REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

CENTRE UNIVERSITAIRE BELHADJ BOUCHAIB D'AIN TEMOUCHENT

Institut des Sciences

Département de Mathématiques et Informatique

Mémoire

Pour l'obtention du Diplôme de Master en Informatique

Option : Réseaux et Ingénierie des Données (RID)

Présenté par :

MR. DERRAR ABDEL RAHIM

MR.AYED ZEDDAM MOHAMMED EL AMINE

Optimisation de la qualité de service (QoS) dans le Cloud Computing en utilisant les Algorithmes Génétiques

Encadrant : Mr. **BOUAFIA Zouheyr** Maitre Assistant "A" à C.U.B.B.A.T.

Soutenu le 27/09/2020

Devant le jury composé de :

Président : Mlle.Bouhalouan	(M.A.A)	C.U.B.B.A.T.
Examinateurs : Mlle.Bouhalouan	(M.A.A)	C.U.B.B.A.T.
Mlle Berrakem	(M.A.A)	C.U.B.B.A.T.
Encadrant : Mr Bouafia	(M.A.A)	C.U.B.B.A.T.

Dédicaces

Je dédie ce travail à :
- À mes chers parents pour leur soutien et amour permanent
- A mes chers frères et sœurs
- À mes amis et mes collègues

DERRAR ABDEL RAHIM

Dédicaces

Je dédie ce travail à :
- À mes chers parents pour leur soutien et amour permanent
- A mes chers frères et sœurs
- À mes amis MEDJAHRI HOSSIN SABRI , MAKNI ABDEL HAKIM , BENTABET AMIRA et mes collègues

AYAD ZEDDAM MOHAMMED EL AMINE

Remerciements

Nous remercions Allah de nous avoir donné le courage et la volonté ainsi que la patience d'avoir pu terminer notre travail .

Nous remercions également notre encadreur Mr BOUAFIA ZOUHEYR pour l'opportunité de réaliser ce sujet sous sa direction et la confiance faite ainsi que ses conseils fructueux, et son temps consacré tout au long du travail .

Nous remercions aussi les membres de jury d'avoir accepté d'être membre dans la soutenance.

Enfin , un merci particulier à tous ceux qui nous ont soutenu de prés ou de loin par leurs soutiens et encouragements .

Table des matières

Table des figures

Table des abréviations

Abréviation	Détail
NIST	National Institute of Standards and Technology
CPU	Central Processing Unit
RAM	Random-Access Memory
XaaS	XasaService
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
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
IBM	International Business Machines Corporation
SDN	Software Design Network
QOS	Quality Of Service
NP	non déterministe polynomial
PMO	Pareto-based Multiobjective Optimization
GA	Genetic Algorithm
CIA	Confidentiality Integrity Availability
CIS	Cloud Information Service
FCFS	First Come, First Served
IDE	Integrated Development Environment
MIPS	Million Instructions Per Second
SJF	Shortest Job First
SLA	Service Level Agreement
VM	Machine Virtuelle
TS-GA	Tournament Selection Genetic Algorithm
SDK	software développement kit
NSGA	Nondominated Sorting Genetic Algorithm
MOGA	Multi-Objective Genetic Algorithm
SPEA	Strength Pareto Evolutionary Algorithm
NSGA II	Nondominated Sorting Genetic Algorithm
TS	Tabu Search
PSO	Particle Swarm Optimization

Tableau 1 – Table des abréviations

Liste des tableaux

Résumé

L'informatique en nuage est une technologie qui fait référence aux ressources et systèmes informatiques disponibles sur demande via Internet. Ces ressources comprennent de l'espace pour le stockage et la sauvegarde des données. Elle comprend également des capacités de traitement programmatique et l'ordonnancement des tâches qui représentent une base importante pour améliorer la qualité de service (QOS) et ainsi améliorer l'utilisation des ressources.

Dans ce travail, nous étudions comment améliorer l'ordonnancement des tâches en utilisant les algorithmes génétiques pour optimiser les normes de qualité service telles que temps d'exécution des tâches, le Coût d'exécution, la fiabilité des ressources, la disponibilité des ressources, en plus la consommation d'énergie.

Mots clés :

Cloud Computing, Métaheuristiques, optimisation Multiobjectif, Algorithmes génétiques, Qualité de service , ordonnancement des tâches.

Abstract

Cloud computing is a technology that refers to the computing resources and systems available on demand via the Internet. These resources include space for data storage and backup. It also includes programmatic processing capabilities and task scheduling that represent an important basis in improving quality of service (QOS) and thus improving resource utilization.
In this work we study how to improve task scheduling work by using genetic algorithms to improve quality standards such as completion time , execution cost, reliability of resources, availability of resources, in addition energy consumption.

Key words :

Cloud Computing, Metaheuristics, Multiobjective optimization, Genetic algorithms, Quality of service, scheduling of tasks.

ملخص
الحوسبة السحابية تشير الى الموارد و الانظمة الحاسوبية تحت الطلب عبر شبكة الانترنت و تشمل تلك الموارد مساحة لتخزين و النسخ الاحتياطي كما تشمل قدرات معالجة برمجية و جدولة المهام التي تعتبر اساس مهم في تحسين جودة المستخدم و بالتالي تحسين استغلال الموارد
في هدا العمل ندرس كيفية تحسين عمل جدولة المهام باستعمال الخوارزميات الجينية لتحسين معايير الجودة مثل توقيت تنفيد المهام و كلفة تنفيد المهام و موثوقية الموارد و توافر الموارد بالاضافة الى تخفيض استهلاك الطاقة
الكلمات المفتاحية
الحوسبة السحابية ،تحسين متعدد الاهداف ، الخوارزميات الجينية ،جودة الخدمة ، جدولة المهام

Introduction

Pour rationaliser l'utilisation des ressources informatiques, de plus en plus des sociétés ont recours à l'hébergement et l'utilisation distante de leurs moyens de calcul ou de stockage. Cette tendance, liée à la fiabilité des réseaux et à la généralisation des techniques de virtualisation, conduit à disposer de ressources d'exécution ou de stockage louées chez des fournisseurs.

