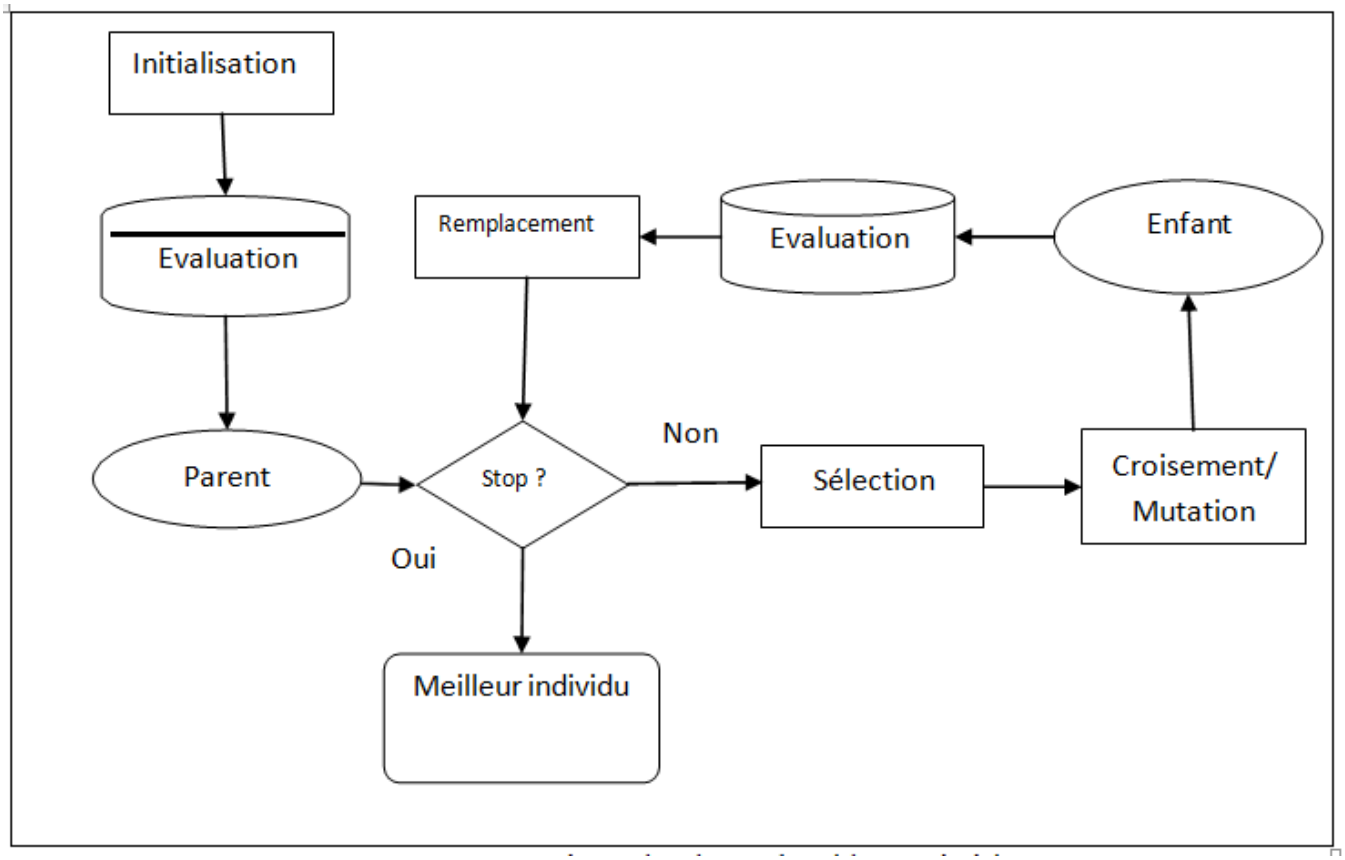


FIGURE 3.3 – Démarche d'un algorithme génétique.[13]

L'algorithme génétique fait évoluer une population composée d'un ensemble d'individus pendant un ensemble de génération jusqu'à ce qu'un critère d'arrêt soit vérifié. Le passage d'une population à une autre est réalisé grâce à des opérations d'évaluation, de sélection, de reproduction (croisement et mutation) et de remplacement. L'algorithme commence la recherche avec un ensemble d'individus. A chaque itération de la procédure de recherche, les meilleurs individus sont sélectionnés pour survivre et se reproduire. La sélection des individus est fondée sur leurs qualités qui sont mesurées à partir d'une fonction appelée « fonction objectif ou fonction fitness ». Ensuite, les individus (appelés parents) sont sélectionnés pour subir des opérateurs de croisement et de mutation permettant la génération d'une autre population d'individus (appelés enfants). Les individus de la nouvelle population seront évalués pour remplacer une partie des individus de la population courante. La figure 2.5 illustre un schéma général des étapes du processus de recherche de l'algorithme génétique.[13] Le processus de recherche de l'algorithme génétique est fondé sur les opérateurs suivants :

-Un opérateur de codage des individus : Il permet la représentation des chromosomes représentant les individus.

-Un opérateur d'initialisation de la population : Il permet la production des individus de la population initiale. Malgré que cet opérateur ne s'intervient qu'une seule fois et au début de la recherche, mais il joue un rôle non négligeable dans la convergence vers l'optimum global. En fait, le choix de la population initiale peut rendre la recherche de la solution optimale du problème traité plus facile et plus rapide.

-Un opérateur de sélection : Il permet de favoriser la reproduction des individus qui ont les meilleures fitness (i.e. les meilleures qualités).

-Un opérateur de croisement : Il permet l'échange des gènes entre parents (deux parent en général), pour créer 1 ou deux enfants en essayant de combiner les bonnes caractéristiques des parents. Le but de cet opérateur est la création de nouveaux individus en exploitant l'espace de recherche.[13]

-Un opérateur de mutation : Il consiste à modifier quelques gènes

des chromosomes des individus, dans le but d'intégrer plus de diversité au sein du processus de la recherche.[13]

- Un opérateur d'évaluation : Il permet de valoriser la qualité des individus, en se basant sur la fonction «objectif» (la fonction fitness) qui permet de calculer la qualité de chaque individu. En outre des différents opérateurs permettant de guider la recherche par l'algorithme génétique, ce dernier nécessite un certain nombre de paramètres de base, sur lesquels dépendent les différents opérateurs cités en dessus. Ces paramètres doivent être fixés à l'avance, ils jouent un rôle très important dans la performance de l'algorithme. On parle de : la taille de la population, la probabilité de croisement, la probabilité de mutation et le nombre maximum de génération.[13]

- La taille de la population : Elle représente le nombre d'individus de la population. S'il est trop grand, le processus de recherche demande un coût de recherche élevé, que se soit en termes d'espace mémoire ou du temps de calcul nécessaires. Cependant, s'il est trop petit, l'algorithme risque d'être tomber dans le cas de la convergence prématurée à cause du manque de la diversité au sein de la population. Il est préférable donc de choisir une taille moyenne en prenant en considération l'instance du problème à traiter.[13]

- La probabilité de croisement : Elle représente la probabilité d'échange de patrimoine (i.e. les gènes) entre deux individus (ou plus). Plus elle est grande, plus elle permet la génération de nouveaux enfants qui peuvent être meilleurs que leurs parents.[13]

- La probabilité de mutation : Elle est en général faible, dans le but d'échapper aux possibilités de modifications radicales des solutions, particulièrement, des solutions de bonnes qualités qui ne nécessitent que peu d'amélioration pour passer aux solutions optimales.

- Le nombre maximum de génération : Ce paramètre peut jouer le rôle d'un critère d'arrêt. Il peut construire un obstacle pour l'algorithme. En fait, il peut empêcher les différents opérateurs d'aboutir à la meilleure solution s'il est trop petit. Comme il peut engendrer un temps de calcul prohibitif dans le cas ou il est trop grand. Ainsi, le choix de sa valeur peut

se baser sur des tests préliminaires.[13]

L'algorithme 3.3 suivant représente l'ensemble des étapes de l'algorithme génétique.

Algorithme 3.3 : algorithme génétique

Début
Initialiser les paramètres nécessaires ;
initialiser une population de N individus ;
Evaluer les N individus ;

Tant que la condition d'arrêt n'est pas satisfaite **faire**
Utiliser l'opérateur de sélection pour sélectionner K individus ;
Appliquer l'opérateur de croisement sur les K individus avec la probabilité Pc ;
Appliquer l'opérateur de mutation sur les individus avec la probabilité Pm ;
Utiliser l'opérateur d'évaluation pour évaluer les enfants obtenus ;
Utiliser l'opérateur de sélection pour remplacer quelques individus parents par des individus enfants ;
Fin Tant que
Retourner la ou les meilleures solutions ;
Fin

L'algorithme génétique a été utilisé pour la résolution de nombreux problèmes académiques et/ou industriels. Son avantage principal est qu'il permet une bonne combinaison entre l'exploitation de solutions et l'exploration de l'espace de recherche. Cela est établi en fonction des opérateurs de croisement et de mutation respectivement. Cependant, son inconvé-

nient réside dans deux points : un temps de calcul assez important pour pouvoir converger vers la solution optimale (quoique l'apparition des ordinateurs de plus en plus puissants a permis de remédier à ce problème). Et le nombre de paramètres importants (taille de la population, paramètres de sélection, paramètres de croisement, paramètres de mutation, critère d'arrêt...).[13]

L'algorithme de colonies de fourmis

Les algorithmes de colonies de fourmis ont été proposés par Marco Dorigo dans sa thèse de doctorat en en 1992 . L'idée de base imite le comportement collectif des fourmis lors de leur déplacement entre la fourmilière et la source de nourriture dans l'objectif de rechercher la nourriture en parcourant le plus court chemin, Quand une fourmi atteint la source de nourriture, elle retrace son chemin de retour vers le nid par une quantité de phéromone pour avertir les autres fourmis qu'il y a de la nourriture à la fin du chemin. Les fourmis les plus rapidement arrivées au nid après avoir visité la source de nourriture sont celles qui empruntent le chemin le plus court, aussi la quantité de phéromone présente sur le plus court chemin est plus importante que celle présente sur le chemin le plus long. Une piste qui présente une plus grande concentration en phéromone est plus attirante par les fourmis, elle a une probabilité plus grande d'être empruntée. La piste courte va alors être plus renforcée que la plus longue, et elle sera choisie par la quasi-totalité des fourmis. L'idée est de représenter le problème à résoudre sous la forme de la recherche d'un meilleur chemin .[19]

L'algorithme de colonies de fourmis a été proposé pour la première fois pour résoudre le problème du voyageur de commerce , il se base sur trois phases essentielles :[13]

Algorithme 3.4 : L'algorithme de colonies de fourmis pour le TSP

Début
Initialiser une population de m fourmis ; Evaluer les m fourmis ;
Tant que la condition d'arrêt n'est pas satisfaite **faire**
Pour i=1 à m faire
Construire le trajet de la fourmi i ;
Déposer des phéromones sur le trajet de la fourmi i ;
Fin pour
Evaluer les m fourmis ;
Evaporer les pistes de phéromones ;
Fin Tant que
Retourner la ou les meilleures solutions ;
Fin

- La construction du trajet de chaque fourmi.
- La distribution de phéromones sur le trajet de chaque fourmi.
- Evaporation des pistes de phéromones.