Dans ce contexte, on parle d'informatique dans les nuages, « Cloud Computing ». Le Cloud Computing présente une technologie prometteuse qui facilite l'exécution des applications scientifiques et commerciales. Il fournit des services flexibles et évolutifs à la demande des clients, via un modèle de paiement à l'usage.

Généralement, le Cloud Computing peut fournir trois types de services : IaaS , PaaS SaaS . Ces services sont offerts avec différents niveaux de qualités de services (QoS) afin de répondre aux besoins de différents groupes de clients. Bien que de nombreux services de Cloud Computing ont une fonctionnalité similaire (par exemple des services de calculs, services de stockages, services de réseaux, etc).

La théorie de l'ordonnancement est une branche de la recherche opérationnelle très importante dans Cloud Computing qui s'intéresse au calcul de dates d'exécution optimales des tâches. Pour cela, il est très souvent nécessaire d'affecter en même temps les ressources nécessaires à l'exécution de ces tâches. Donc le problème d'ordonnancement des tâches est un enjeu important qui influence considérablement les performances de l'environnement Cloud Computing. Les fournisseurs et les utilisateurs de services Cloud ont des exigences et des objectifs conflictuels. Un bon ordonnanceur doit mettre en œuvre un compromis acceptable entre ces objectifs.

Notre contribution dans le cadre de ce projet de fin d'études du master consiste à utiliser un algorithme génétique TS-GA pour l'optimisation de plusieurs objectifs dans le Cloud Computing. La problématique traitée concerne l'ordonnancement de tâches multiobjectif dans le Cloud Computing par rapport à plusieurs critères qui sont : le coût, la fiabilité, la consommation énergétique, le makespan et la disponi-

bilité.

Le simulateur CloudSim a été utilisé dans notre projet pour évaluer notre contribution. Les résultats obtenus sont très satisfaisants. Nous avons comparé trois versions de la fonction objective de l'algorithme TS-GA à savoir Pareto , agrégation et Pareto agrégation

Le reste de ce rapport est organisé comme suit :

Dans le premier chapitre, nous présenterons quelques notions de base sur le Cloud Computing ainsi que le processus d'ordonnancement des tâches qui est le contexte de travail de notre projet.

Le deuxième chapitre sera dédié à la présentation des méthodes de résolution des problèmes multiobjectifs en se focalisant en particulier sur les métaheuristiques, nous présenterons également une comparaison entre quelques méthodes d'ordonnancement des tâches dans le Cloud Computing.

Dans le dernier chapitre, nous présenterons d'unei manière détaillée l'algorithme génétique TS-GA utilisé dans notre projet, et nous présenterons ensuite les résultats trouvés.

Nous clôturons par une conclusion qui discute les contributions réalisées dans notre projet de fin d'étude ainsi que les perspectives des travaux futures envisagées .

Chapitre I

Concepts fondamentaux du Cloud Computing

I.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, et nous allons aborder aussi quelques concepts sur l'ordonnancement des tâches et les qualités de service (QOS) dans le Cloud Compting

I.2 Paradigme du Cloud Computing

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

I.2.1 Définition

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. » [35].

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. »

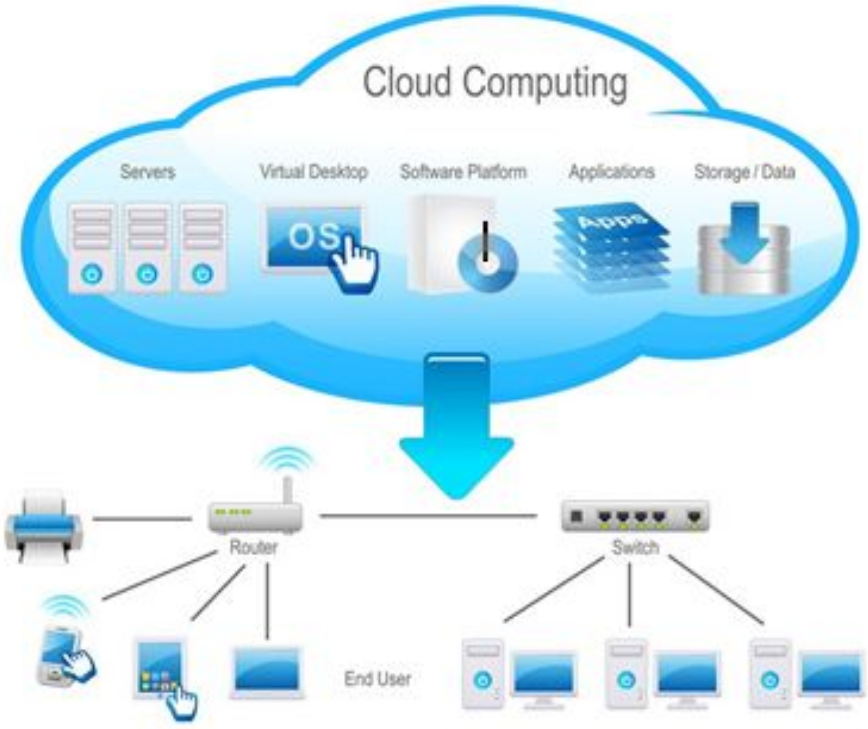

Figure I.1 – Cloud computing [41] .

I.3 Les Composants du Cloud Computing

Les technologies existantes telles que « grid computing » ou « utility computing » sont des composantes importantes du Cloud Computing [45]

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.
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.
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 exigences de conformité.

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.
5. **Gestion d'entreprise :** La couche de gestion d'entreprise manipule le cycle de vie des ressources virtualisées 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.[7]

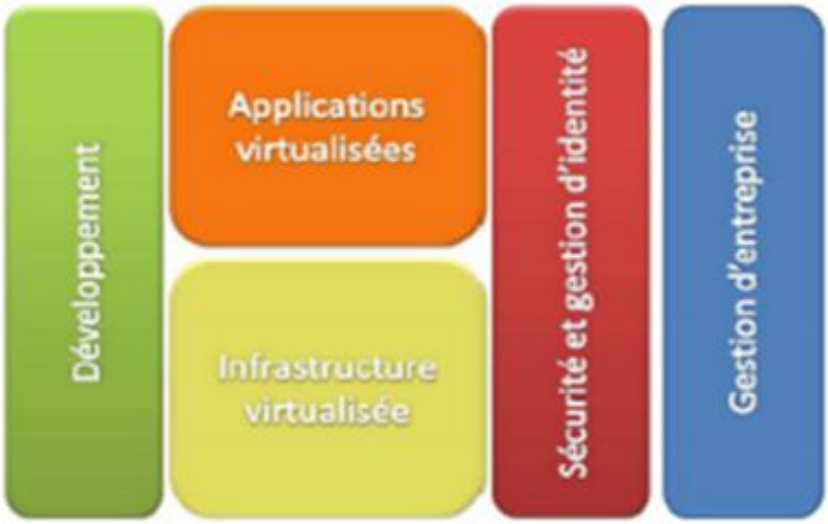

Figure I.2 – Les composants du Cloud Computing [7].

I.3.1 Notions de base

— **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.[4]

— **Machine physique :** (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.[41]

— **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. [41]

— **Tâches :** (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. [41]

— **Courtier :** (Broker) est l'intermédiaire entre les VMs et les Cloudlets (les tâches).

— **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]

— **Accord de niveau de service :** (Service level agreement SLA) 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.[45]

I.4 Les caractéristiques du Cloud Computing :

L'agence américaine National Institute of Standards and Technology (NIST) a identifié plusieurs caractéristiques essentielles pour un cloud performent qui sont :

- **Un 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.[26]
- **Une large accessibilité via le réseau :** Les services sont accessibles en ligne et sur tout type de support (ordinateur de bureau, portable, smartphone, tablette). Tout se passe dans le navigateur internet.[13]

- **Mesurabilité du service :** Dans un environnement de type Cloud le fournisseur de la solution de Cloud est capable de mesurer de façon précise la consommation des différentes ressources (CPU, Stockage, bande passante...), cette mesure lui permet ensuite de facturer le client selon l'usage. [12]

- **Élasticité rapide :** Grâce au Cloud, il est possible de disposer le plus de ressources, très rapidement pour soutenir une forte demande (par exemple : pour garantir une bonne expérience d'achat aux clients sur une plateforme Web de e-commerce durant les fêtes de fin d'année). Inversement, au-delà du provisionning de ressources, il est possible avec le Cloud de diminuer les ressources utilisées (ex : en cas de baisse d'activité sur cette même plateforme Web de e-commerce) si celles-ci sont supérieures à ce qui est réellement nécessaire [12].
- **La mutualisation des ressources :** Les ressources utilisées pour exécuter le service sont mutualisées pour servir à de multiples clients. Les multiples serveurs sollicités, totalement interconnectés, ne forment plus qu'une seule ressource virtuelle puissante et performante. Cette caractéristique est permise lorsque la solution déployée est multilocative.[13]

I.5 Les contraintes du Cloud Computing

La mise en ouvre d'une solution de Cloud Computing implique cinq principales contraintes qui doivent être prises en compte lors du choix, puis, du déploiement du service :

- **La sécurité des accès et des données :** Les risques de sécurité sont accrus par la délocalisation des ressources. L'accès au service induit donc des connexions sécurisées et authentifiées pour les utilisateurs et des garanties sur la confidentialité, l'intégrité et la traçabilité des données confiées stockées sur l'infrastructure du fournisseur. Le risque de perte de donnée doit également être anticipé via une procédure de sauvegarde adaptée.[13]
- **La localisation des données :** Le Cloud n'ayant pas de frontières, il faut pouvoir disposer d'un engagement sur les lieux de stockage des données, et s'assurer que les réglementations des pays sont en conformité avec les réglementations auxquelles le client est soumis ou bien qu'il souhaite avoir .[13]
- **Le niveau de qualité du service :** Le client doit disposer d'un droit de regard sur la qualité des prestations et le niveau de performance du service, avec un engagement contractuel de la part du fournisseur .[13]
- **Le coût à long terme :** Si les dépenses d'investissement sont limitées ou nulles, les dépenses d'exploitation doivent être correctement évaluées en disposant de métriques précises et à l'avantage du client pour mesurer la consommation réelle des ressources . [13]

- **La réversibilité du service :** En cas de rupture de contrat ou de changement de fournisseur, le client doit s'assurer de la récupération et de la destruction de ses données sur l'infrastructure du fournisseur après sa migration.[13]

I.6 Modèles du Cloud Computing

Cette section présente les modèles du Cloud. Nous commençons par détailler les modèles de service, puis nous donnons les modèles de déploiement .

I.6.1 Modèles de service du Cloud Computing

XaaS (X as a Service) représente la base du paradigme du Cloud Computing, où X représente un service tel qu'un logiciel, une plateforme, une infrastructure, un Business Process, etc. Nous présentons, dans cette sous-section les différents modèles de services.