L'algorithme 3.4 représente le schéma général de l'algorithme de colonies de fourmis pour le problème du voyageur de commerce (TSP : Traveling Salesman Problem). Il est à noter qu'à part le problème du voyageur de commerce, l'algorithme de colonies de fourmis a été appliqué avec succès sur d'autres problèmes d'optimisation comme : les problèmes des tournées de véhicules, le problème d'affectation quadratique. . . etc.[13]

La recherche coucou(cuckoo search)

« cette algorithme est notre approche implémenter »

La recherche coucou (CS : Cuckoo Search) a été proposée en 2009 par Yang et Deb [Yang et Deb, 2009 ; Yang et Deb, 2010]. La recherche coucou

s'inspire du comportement de reproduction d'une espèce spéciale d'oiseaux parasites de nids appelés « Coucous ». Dans l'algorithme de la recherche coucou, une solution possible est appelée « nid » ou « coucou ». En fait, la recherche coucou part du principe que chaque nid comporte un seul coucou. Au cours du processus de la recherche de l'algorithme CS, chaque coucou crée son propre poussin en fonction de sa représentation actuelle (le coucou lui même) . L'évaluation de la qualité du poussin et de son père permet de sélectionner lequel d'entre eux survivra et subira quelques modifications dans le but d'améliorer sa qualité. La figure 3.4 représente un organigramme résumant les étapes de l'algorithme de la recherche coucou .[13]

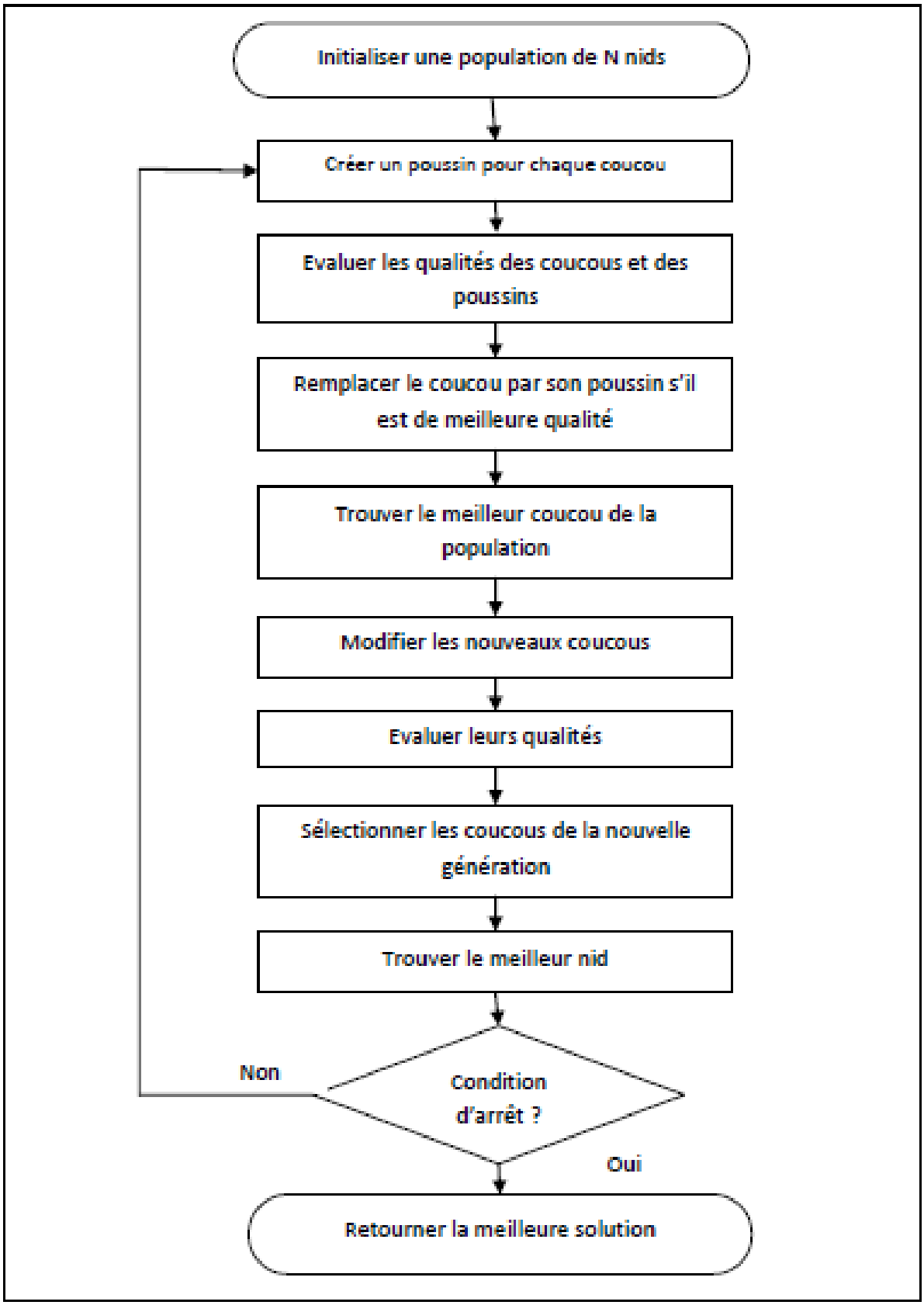


FIGURE 3.4 – Organigramme de l'algorithme de la recherche coucou[13] .

a) Comportement de reproduction de coucou :

Les coucous sont des oiseaux fascinants, non seulement à cause des beaux

sons qu'ils peuvent faire, mais aussi à cause de leur stratégie de reproduction agressive. Certaines espèces comme les coucous ani et guira pondent leurs oeufs dans des nids communs, bien qu'ils puissent retirer les oeufs d'autrui pour augmenter la probabilité de leurs propres oeufs. Il s'agit du parasitisme obligatoire couvé (parasitisme de ponte) qui consiste à pondre des oeufs dans les nids d'autres oiseaux hôtes (souvent d'autres espèces). Certains oiseaux hôtes peuvent engager un conflit direct avec les coucous intrus. Si un oiseau hôte découvre que les oeufs ne sont pas les leurs, ils vont soit jeter ces oeufs exotiques ou tout simplement abandonner son nid et construire un nouveau nid ailleurs.
En outre, le moment de la ponte de certaines espèces est également incroyable. Les coucous parasites choisissent souvent un nid où un oiseau hôte vient de pondre ses propres oeufs. En général, les oeufs de coucou éclosent légèrement plus tôt que les oeufs des hôtes. Une fois le premier poussin de coucou est éclos, la première action instinctive à prendre est d'expulser les oeufs de son oiseau hôte en les propulsant aveuglément hors du nid, ce qui augmente la part de nourriture du poussin coucou fournie par son oiseau hôte. Des études montrent également qu'un poussin de coucou peut aussi imiter l'appel des poussins hôtes pour avoir accès à plus de nourriture. [19]

b) cuckoo search et Levy Flights :

Le vol Levy a été introduit par le mathématicien français Paul Levy en 1937. Il s'agit d'un algorithme de marche aléatoire dans lequel les étapes sont définies en termes de longueurs d'étapes qui suivent une certaine distribution de probabilité dans laquelle les directions des étapes doivent être aléatoires.
L'étude de Reynolds et Fyre [19] sur les vols Lévy indique que la plupart des animaux et des insectes explorent leurs paysages à l'aide d'une série de trajectoires rectilignes suivies d'un virage à 90°. Ce type de comportement s'appelle le modèle de recherche sans échelle intermittente de type Lévy-flight.
Le cuckoo search et à partir d'un emplacement bien connu, il générera une toute nouvelle génération à des distances réparties de manière aléatoire en

fonction des vols Levy. Ensuite, la nouvelle génération sera évaluée pour sélectionner la plus prometteuse. Le processus est répété jusqu'à ce que les critères d'arrêt soient satisfaits .[19]

c) Représentation de l'algorithme :

Pour simplifier la description de la recherche de coucous, il y a trois règles idéalisées suivantes :

• Règle 1 : chaque coucou pond un oeuf à la fois et le déverse dans un nid choisi au hasard.

• Règle 2 : les meilleurs nids avec des oeufs de haute qualité seront conservés aux générations suivantes.

• Règle 3 : le nombre de nids hôtes disponibles est fixe et l'oeuf pondu par un coucou est découvert par l'oiseau hôte avec une probabilité $pa \in [0, 1]$. Dans ce cas, l'oiseau hôte peut jeter l'oeuf ou abandonner le nid et construire un nid complètement nouveau.