1. **Software as a Service SAAS :** Le SaaS repose sur la capacité d'offrir à un client un environnement opérationnel complet avec des applications qui tournent sur une infrastructure Cloud. Les applications sont accessibles pour tous les périphériques clients (ordinateur personnel, tablette, Smartphone...) à travers une interface client léger tel qu'un navigateur web. Le client n'a pas la possibilité de gérer ou contrôler l'infrastructure Cloud sous-jacente tel-que l'infrastructure réseau, les serveurs ou les capacités de stockage mais il peut néanmoins configurer les paramètres d'application spécifiques à son utilisation. Fournisseurs actuels : SalesForce et Google Apps, Cloud9 Analytics.[8].
   D'autres catégories de services qui dérivent de SAAS sont également connues :
   — **Business Process as a service (BPaaS) :** Il fournit des services regroupant des applications qui intègrent la logique, les données et les processus de business des entreprises [50].
2. **Platform as a Service PAAS :** Le PAAS est une excroissance du modèle de déploiement Software as a Service. Une architecture PAAS est un modèle composé de tous les éléments nécessaires pour soutenir la construction, la livraison, le déploiement et le cycle de vie complet des applications et des services exclusivement disponibles à partir d'internet. C'est une architecture sans téléchargement ou installation de logiciels pour les développeurs, responsables informatiques . Elle est également connue sous le nom de « cloudware » [22]. Fournisseurs actuels : Google App Engine , Engine Yard , Red Hat OpenShift , Heroku [9].

3. **Infrastructure as a Service (IaaS) :** Plateforme sur laquelle des administrateurs IT vont pouvoir déployer une infrastructure (machine(s) virtuelle(s) et socle applicatif et applications...). Ce modèle qui est une évolution des centres de données virtualisés permet au client de faire abstraction du modèle physique (gestion des serveurs physique, des éléments relatifs aux centres de données comme l'électricité, la climatisation, la sécurité physique). Dans ce modèle, le fournisseur contrôle le matériel et la couche de virtualisation. Au niveau des données, le contrôle est partagé au niveau de la machine virtuelle (qui est stocké et sauvegardé par le fournisseur de Cloud de type IaaS). [12] Fournisseurs actuels : Amazon AWS , Google Computer Engine , Microsoft Azure , IBM SmartCloud Enterprise [9]

   D'autres catégories de services qui dérivent de IaaS sont également connues :

   — **Network as a Service (NaaS) :** Il permet la création de réseau à la volée entre machines virtuelles et de gérer sa configuration. Ces services sont très souvent associés avec des SDN (Software Design Network) qui facilitent énormément la manière de configurer et de gérer son réseau. Exemple : OpenNaas [9].

   — **Data as a Service (DaaS)** Le DaaS permet l'utilisation de données délocalisées. Ce type de service est utilisé par des applications composites, dites mashup dont le rôle est de corréler des données venant de milieux hétérogènes. Exemples : Factual , InfoChimps [9].

   — **Storage as a Service (STaaS)** Cette catégorie de service offre des solutions de stockage de fichiers chez des prestataires externes tels que : Amazon S3 , iCloud , SkyDrive , Wuala , Google Drive, Ubuntu One ou Dropbox. [50]

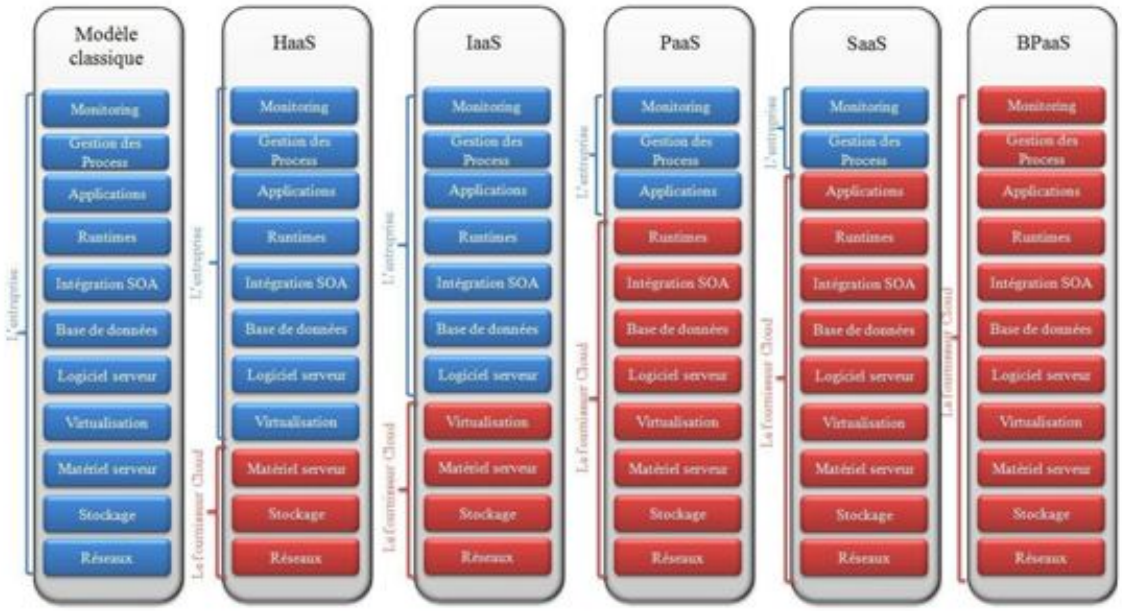

Figure I.3 – Différents modèles service du Cloud Computing.

I.6.2 Modèle de déploiement du Cloud Computing

On distingue quatre modèles de déploiement du Cloud. Chacun de ces modèles correspond à une architecture particulière décrite , après ces modèles de déploiement sont représentés par la figure I.4

- **Cloud privé :** L'infrastructure est réservée pour l'utilisation exclusive d'un seul organisme comprenant plusieurs consommateurs (unités d'affaires). L'infrastructure peut être détenue et gérée par l'organisme, une tierce partie ou toutes les deux dans ses locaux ou ailleurs. [8]
- **Cloud public :** L'infrastructure Cloud est ouverte au public ou à de grands groupes industriels. Cette infrastructure est possédée par une organisation qui vend des services Cloud. Pour les consommateurs, il n'y a donc aucun investissement initial fixe et aucune limite de capacité. Les fournisseurs de Cloud public facturent à l'utilisation et garantissent une disponibilité de services au travers des contrats SLA [47].
- **Cloud hybride :** Un Cloud Hybride est l'utilisation de plusieurs Clouds, publics ou privés. Ces infrastructures sont liées entre elles par la même technologie qui autorise la portabilité des applications et des données. C'est une excellente solution pour répartir ses moyens en fonction des avantages recherchés [35].

- **Cloud de communauté :** L'infrastructure Cloud est déployée en vue d'un usage exclusif par une communauté spécifique d'utilisateurs partageant des besoins informatiques communs (sécurité, stratégie, conformité, etc.).[26]

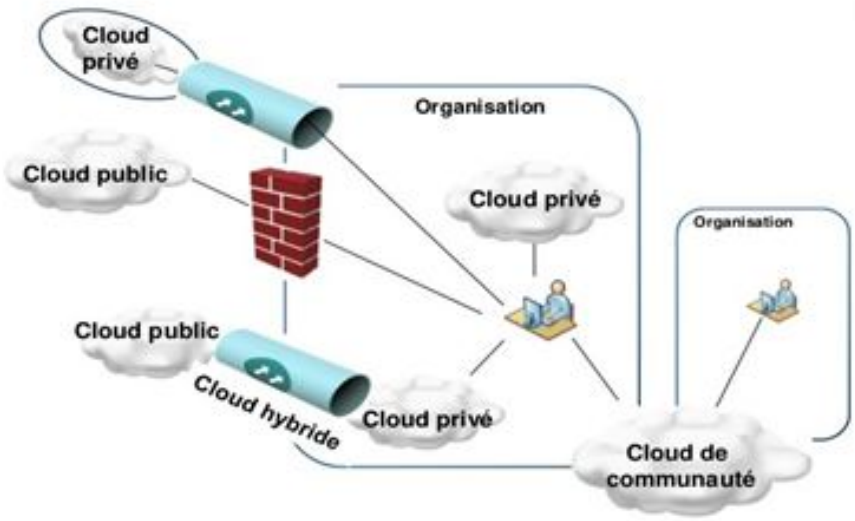

Figure I.4 – Différents modes de déploiement du Cloud Computing .

I.7 Les fournisseurs du Cloud Computing

Dans cette section nous allons présenter les principaux fournisseurs de Cloud IAAS :

Les offres	GoGrid [30]	Joyent [48]	Rackspace [40]	VMWare [46]	Amazon EC2 [27]
présentation	GoGrid fournit un dynamique et un scaling des ressources en temps réel. L'une des principales caractéristiques est la technologie RAID $6+0$ qui garantit la résilience, la disponibilité. -fournisseur propose différentes solutions pour les bases de données SQL et NoSQL	Joyent propose trois types de services. Le service de calcul facilite l'exécution d'applications fiables et résilientes sur le Cloud. -Le service de stockage " manta " offre un service de stockage d'objets distribués. Le troisième service fourni facilite l'utilisation de Cloud prive . les fonctionnalités du fournisseur sont le zonage, la mise en cache à haute intégrité et traçage etc.	- Rackspace offre une disponibilité à 100% des applications. En outre, les utilisateurs peuvent également créer des images instantanées de sauvegarde pour une utilisation ultérieure. Un API RESTful est fournie avec un panneau de contrôle basé sur le Web pour contrôler les ressources Cloud	La solution Cloud de VMWare est basée sur sa suite vCloud qui fournit un accès basé sur API pour gérer et contrôler le Cloud. La caractéristique la plus importante du fournisseur est une fiabilité améliorée basée sur une approche prédictive et sur les données pour le signalement des incidents et la gestion des problèmes	Amazon est un fournisseur IaaS qui offre les services calculs (EC2) et stockage(S3). (EC2) est basé sur des concepts de zones qui offrent une isolation des pannes et une robustesse contre les défaillances dans d'autres zones. Différentes zones se sont combinées pour former des régions. Le stockage est également proposé en tant que service de stockage simple (S3).
Virtualisation	Xen hypervisor	Solaris container	OpenStack-based	vSphere	Xen hypervisor
Modèle de payement	Pay-as-you-go, après Abonnement	Pay-as-you-go	Pay-as-you-go	abonnement	Pay-as-you-go après Abonnement
Principaux produits / services	Calcul, équilibrage de charge, stockage, distribution de contenu réseau	Calcul, stockage, Cloud privé	Gestion de Cloud, stockage en bloc, équilibrage de charge, surveillance	Calcul, , gestionnaire de récupération	Elastic block store, IP addresses, virtual private Cloud, Cloud watch, clusters
Flash	512 Ko	256 Ko	32 Mo	1 Mo	Néant
Coût mémoire ($ par GB)/mois	0.45	42.21	23.08	18.42	4.30
Coût de stockage ($ par 100 GB)/mois	15	37.97	6.94	39.66	44.47
Coût de cpu ($ par CPU)/mois	70	25	28.38	62.44	16.08

Tableau I.1 – Comparaison des fonctionnalités fournies par les principaux fournisseurs de Cloud Computing IaaS [37]