Cette dernière hypothèse peut être approchée par la fraction pa des n nids remplacés par de nouveaux nids (avec de nouvelles solutions aléatoires). De ce fait, un nid dans le cuckoo search est un individu de la population telle que le nombre de nids est fixé (égal à la taille de la population). Si le nid est abandonné, il faut le remplacer par un nouveau. Un oeuf dans un nid représente une solution adoptée par un individu de la population. Chaque oeuf du coucou est une nouvelle solution candidate pour une place dans la population Pour un problème de maximisation, la qualité d'une solution peut simplement être proportionnelle à la valeur de la fonction objectif.(on dit toujours fonction objectif). Sur la base de ces trois règles, les étapes de base de la recherche coucou peuvent être résumées sous la forme du pseudo-code suivant :

Algorithme 3.5 : algorithme cuckoo search

Debut
Initialiser une population de n nids d'accueil xi (I = 1,2,...,n)
Tant que (t < NbrMaxGénération) **fair**
Choisir une solution j au hasard
et générer une solution par vol de Lévy
Evaluation de la fitness fi de la solution
si (Fi >Fj) **alor**

Remplacer j par la nouvelle solution
Fin
On classe les solutions par leur fitness et on trouve les meilleures solutions courante.
la fraction pa des moins bons solutions sont abandonnés et une quantité équivalente de nouvelles solutions est générée par le vol de Lévy.
Les meilleurs solutions passent a l'itération suivante.

Fin Tant que
Fin

L'algorithme suivant présente une partie de notre implémentation décrivant la marche de lévy.

Algorithme 3.6 : La méthode lévy flight :

Debut
Variables
x, s, u, v, step, stepsize, best, sigma : double ;
rand :random ;
beta 1.5 ;
calculgama(x) ;
calcullogdegama(x) ;
sigma=Math.pow((gamma(1+beta)*Math.sin(3.1415*beta/2.0)/(gamma(
(1.0+beta)/2.0)*beta*Math.pow(2.0,(beta1.0)/2.0))),(1.0/beta)) ;
u = (rndm.nextGaussian()) * sigma ;
v = (rndm.nextGaussian()) ;
step = u/Math.pow((Math.abs(v)), (1/beta)) ;
stepsize = 0.01 * step * (s best) ;
s = s + stepsize * rand.nextGaussian() ;
Si (s>1||s<-1) alor
s = levy() ;
Fin
récupérer la valeur de s ;
Fin

les fonctions logGamma et Gama sont calculées comme suit :

La méthode logGamma : Variables
x, tmp, ser,resultat : double ;
tmp =(x0.5) *Math.log(x + 4.5) (x + 4.5) ;
ser=(1.0 + 76.18009173 / (x+0) 86.50532033 / (x+1)+24.01409822/
(x+2)1.231739516/(x+3)+0.00120858003/(x+4)0.00000536382/(x+5) ;
resultat=(tmp +Math.log(ser*Math.sqrt(2*Math.P I))) ;

La méthode Gamma :
Variables
R, x : double ;
R = Math.exp(logGamma(x)) ;

3.6 Conclusion

Dans ce chapitre, nous avons présenté quelques notions fondamentales sur les métaheuristiques, et les classifications des métaheuristiques en se basant soit sur la fonction objectif ou bien sur le nombre des solutions ou suivant la source d'inspiration.
Enfin nous sommes concentré sur la métaheuristique cuckoo search afin de l'utiliser dans le cadre de notre projet de fin d'études. Cette métaheuristique a été appliquée dans pas mal des problèmes au cours de ces dernières années. Son algorithme est très simple, il donne de bons résultats et peut facilement s'adapter à des problèmes d'optimisation multi-objectifs dans les Clouds.
Dans le chapitre suivant, nous allons proposer une nouvelle approche d'ordonnancement des workflows basées sur la métaheuristique cuckoo search, et le langage java, l'environnement de développement Netbeans. Ensuite le simulateur CloudSim, son architecture, les classes fondamentales.

CHAPITRE 4

IMPLEMENTATION

SOMMAIRE

4.1 Introduction

L'objectif de ce chapitre est de décrire en détail notre contribution dans le cadre de ce projet de fin d'études. Cette contribution consiste à utiliser un algorithme basé sur la métaheuristique cucko search pour l'ordonnancement des workflows multiobjectifs dans le Cloud Computing par rapport à trois critères qui sont : le temp d' exécution , le coût, la consommation d'énergie.

Dans ce chapitre, nous allons présenter le langage java, l'environnement de développement Netbeans, Ensuite le simulateur CloudSim, son architecture, les classes fondamentales.

Par la suite, nous expliquerons notre approche et nous citerons aussi les différentes étapes et les configurations nécessaires avant le lancement de la simulation. Nous conclurons le chapitre par une analyse des résultats.

4.2 Langage et environnement de développement

Ce projet a été implémenté et simulé dans un environnement qui possède les caractéristiques suivantes :

• Une machine avec un processeur intel(R) core i5 3337U, (CPU) une vitesse de 1.80 GHZ avec une capacité de mémoire de 4GB. sous le système d'exploitation windows7 de 64bits.

- L'IDE Netbeans

• Le simulateur CloudSim version 3.0.3 développée avec langage de programmation java.

4.2.1 Langage de programmation Java

Java est un langage de programmation orientée objet. Il a été crée par James Gosling et Patrick Naughon, deux employés de Sun micro système avec la collaboration de Bill joy(Co- fondateur de Sun Microsystems en 1982), présenter pour la première fois le 23 mai 1995 au Sun World.[17]

La particularité principale de langage Java est que, les logiciels écrits avec ce dernier sont facilement portables sur plusieurs systèmes d'exploitation tels que : Unix, Microsoft Windows, Mac OS ou Linux.

Netbeans

NetBeans est un environnement de développement intégré, placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres par l'ajout de greffons [20]

4.2.2 CloudSim

C'est un outil de simulation extensible complet pour la modélisation et simulation du Cloud Computing. Il permet d'étendre et de définir des politiques pour tous les systèmes Composants. Il prend en charge la modélisation du système et du comportement comme les centres de données, les machines virtuelles et l'approvisionnement des ressources. Il est considéré comme l'outil de simulation cloud le plus populaire.[10]

Le simulateur CloudSim comprend plusieurs classes, qui peuvent être classées en deux catégories : une catégorie qui contient des classes qui modélisent les entités du système Cloud comme les centres de données, le

courtier (broker), les machines physiques (Hosts), les machines virtuelles (Vms), et une autre catégorie qui contient des classes qui modélisent la politique d'ordonnancement. Nous allons détailler quelques classes de base de Cloud

classes de CloudSim :

Le simulateur ClouSim est composé de plusieurs classes (figure 4.1). Parmi les classes fondamentales qui forment les blocs constitutifs du simulateur CloudSim, nous pouvons citer :

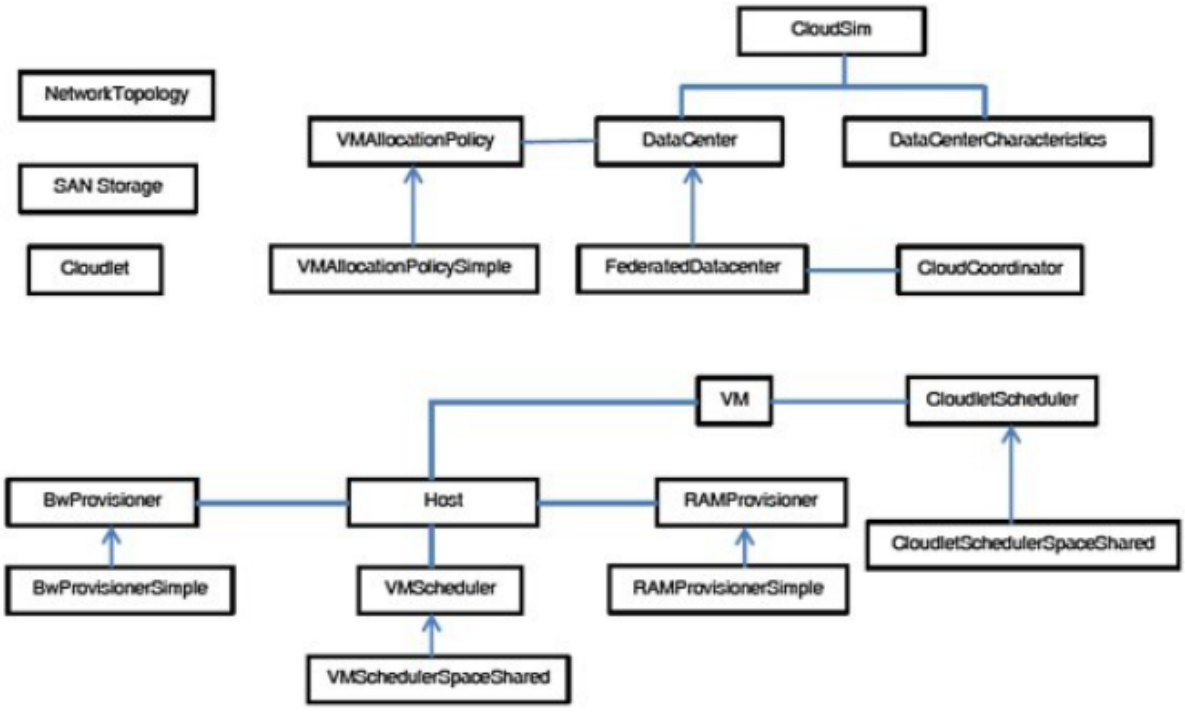


FIGURE 4.1 – Diagramme de classe toolkit CloudSim .[5]

1- Classe Datacenter : Cette classe modélise l'infrastructure qui représente le matériel (hôtes) fourni par le fournisseur cloud sous forme de service. Il existe aussi la classe PowerDatacenter qui hérite de la classe datacenter, elle est dans le package Power de CloudSim, l'avantage est qu'elle permet une simulation large qui comprend le calcul de l'énergie consommée.[5]