I.8 L'ordonnancement dans le Cloud Computing

I.8.1 Définition :

L'ordonnancement des tâches est un problème NP-complet. Un problème d'ordonnancement consiste à ordonner dans un temps un ensemble des tâches contribuant à la réalisation d'un même projet. L'objectif est de minimiser la durée de réalisation du projet compte tenu des contraintes d'antériorité reliant les différentes tâches. De plus, on détermine les calendriers de réalisation de chacune de ces tâches ainsi que les marges de manœuvre associées. [25]

I.8.2 Processus d'ordonnancement

Se composent de tout ou partie des étapes suivantes :

— **La phase de priorisation des tâches (task prioritizing) :** cette phase établit l'ordre des tâches de départ selon leurs priorités. De façon plus précise, les tâches de départ sont celles envoyées par l'utilisateur du Cloud et qui arrivent au système d'ordonnancement comme étant le point d'entrée du Cloud. Dans notre étude, nous considérons qu'aucune tâche n'est prioritaire, donc la file d'attente est ordonnée selon le principe du premier arrivé premier servi.

— **La phase d'allocation des ressources : (resource provisioning/allocation) :** réserve ou alloue un ensemble de ressources, c'est-à-dire qu'elle calcule le nombre de machines virtuelles pour l'ordonnancement des tâches.

— **La phase d'ordonnancement (scheduling) :** cette phase sélectionne les ressources (machines virtuelles, CPU, disque, mémoire) parmi celles précédemment allouées. Cette phase fait l'ordonnancement de chaque tâche selon un algorithme qui définit la manière avec laquelle les tâches sont affectées aux machines virtuelles à travers les commutateurs du centre de données [25]

I.8.3 Problème NP-complet

Définition : Un problème de décision est dit NP-complet si tout problème de la classe NP lui est polynomialement réductible. Notons qu'il existe des problèmes dans NP qui ne sont pas dans P et qui, vraisemblablement, ne seront pas dans la classe NPC (pour dire NP-complet) Le concept central relié à la définition de la NP-complétude est celui de la réduction entre problèmes.

Définition formelle :
Un problème de décision est NP-complet c si :

1. c est dans NP.
2. Chaque problème NP est réductible à temps polynomiale c

I.8.4 Rôle d'ordonnanceur

Un ordonnanceur (scheduler) devra trouver la machine la plus appropriée pour traiter les tâches qui lui sont soumises. Les ordonnanceurs peuvent aller du plus simple (allocation de Type round-robin) au plus compliqué (ordonnancement à base de priorités). L'ordonnanceur détermine le taux de charge de chaque ressource, afin de bien ordonnancer les prochaines tâches. Il peut aussi superviser le déroulement d'une tâche jusqu'à sa terminaison, la soumettre à nouveau si elle est brusquement interrompue et la terminer prématurément si elle se trouve dans une boucle infinie d'exécution.[25]

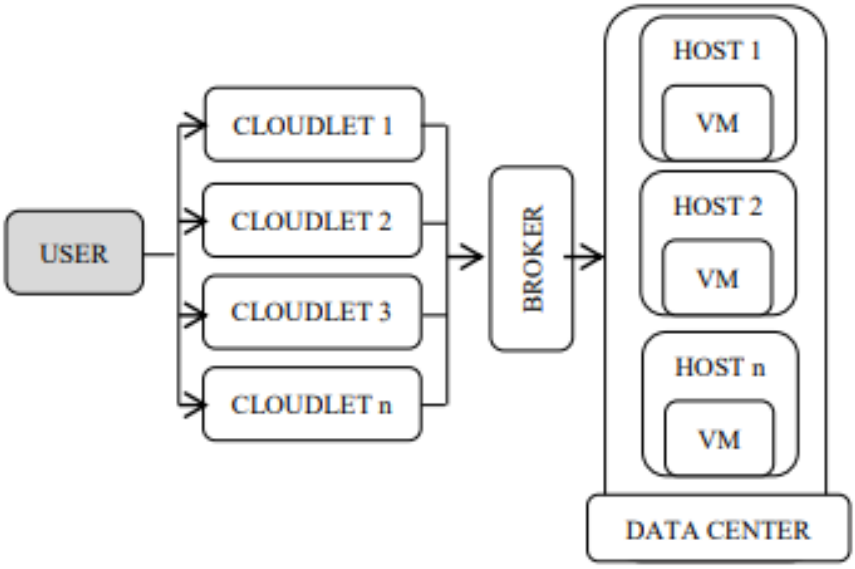

Figure I.5 – Les différents acteurs d'ordonnancement

I.9 Qualité de service (QOS) dans le Cloud Computing

L'utilisation de services de Cloud Computing est basée sur la définition d'un contrat, appelé SLA (Service Level Agreement), qui lie le fournisseur des services à ses utilisateurs. Ainsi, dans un SLA se trouve un ensemble de clauses, que le fournisseur s'engage à respecter. Celles-ci peuvent concerner divers domaines :

- Performance du matériel
- Disponibilité des services
- Sécurisation des données
- Coûts de location des machines virtuelles, etc...

La qualité de service est fréquemment modélisée par des métriques basiques telles que le temps de réponse, la consommation énergétique ou le coût des services.[25]

I.10 Conclusion

Ce chapitre a été consacré à la présentation de quelques notions de base du Cloud Computing nécessaires à la compréhension de ce domaine, en particulier son architecture, ses types ainsi que les principaux fournisseurs IAAS. Ensuite nous avons abordé le processus d'ordonnancement des tâches dans le Cloud Computing. Le deuxième chapitre de notre rapport sera consacré à la présentation de méthodes liées à la résolution des problèmes multiobjectifs, et les algorithmes d'ordonnancement des tâches dans Cloud Computing.

Chapitre II

Ordonnancement des tâches multiobjectif dans le Cloud Computing

II.1 Introduction

L'optimisation multiobjectif est une approche très importante du fait de la nature multiobjectif de la plupart des problèmes réels, tels que le problème d'ordonnancement des tâches dans le cloud. Les premiers travaux menés sur les problèmes multi objectifs furent réalisés au 19ème siècle, sur des études en économie par Edgeworth (Edgeworth, 1881) et généralisés par Pareto (Pareto, 1896).

Ce chapitre est organisé en deux parties. Dans la première partie, nous donnons un état de l'art de l'optimisation multiobjectif. Nous présentons d'abord un ensemble de définitions et de concepts de base de l'optimisation multiobjectif dans la section II.2 Puis nous présentons l'optimisation multiobjectif et l'aide à la décision dans la section II.3 . La section II.4 traite les méthodologies de résolution et nous présentons les concepts communs aux métaheuristiques,dans la deuxième partie de ce chapitre, nous donnons un panorama de méthodes d'ordonnancement des tâches dans le Cloud Computing.

II.2 Concepts de base

II.2.1 Formulation générale d'un problème d'optimisation multiobjectif

L'optimisation multiobjectif consiste à sélectionner un vecteur de décisions qui satisfait les contraintes et optimise le vecteur objectif dont les éléments représentent les fonctions objectives,
ces dernières forment une description mathématique du critère de performance, ces fonctions sont généralement en conflit. Le terme « optimiser » veut dire trouver une solution de telle façon que toutes les fonctions objectives renvoient des valeurs acceptables.

Un problème de minimisation est décrit comme ci-dessous [51] :

$$\begin{aligned} minimiser\ y = &\quad (f_1(x), f_2(x), ..., f_n(x)) \\ Avec\ x = (x_1, ..., x_m) &\in X \\ y = (y_1, ..., y_m) &\in Y \end{aligned}$$

Où x est appelé vecteur de décision constitué de m variables de décision, dans un algorithme génétique il représente un individu (solution potentielle). Et y vecteur objectif de n objectifs. X représente l'espace de décision, et Y l'espace des objectifs.

II.2.2 Notions de dominances et d'optimalité

II.2.2.1 Définition 1 : Dominance faible de Pareto :

Une solution $y \in y_1, y_2, \ldots, y_m$ domine faiblement une solution $z \in z_1, z_2, \ldots, z_m$ Si seulement si pour $i \in 1, 2, \ldots m$ on a $f_i(y) \leq f_i(z)$ [5] Si la solution y domine faiblement la solution z nous allons noter y_z.

II.2.2.2 Définition 2 : Dominance forte de Pareto

Une solution $y = (y_1, ..., y_m)$ domine fortement une solution $z = (z_1, ..., z_m)$. Si seulement si pour $i \in (1, 2, ..., m)$ on a $f_i(y) < f_i(z)$ [5]

II.2.2.3 Définition 3 : Front de Pareto :

Le front (frontière) de Pareto est l'ensemble des solutions Pareto optimales qui sont composées des points, ne sont dominés par aucun autre Le front de Pareto appelé aussi surface de compromis ou l'ensemble des solutions efficaces [39].

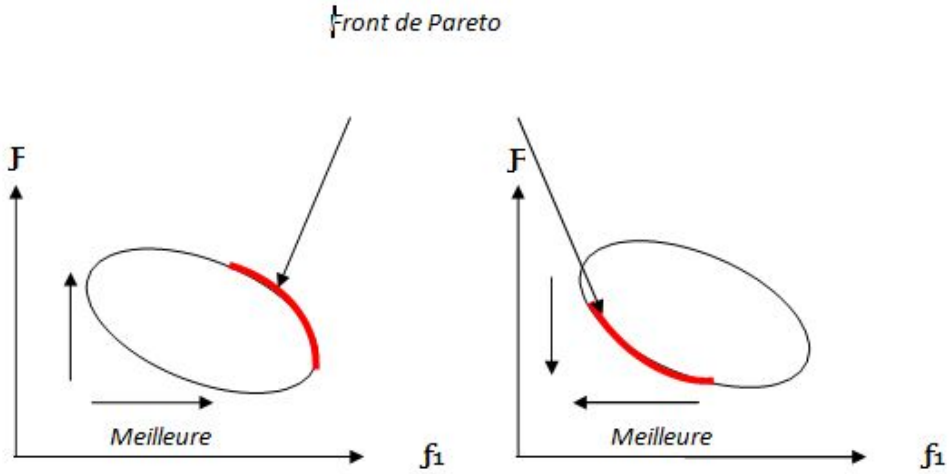

FIGURE II.1 – Le front de Pareto [39].

II.3 Classification des méthodes de résolution des problèmes multiobjectifs :

Dans les différentes publications, nous rencontrons deux classifications différentes des approches de résolution de problème multiobjectif :

II.3.1 Classification "point de vue décideur" :

Cette classification est essentiellement utilisée en recherche opérationnelle. Les décisions étant considérées comme un compromis entre les objectifs et les choix spécifiques du décideur (contraintes de coût, de temps, etc.), un décideur choisit une méthode en fonction de l'aide qu'elle va lui apporter. [5] [6]

II.3.1.1 Approches d'optimisation à priori :

Dans ces méthodes, le compromis que l'on désire effectuer entre les différentes fonctions objectives a été déterminé avant l'exécution de la méthode d'optimisation. Pour obtenir la solution de notre problème, il suffira de faire une et une seule recherche et, en fin d'optimisation, la solution obtenue reflétera le compromis que l'on désirait effectuer entre les fonctions objectives avant de lancer la recherche. Cette méthode est intéressante dans le sens où il suffit d'une seule recherche pour trouver la solution. Cependant, il ne faut pas oublier le travail de modélisation du compromis qui a été effectué avant d'obtenir ce résultat. Il ne faut pas, non plus, oublier que le décideur est "humain" et qu'il sera susceptible de constater que la solution obtenue en fin d'optimisation ne le satisfait finalement pas et qu'il souhaite maintenant, après avoir examiné cette solution, une solution qui opère un autre compromis entre les fonctions objectif.[10]