2- Classe DatacenterBroker : Cette classe modélise le broker (courtier), qui est l'intermédiaire entre l'utilisateur et le fournisseur de services cloud. Le broker réagit comme un négociateur pour garantir une meilleure allocation qui satisfait la qualité de service de l'utilisateur. Pour cela il se base sur les CIS (Cloud Information Service) qui contient les caractéristiques des datacenters fournis par le fournisseur.[5]

3- Classe HOST : Cette classe modélise les ressources physiques, elle comprend des informations essentielles concernant la mémoire, la bande passante, CPU. Elle garantit une politique d'allocation des ressources (mémoire, stockage, bande passante) aux machines virtuelles, il y a aussi la classe PowerHost qui hérite de cette classe qui permet une simulation qui prend en charge la notion de l'énergie pour les hôtes.[5]

4- Classe VM : Cette classe modélise les machines virtuelles qui sont hébergées aux niveaux des machines physiques. Elle comprend aussi les caractéristiques liées aux machines virtuelles tels que la mémoire, la bande passante, CPU...[5]

5- Cloudlet (Tache) : Cette classe modélise les tâches qui seront assignées aux machines virtuelles pour un traitement bien défini (traitement ou bien stockage).[5]

Architecture de CloudSim :

La structure logicielle de CloudSim et ses composants sont représentés par une architecture en couches comme indiqué dans la figure 4.2 Les

premières versions de CloudSim utilisent SimJava, un moteur de simulation d'événement discret qui met en oeuvre les principales fonctionnalités requises pour des structures de simulation de haut niveau. Parmi les fonctionnalités, nous avons la formation d'une file d'attente et le traitement d'événements, la création de composants système (les services, les machines physiques (Host), le centre de données (Data Center), le courtier (Broker), les machines virtuelles), la communication entre les composants et la gestion de l'horloge de simulation.[5]

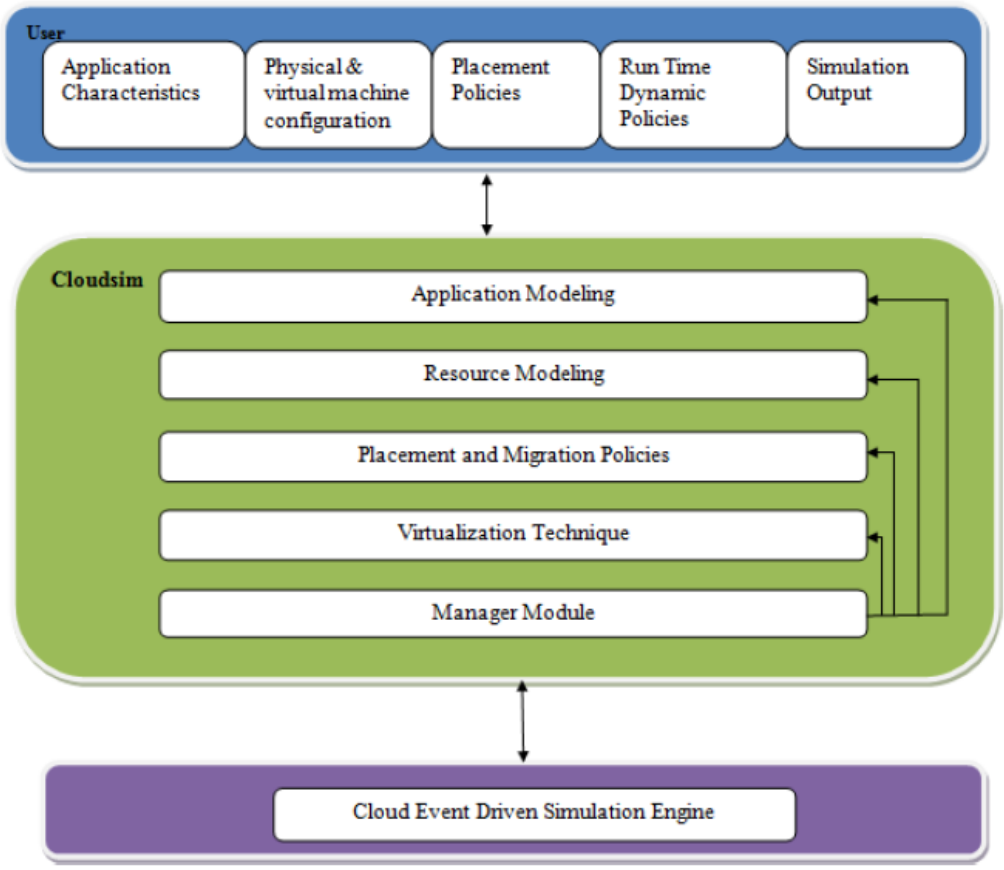


FIGURE 4.2 – Architecture de toolkit CloudSim .[5]

Politiques d'ordonnancement :

Le simulateur CloudSim fournit deux politiques d'ordonnancement : 1- La politique d'ordonnancement SpaceShared (Espace partagé) : Dans cette politique d'ordonnancement, le broker (ordonnanceur) planifie une seule tâche sur une machine virtuelle à un instant donné, et après avoir terminé l'exécution de la tâche, il lance une autre tâche sur la même machine

virtuelle. La politique espace partagé suit la même procédure que l'algorithme premier arrivé premier servi. Cette politique est aussi utilisée pour l'affectation des machines virtuelles aux machines physiques (host).[15]

2- La politique d'ordonnancement TimeShared (Temps partagé) : Dans cette politique d'ordonnancement, le broker planifie plusieurs tâches sur la même machine virtuelle. Il partage le temps entre toutes les tâches sur la même machine virtuelle simultanément. Cette politique est aussi utilisée pour l'affectation des machines virtuelles aux machines physiques (hosts) [15]

Mode de fonctionnement

Le fonctionnement de CloudSim se fait comme suit : la création des machines virtuelles au niveau des machines physiques qui sont hébergées par le datacenter. À son tour, il envoie ses caractéristiques (nombre d'hôtes, machines virtuelles...) à un registre de Cloud information Service (CIS). À l'arrivée des tâches qui sont envoyées par un utilisateur au broker, il récupère les caractéristiques des datacenters de CIS, il agit comme un négociateur afin de choisir les machines virtuelles qui satisfait les exigences de l'utilisateur.

4.3 Approche proposée

4.3.1 Objectif du travail

L'objective de notre projet, est de proposer un mécanisme d'ordonnancement des workflows dans le Cloud Computing, nous avons proposé une approche qui consiste à gérer les machines virtuelles au sein d'un système Cloud Computing de manière optimale . Cette approche est basée sur la métaheuristique «cuckoo search», qui prend en considération les trois

critères suivants : temps d'exécution , consommation d'énergie, coût de traitement (cost).

4.3.2 Modèle d'ordonnancement

Pour un fournisseur d'une infrastructure IaaS offrant un ensemble de machines virtuelles VM, un workflows d'un utilisateur composé d'un ensemble T de n tâches qui doivent être exécutées sur ces VM. Le problème d'ordonnancement des workflows sur l'IaaS revient à construire un mapping M des tâches aux VM , qui optimise les métriques conflictuelles suivantes : le temp d'exécution, le coût,l'energie consommée.
par conséquent, le problème d'ordonnancement des workflows peut être formulé comme le problème d'optimisation multi objectif suivant :
Minimiser temp d'exécution Minimiser Coût de traitement Minimiser Consommation d'énergie

Etape d'ordonnancement

Cette étape décrit la stratégie d'ordonnancement que nous avons proposée ainsi que les outils utilisés, notamment la fonction objective, critères d'ordonnancement , on a utilisé La politique d'ordonnancement SpaceShared.
Notre démarche d'ordonnancement se fait selon les étapes suivantes :

- Temps d'exécution : Représente le temps nécessaire pour une tâche pour qu'elle s'exécute au niveau de la machine virtuelle.Nous utilisons cette formule :

$$LeTempsd'exécution = \frac{Cloudletlength}{(VMMIPS * VMPEsNumber)}$$

Où :

— Cloudletlength : signifie la taille de la tâche.
— VM PEsNumber : signifie le nombre de coeur de CPU occupé par la machine virtuelle.
— VM MIPS : est la vitesse du processeur de la machine virtuelle

• Makespan : L'une des mesures les plus utilisées dans l'évaluation des algorithmes d'ordonnancement des workflows est le temps de complétion ou makespan. Le makespan est la différence entre la date de soumission de la tâche et la date de réception des résultats .[12]

Le makespan est donné dans l'équation suivante :
Makespan = max(temps d'excution(i))

— Temps d'exécution : Représente le temps nécessaire pour une tâche pour qu'elle s'exécute au niveau de la machine virtuelle.

• Coût de traitement : Représente le coût nécessaire pour qu'une tâche s'exécute au niveau de la machine virtuelle. [27]

Coût de traitement = (Prix * Temps d'exécution du tache)

- Le prix d'utilisation de CPU par second. Il est fixer a 3.0
- Temps d'exécution signifie le temps d'exécution d'une tâche.

• La consommation d'énergie : Représente l'énergie consommée au niveau de la machine virtuelle pour faire un traitement d'une tâche. [31]

consommation d'énergie =(maxPower/psize)*cpuutil*(c.getFinishTime()-c.getExecStartTime())

- maxpower : la puissance maximale des serveurs qui se produit lorsque la charge de travail du centre de données est au maximum.
- psize : nombre de cpu au niveau de la machine.
- cpuutil est le prosésseur utilisé : cpuutile = v.getMips()/hostmips
- c.getFinishTime() : le debut de temp d'exécution de la tache (i)
c.getExecStartTime() : la fin de temp d'exécution de la tache (i)

Fonction objectif

Nous allons utiliser une approche qui consiste à faire une somme pondérée des fonctions présentées précédemment où chaque fonction est associée à un poids selon son importance. La somme des poids doit être égale à 1. Cette somme représente la fonction objectif (fitness) de notre métaheuristique. Dans nos simulations, nous avons utilisé p1=0.4 pour la consommation d'énergie , p2=0.25 pour le coût et p3=0.35 pour le temps d'exécution. L'algorithme de la fonction objective est le suivant :

Algorithme 4.1 :fonction objectif

entries :F,poid1,poid2,poid3 : Réel

$$F = \left\{ \begin{array}{l} Poid1 * cirière1+ \\ Poid2 * cirière2+ \\ Poid3 * cirière3+ \end{array} \right\}$$

où p1=0.4(consommation d'énergie), p2=0.25 (coût)
et p3=0.35 (temps d'exécution)

L'ordonnancement des workflows

Cette étape consiste à faire une présimulation,afin de trouver les meilleures affectations des workflows aux machines virtuelles et les envoyer au broker.L'algorithme ci-dessous présente notre travail(approche proposée) :

Algorithme 4.2 :Algorithme d'ordonnancement.

Debut
entries :T :liste des taches, Vsorted : liste des Vms, F : liste vide.
sorties : liste des affectation des taches aux machines virtuelles.
pour(chaque tache dans la liste des taches)
pour (i=0 a 6% $\in Vsorted$)$choisirVmiauhasard$
Évaluation(tâche,Vmi)
ajouter Fi dans la liste F
fin pour
pour (i=0 a 10% $\in Vsorted$)$choisirVmjparlevoldelévy$
Évaluation(tâche,Vmj).
ajouter Fi dans la liste F.
triée la liste F et récupérer le grand Fi.
Si (Fj < u.Fworst)
remplacer la Vmi par la nouvelle solution Vmj.
fin si
fin pour
fin pour
Triées F et choisir la meilleur solution.
Affecter la tâche a la meilleur Vm.
fin

4.4 Implémentation

1- La fenetre principale de l'application : la fenetre principale de notre application voir (figure 4.3) ,l'utilisateur doit remplir les informations du cloud pour la création de nombre de datacenter , nombre de Hosts , nombre de VMs ,et nombre des Taches.

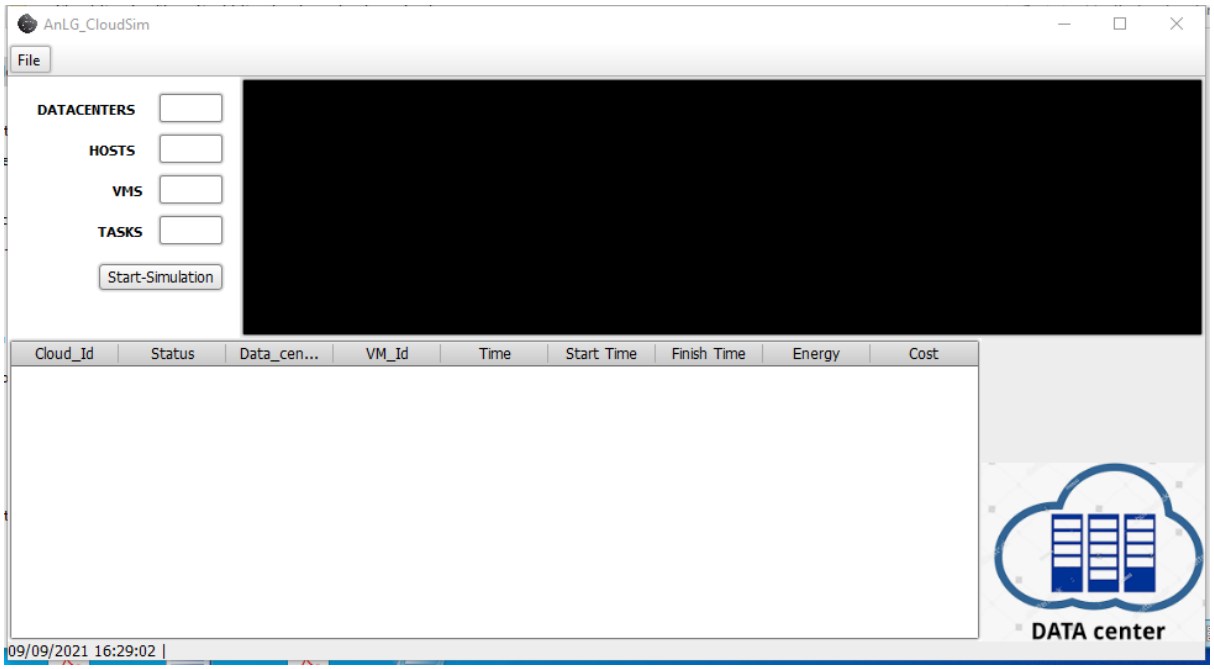


FIGURE 4.3 – Caractéristiques de fenetre pricipale

2- La configuration des Hotes

Pour la création des hotes, le simulateur ClouSim à besoin de la configuration des hôtes (voir figure 4.4), l'utilisateur remplit les informations suivantes : , le nombre des coeurs de cpu, la vitesse de cpu en MIPS, la ram en MB, stockage en MB, la bande passante en Mbit/s. Puis pour la création, il clique sur « Save ».

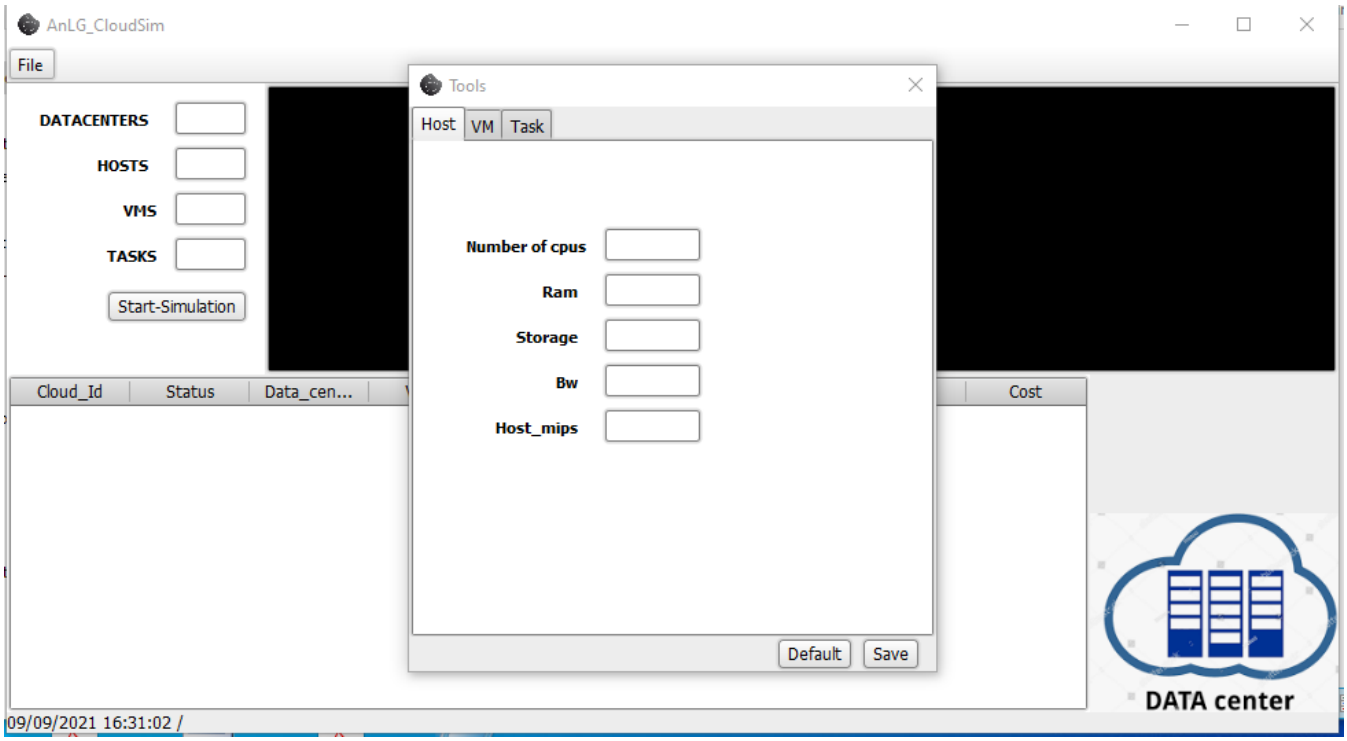


FIGURE 4.4 – Caractéristiques de machines physiques(Host)

3-La configuration des VMs

En glissant sur «Vm» (voir figure 4.5) qui permet de configurer les vms en remplissant les informations suivantes : nombre de coeurs cpu de vm, la vitesse de cpu en mips, la ram, stockage, la bande passante. Pour la création des Vms, il faut cliquer sur «Save».

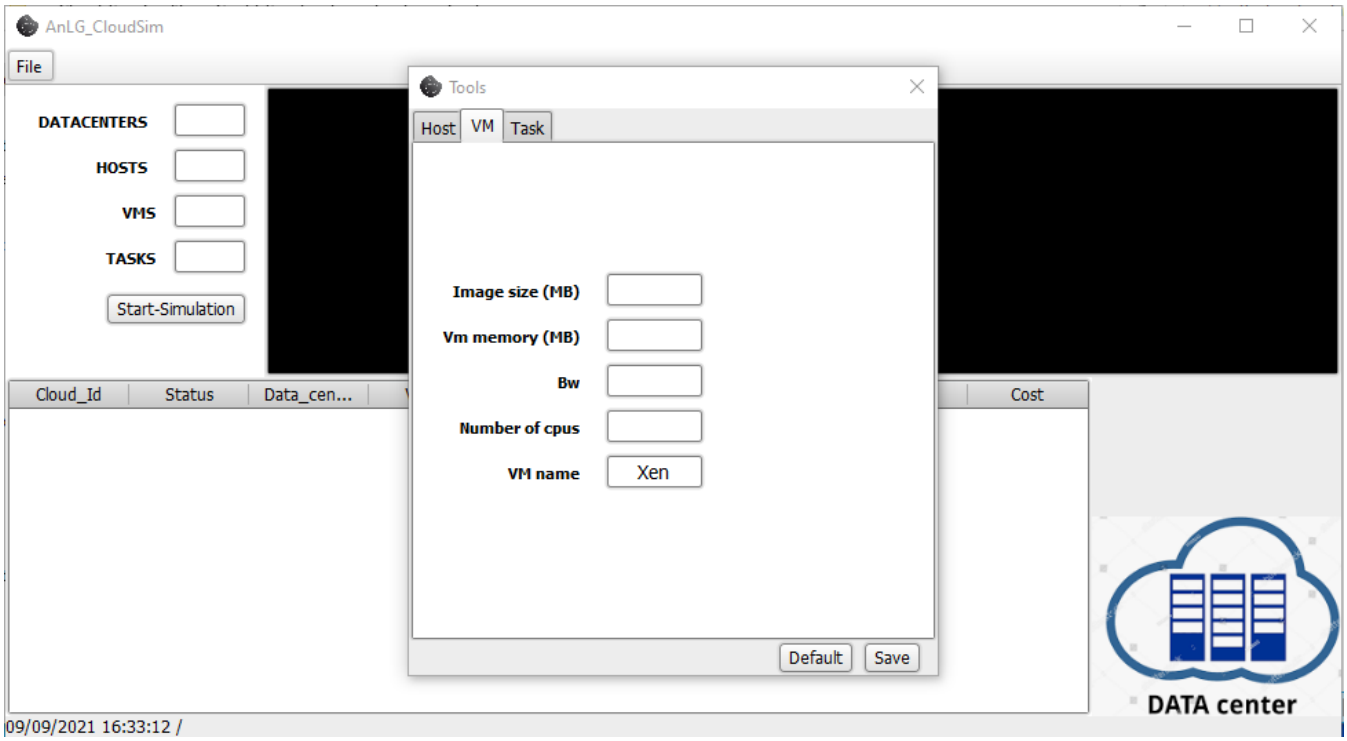


FIGURE 4.5 – Caractéristiques de machines virtuelles

4 - La configuration des tâches

Le bouton Task permet la configuration des tâches, pour cela comme c'est illustré dans la figure 4.6, en remplissant les champs suivants : le nombre de tâches, la taille de la tâche, fichier de la tâche en entrée . Pour la création des tâches, il faut cliquer sur «Save».

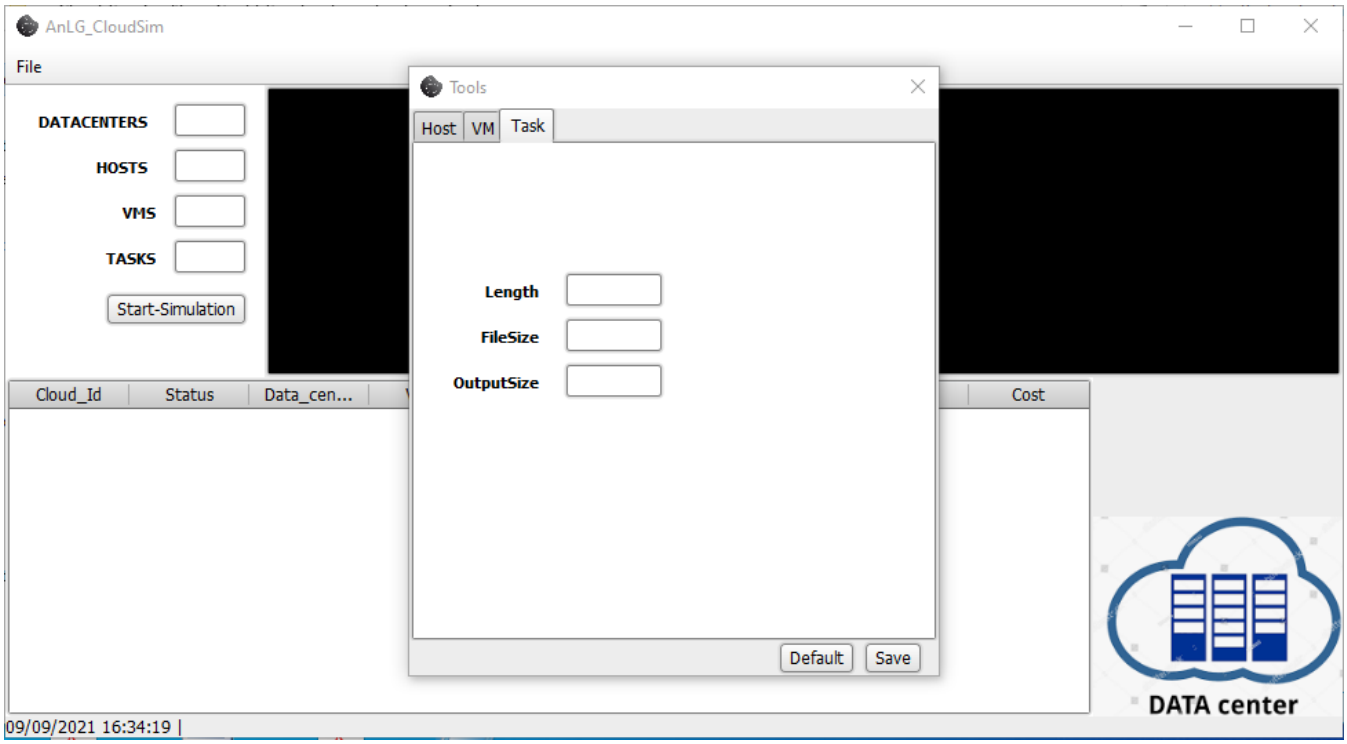


FIGURE 4.6 – Caractéristiques des tâches

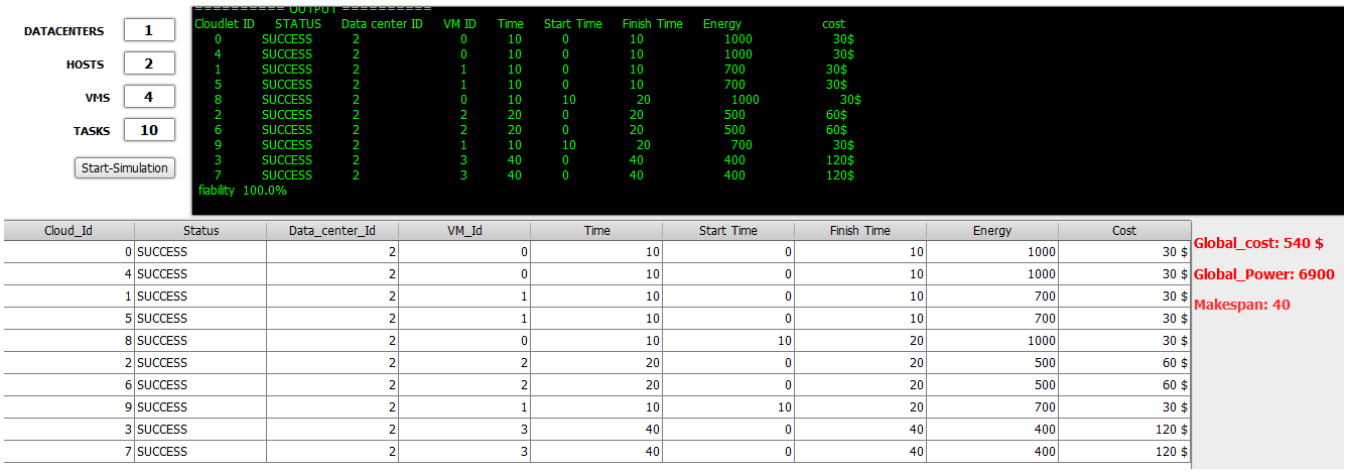
5 - Affichage des résultats)

dans cette affichage en lance un workflow de 10 taches avec les configurations suivantes :

- (1) Datacenter
- (2) Hotes
- (4) VMs
- un workflow de 10 taches.(Application de 10 taches)

D'aprés la simulation, le résultat est affiché dans la console de notre applicatio (AnLG) qui contient les informations suivantes : l'ID de la tâche, ID de datacenter, ID de Vm, le statut de la tâche, le temps d'exécution de la tâche et le temps de début et la fin de la tâche, énergie consommée

par la tâche, le coût de traitement de la tâche, le coût total des tâches, énergie consommée totale des tâches et le makespan.



Cloud_Id	Status	Data_center_Id	VM_Id	Time	Start Time	Finish Time	Energy	Cost
0	SUCCESS	2	0	10	0	10	1000	30 $
4	SUCCESS	2	0	10	0	10	1000	30 $
1	SUCCESS	2	1	10	0	10	700	30 $
5	SUCCESS	2	1	10	0	10	700	30 $
8	SUCCESS	2	0	10	10	20	1000	30 $
2	SUCCESS	2	2	20	0	20	500	60 $
6	SUCCESS	2	2	20	0	20	500	60 $
9	SUCCESS	2	1	10	10	20	700	30 $
3	SUCCESS	2	3	40	0	40	400	120 $
7	SUCCESS	2	3	40	0	40	400	120 $

Global_cost: 540 $
Global_Power: 6900
Makespan: 40

FIGURE 4.7 – Résultat de la simulation

Simulations

nous avons comparé l'approche proposé avec l'algorithme RR (Round Robin) qui est deja definie dans le chapitre 2 (Ordonnancement des Taches et workflows).
on a pas un code source d'un autre algorithme d'ordonnancement des workflows pour le comparé avec notre approche proposé.

Cet algorithme(RR) est implémenté dans le simulateur CloudSim (par défaut).
nous avons mesuré le temps d'exécution, le coût et la consommation d'énergie.
voici les configuration du cloud utilisé : 4 DataCenters contenant 4 hotes et 15 VMs et pour les taches (100 et 500 et 1000 taches).

les 15 hotes sont configurés comme indiquer dans le tableau 4.1 avec

une vitesse en MIPS =2500.

PES	Bande passante	RAM	Stockage
4	10000 Bit/s	4 GB	1000000 MB

TABLE 4.1 – Les Les paramètres des Host.

Le nombre de VM est fixé à 15, avec une vitesse en MIPS =2500.

Machine virtuelle	PES	Bande passante	RAM	Stockage
	1	1000 Bit/s	512 MB	10000 MB

TABLE 4.2 – Les Les paramètres des VMs.

On a les memes paramètres de Hotes et Vms et les taches pour les deux simulations avec notre approche proposé et Round Robin (RR).
Les simulations consiste à varier le nombre des taches (100, 500, 1000) avec les paramètres montrés dans le tableau suivant :

taches	Longueur	Taille du fichier d'entrée	Taille du fichier de sortie
	20000 MB	300 byte	300 byte

TABLE 4.3 – Les Les paramètres des taches.

Comparaison par rapport au temps d'exécution

Le tableau suivant montre les résultats de comparaison entre notre approche proposé et le RR par rapport au temp d'éxécution.

Approche / nombre des taches	Approche proposé	RR
100	280	420
500	1360	2040
1000	2680	4020

TABLE 4.4 – Comparaison entre notre approche proposé et le RR en termes de temp d' exécution.

La figure suivante représente les résultats obtenus en terme de temp d' exécution

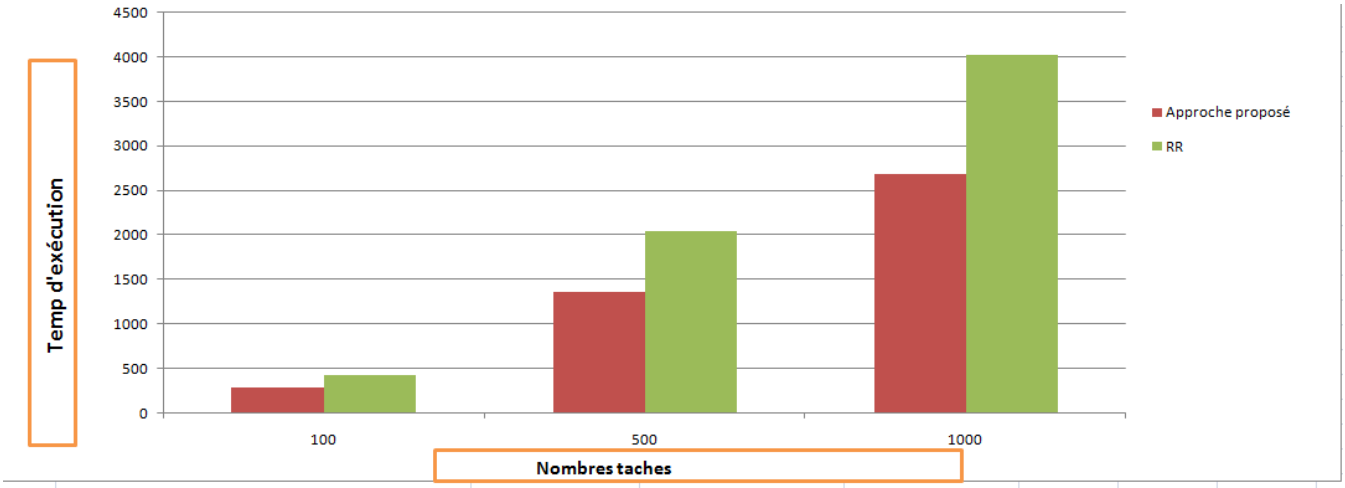


FIGURE 4.8 – Comparaison entre notre approche proposé et le RR en termes de temps d'exécution

Le nombre des taches à un impact sur le temps d'exécution pour le RR par contre les performances de notre approche proposé par rapport au RR sont meilleures, en particulier pour 500 et 1000 taches

Comparaison par rapport au coût

Le tableau suivant montre les résultats de comparaison entre notre approche proposé et le RR par rapport au coût.

Approche / nombre des taches	Approche proposé	RR
100	5580	10260
500	27990	62910
1000	55980	108000

TABLE 4.5 – Comparaison entre notre approche proposé et le RR en termes de coût

La figure suivante représente les résultats obtenus en terme de coût

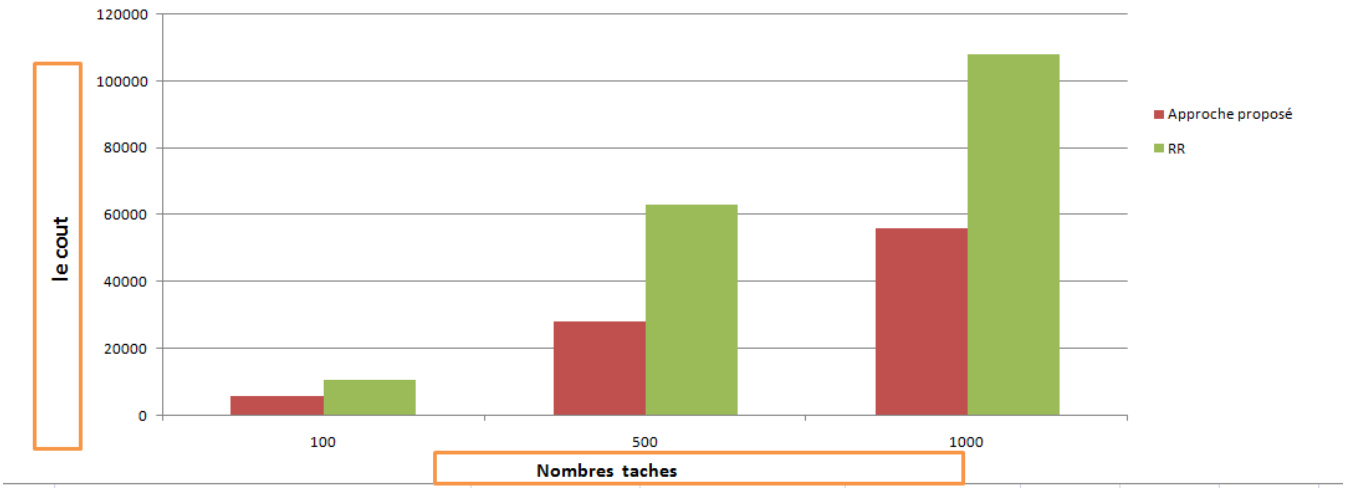


FIGURE 4.9 – Comparaison entre notre approche proposé et le RR en termes de coût

Le nombre des taches à un impact sur le coût pour le RR par contre les performances de notre approche proposé par rapport au RR sont meilleures, en particulier pour 500 et 1000 taches.

Comparaison par rapport au consomation d' énergie

Le tableau suivant montre les résultats de comparaison entre notre approche proposé et le RR par rapport au consomation d' énergie.

Approche / nombre des taches	Approche proposé	RR
100	66900	157500
500	333600	705180
1000	666900	1710900

TABLE 4.6 – Comparaison entre notre approche proposé et le RR en termes de consomation d' énergie.

La figure suivante représente les résultats obtenus en terme de consomation d' énergie

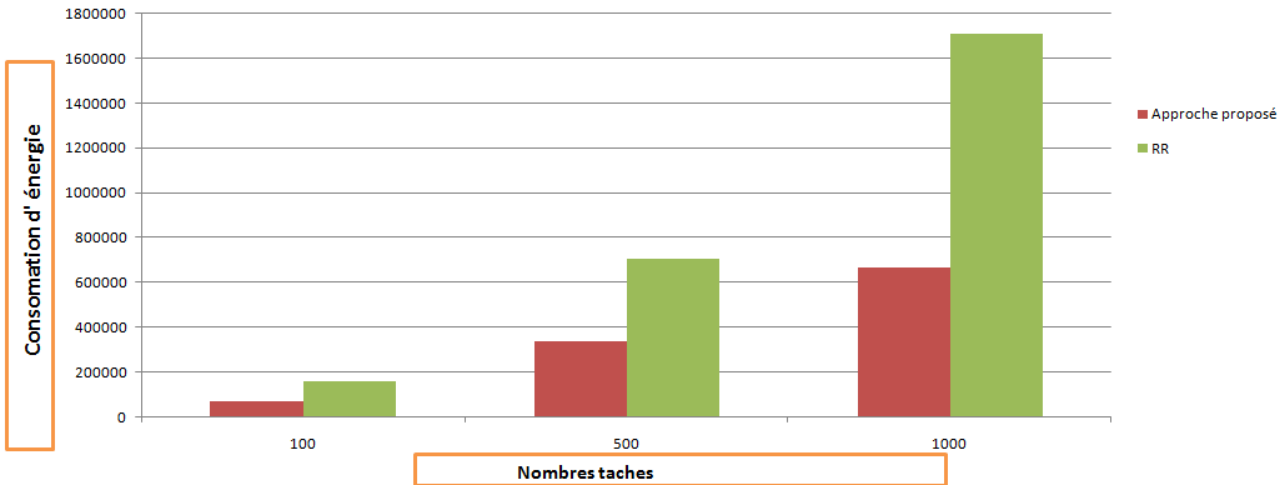


FIGURE 4.10 – Comparaison entre notre approche proposé et le RR en termes de consomation d' énergie

Le nombre des taches à un impact sur l'énergie consommée pour le RR par contre les performances de notre approche proposé par rapport au RR sont meilleures, en particulier pour 500 et 1000 taches.

4.5 Conclusion

Ce chapitre contient la présentation de notre approche proposé . Nous avons présenté les outils de simulations utilisés, le langage JAVA, l'IDE NetBeans ainsi que le simulateur CloudSim.
Nous avons implémenté les deux phases de notre algorithme à savoir la phase d'initialisation (fenêtres principale de configuration) et la phase d'ordonnancement (estimation, fonction objectif et ordonnancement).
Notre application permet d'afficher les statistiques sur l'ensemble des tâches après chaque simulation.
Les résultats de notre simulation montrent l'efficacité de notre approche par rapport à l'approche classique RR concernantle temps d'exécution, le coût et la consomation d'énergie.

CONCLUSION GENERALE

Le Cloud Computing présente une technologie prometteuse qui facilite l'exécution des applications dans divers domaines. Ces applications contiennent généralement de nombreuses tâches. Ces tâches requièrent pour leurs exécutions sur les machines du Cloud un ordonnancement.

Dans ce travail, nous avons proposé un algorithme pour l'optimisation des trois objectifs (temp d'exécution et le coût et consomationd'énergie) dans le Cloud computing. Cet algorithme est basé sur la métaheuristique cuckoo search permettant de trouver la solution optimale en un temps raisonnable.

nous avons comparé notre approche proposé avec un algorithme classique déjà implémenté dans CloudSim qui est le Round Robin.

Les simulations réalisées avec CloudSim donne des bonne résultats de notre approche proposée par rapport à l'algorithme Round Robin.

notre perspective est d'ajouter d'autres contraintes tels que la fiabilité, l'équilibrage de charge et la disponibilité et la consolidation. Appliquer d'autres métaheuristiques serait aussi possible afin de se comparer avec le cuckoo search.

BIBLIOGRAPHIE

[1] A.Griffith R.Joseph D. Katz R. H. Konwinski. Armbrust, M.Fox. *« Above the clouds : A berkeley view of cloud computing (Vol. 17). Technical Report UCB/EECS-2009-28, EECS Department". University of California, Berkeley ».* 2009.

[2] Bilal Mazhar Shehzad Ali Rabiya Jalil Javaria Khalid Babur Hayat Malik, Mehwashma Amir. *« Comparison of Task Scheduling Algorithms in Cloud Environment ».* Department of CS and IT The University of Lahore, Gujrat Campus, Pakistan,2018, p.387.

[3] L. Baccouche. *Thèse de doctorat, l'institut national polytechnique de Grenoble « Un Mécanisme d'Ordonnancement Distribué de Tâches Temps Réel ».* 2004.

[4] Marco Tomassioni. Bastien Chopard. *« Une introduction aux métaheuristique.».* Novembre 2017, p(149,169), p(83,103), p(107,114), p(65,73), p(40,50).

[5] Samah Bouamama. *« Gestion des ressources dans les Cloud Computing à base des modèles économiques ».* Thèse de doct. Université d'Oran1-Ahmed Ben Bella, URL : https ://theses.univ-oran1.dz/document/ TH3342.pdf., 2011, p. 7.

[6] «Qu'est ce que le nuage (CLOUD) ?». (consulté en 2021),URL : https ://www.danslenuage.quebec/blogue/quest-ce-que-le-nuage-cloud.

[7] Green IT Consulting. *«Cloud Computing»*. URL :http ://www.greenit-monaco.com/cloud-computing.htm, 2009-2014.

[8] « Introduction Plan Introduction Notion doptimisation combinatoire ». (consulté en 2021), URL : https ://slidetodoc.com/chapitre-1-introduction-plan-introduction-notion-doptimisation-combinatoire/.

[9] Définition du Cloud Computing selon NIST. *« Les modèles de déploiement du cloud computing selon NIST : »*. URL :https ://www.hebergeurcloud.com/definition-cloud-computing-selon-nist/.

[10] Khaled M Khalil et al. *« Cloud Simulator–An Evaluation Study »*. URL :http ://www.foibg.com/ijima/vol06/ijima06-01-p01.pdf., In : International Journal "Information Models and Analyses"(2017), p. 1-23.

[11] Malik et al. *« Comparison of Task Scheduling Algorithms in Cloud Environment »*. URL :https ://-thesai.org/Downloads/Volume9No5/ $Paper_50$ — $JournalofAdvancedComputerScienceandApplication(2018), p.1-7.$

[12] Hamad A et Omara.. *« Genetic-based task scheduling algorithm in cloud computing environment »*. URL : https ://www.researchgate. net/publication/$301945977_Genetic - Based_Task_Scheduling_Algorithm_in_Cloud_Computing_Environment.$, $In : International Journal of Advanced Computer Science and Applications(201$ 556.

[13] Amira Gherboudj. *Thèse Pour l'obtention du diplôme de Doctorat 3ème cycle LMD Spécialité : Informatique « Méthode de Resolution de Problémes Defficiles Académiques »*. Université de Constantine2, 2013.

[14] harabi Salaheddine Abderrahmen-Benallal Mohamed Islam. *Memoire de master « Ordonnancement Multi-critères dans le Cloud Computing »*. 2018 – 2019 , p.12-13-14.

[15] Djebbar Esma Insaf. *« optimisation d'ordonnancement et d'allocation de ressources dans les cloud computing »*. URL : https ://theses.univ-oran1.dz/ document/15201701t.pdf., Thèse de doct. Université d'Oran 1-Ahmed Ben Bella, 2016.

[16] Wang-L.Tao C. Kramer J.Kunze, M.Castellanos and W.Karl.. *« Scientific cloud computing : Early definition and experience. In High Performance Computing and Communications, HPCC'08. 10th IEEE International Conférence »*. "2008,(pp. 825-830)".

[17] JAVA LANGAGE. langage de programmation java». URL :http ://fr.Wikipédia.org/wiki/Java(langage)., (2008) , (cf. p. 48, 49).

[18] Mell.P and Grance. *« The NIST definition of cloud computing »*. 2011.

[19] Bourmaki Imane Mohammedi Meriem. *« Algorithme d'optimisation multi-objectifs dans le Cloud computing »*. URL :http ://dspace.univ-tlemcen.dz/bitstream/112/16404/1/BOURMAKI-Imane-MOHAMMEDI-Meriem.pdf, Mémoire de fin d'études Pour l'obtention du diplôme de Master en Informatique : Réseaux et Systèmes et Distribués (R.S.D) 2019.

[20] « NetBeans». (consulté en 2021),URL :https ://fr.wikipedia.org/wiki/NetBeans ,visite 2021.

[21] Tim Grance et al. Peter Mell. *«The NIST definition of cloud computing .»*. URL :https ://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf., 2011.

[22] Ray. P.Pradham, P. Behera B. *« Modified Round Robin Algorithm for Resource Allocation in Cloud Computing »*. 2010.

[23] Redhat. *«Qu'est-ce qu'un fournisseur de cloud ?»*. (consulté en 2021)".URL :https ://www.redhat.com., 2016.

[24] E.Bourgeois R.Hennion, H.Tournier. *«Cloud Computing»*. 2013.

[25] Francine Berman. Richard Wolski. *« Adaptive Computing on the Grid Using AppLeS, IEEE Transactions on Parallel and Distributed System »*. 2002.

[26] Anton Belo Glazov Cesar A.F de Rose Rajkumar Buyya. Rodrigo N.calhieros, Rajiv ranjan. *«CloudSim a toolkit for modeling and simulation of cloud computing environ- ments and evaluation of resource provisioning algorithms»*. "24 août 2010".

[27] Sellami Tiako et Chelouah and Ahmed-Nacer. *« Immune Genetic Algorithm For Scheduling Service Workflows With QOS Constraints in Cloud Computing »*. URL : https ://www.researchgate.net/publication/260839376$_I mmune_g enetic_a lgorith$ 15..

[28] Idir Tir et Matthias ali Belkacem Vary-Sinamal. *«Data center etcloud computing»*. Utec marne la vallée, 2016.

[29] Vallée. Vary-Sinamale et Belkacem. *« Data centers et cloud coputing ». Thèse de doct. UTEC Marne La Vallée,*. URL :http ://docplayer.fr/57913951- M2- m2i- gr- a- datacenters-cloudcomputing-memoire- de- recherches- utec- marne- la-vallee-arthurvary-sinamale-idir-tiret-matthias-ali-belkacem.html., 2016,p.6.

[30] Meslem yamina et Debbas soumia ismahèn. *«Ordonnancement et réplication de données dans le Cloud Computing»*. Université Dr. Tahar Moulay SAIDA, 2016-2017.

[31] Sonia Yassa. *« Allocation optimale multicontraintes des workflows aux ressources d'un environnement Cloud Computing »*. 2015.

[32] Buyya R. Yeo C. S. and S. Venugopal. *« Market-oriented cloud and atmospheric computing : Hype, reality, and vision. In 10th IEEE Proc Inter- national Conference on High Performance Computing and Communications »*. "2008,(pp. 5-13)".

[33] public cloud et hybrid cloud » «Private cloud. (consulté en 2021),URL :- https ://www.globalsp.com/private-cloud-public-cloud-et-hybrid-cloud/.

[34] « Cloud Computing Services ». (consulté en 2021),URL :https ://www.muchtech.org/2017/07/cloud-computing-services.html.

Résumé

Le cloud Computing est de plus en plus reconnu comme une nouvelle façon d'utiliser les services de calcul, de stockage et de réseau. Dans ce projet, nous abordons le problème d'ordonnancement de workflows sur les infrastructures du cloud computing. La majorité des travaux existants se concentrent principalement sur un ou deux objectifs. Notre objectif est de développer et d'évaluer un algorithme d'optimisation multiobjectif pour l'ordonnancement de workflows.

Mots Clés : Cloud computing, workflows, ordonnancement, Métaheuristique, optimisation multiobjectif, Cloudsim.

Abstract

Cloud computing is increasingly recognized as a new way of using compute, storage and networking services. In this project, we address the problem of scheduling workflows on cloud computing infrastructures. Most of the existing work focuses mainly on one or two objectives. Our goal is to develop and evaluate a multiobjective optimization algorithm for workflow scheduling.
Keywords : Cloud computing, workflows, scheduling, Metaheuristics, multiobjective optimization, Cloudsim.

ملخص

يتم التعرف على الحوسبة السحابية بشكل متزايد كطريقة جديدة لاستخدام خدمات الحوسبة والتخزين والشبكات. في هذا المشروع ، نعالج مشكلة جدولة سير العمل في البنى التحتية للحوسبة السحابية. يركز معظم العمل الحالي بشكل أساسي على هدف أو هدفين. هدفنا هو تطوير وتقييم خوارزمية تحسين متعددة الأغراض لجدولة سير العمل.

الكلمات الرئيسية: الحوسبة السحابية ، سير العمل ، الجدولة ، Metaheuristics ، التحسين متعدد الأهداف ، Cloudsim.