II.3.1.2 Approches d'optimisation progressive (interactive) :

Dans ces méthodes, on cherchera à questionner le décideur au cours de l'optimisation afin que celui-ci puisse réorienter la recherche vers des zones susceptibles de contenir des solutions qui satisfassent les compromis qu'il souhaite opérer entre les fonctions objectives. Ces méthodes, bien qu'appliquant une technique originale pour modéliser les préférences du décideur, présentent l'inconvénient de monopoliser l'attention du décideur tout au long de l'optimisation. Cet inconvénient n'est pas majeur lorsque l'on a affaire à des problèmes d'optimisation pour lesquels la durée d'une évaluation de la fonction objective n'est pas importante. Pour les autres problèmes, l'utilisation d'une telle méthode peut être délicate. Il est en effet difficile de monopoliser l'attention du décideur pendant une période qui peut s'étendre sur plusieurs heures en lui posant, de plus, des questions toutes les dix minutes, voire même toutes les heures [10].

II.3.1.3 Approches à posteriori :

Dans ces méthodes, on va chercher un ensemble de solutions bien réparties dans l'espace de solutions. Le but sera ensuite de proposer ces solutions au décideur pour qu'il puisse sélectionner la solution qui le satisfait le plus en jugeant les différentes solutions proposées. Ici, il n'est plus nécessaire de modéliser les préférences du décideur. On se contente de produire un ensemble de solutions que l'on transmettra au décideur. Il y a donc un gain de temps non négligeable vis-à-vis de la phase de modélisation des préférences de la famille des méthodes à priori. L'inconvénient qu'il faut souligner est que, maintenant, il faut générer un ensemble de solutions bien réparties. Cette tâche est non seulement difficile, mais, en plus, peut requérir un temps d'exécution prohibitif [10].

II.3.2 Classification "point de vue concepteur" :

Ce classement adopte un point de vue plus théorique articulé autour des notions d'agrégation et d'optimum de Pareto. Ces notions sont développées dans les sections suivantes car nous adoptons cette classification pour présenter les différentes méthodes.[5] [39]

II.3.2.1 Approches heuristiques :

Les approches heuristiques peuvent être divisées en deux classes : d'une part les algorithmes spécifiques à un problème donné qui utilisent des connaissances du domaine et d'autre part des algorithmes généraux (méta-heuristique) applicables à une grande variété de PMO. Les approches heuristiques ne garantissent pas de trouver de manière exacte tout l'ensemble des solutions Pareto.

Méthodes exploitant une métaheuristique

Méthodes souvent inspirées de mécanismes d'optimisation rencontrés dans la nature. Elles sont utilisées pour les problèmes où on ne connaît pas d'algorithmes de résolution en temps polynomial et pour lesquels on espère trouver une solution approchée de l'optimum global. Elles cherchent à produire une solution de meilleure qualité possible dictée par des heuristiques avec un temps de calcul raisonnable en examinant seulement une partie de l'espace de recherche.

1. **La recherche locale** La recherche locale (RL) est une méthode classique en recherche opérationnelle qui consiste à explorer à une itération donnée le voisinage de la solution courante. L'exploration se fait en modifiant un ensemble de composantes de la solution courante pour se déplacer vers une nouvelle solution. Le processus est répété itérativement jusqu'à ce qu'un critère d'arrêt choisi soit rencontré. Ce type d'approche nécessite la définition de plusieurs concepts : celui de voisinage associé à une solution, et celui de mouvement effectué entre deux itérations. Le voisinage d'une solution est défini en fonction du problème à résoudre. Il représente, par définition, l'ensemble des solutions accessibles depuis la solution courante. [2]
2. **Le recuit simulé :** Le recuit simulé (Simulate dannealing) s'inspire du processus du recuit physique. Le processus du recuit simulé répète une procédure itérative qui cherche des solutions de coût plus faible tout en acceptant de manière contrôlée des solutions qui dégradent la fonction de coût.[19]
3. **La recherche Tabou :** La recherche tabou (TS) est une méthode de recherche locale combinée avec un ensemble de techniques permettant d'éviter d'être piégée dans un minimum local ou la répétition d'un cycle. La recherche tabou est introduite principalement par Glover. Cette méthode a montré une grande efficacité pour la résolution des problèmes d'optimisation difficiles. En effet, à partir d'une solution initiale s dans un ensemble de solutions local S, des sous-ensembles de solution N(s) appartenant au voisinage S sont générés. Par l'intermédiaire de la fonction d'évaluation nous retenons la solution qui améliore la valeur de f, choisie parmi l'ensemble de solutions voisines N(s).[18] [19] [20]
4. **Les algorithmes génétiques :** Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils ont été adaptés à l'optimisation par John Holland , également les travaux de David Goldberg ont largement contribué à les enrichir . Leur fonctionnement est extrêmement simple, on part d'une population de solutions potentielles (chromosomes) initiales, arbitrairement choisies. On évalue leur performance (Fitness) relative. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. Quelques individus se reproduisent, d'autres disparaissent et seuls les individus les mieux adaptés sont supposés survivre.[21][24]

II.3.2.2 Approches Pareto :

Ces méthodes sont fondées sur la notion de dominance au sens de Pareto qui privilégie une recherche satisfaisant au mieux tous les objectifs. Une seule résolution permet d'approximer l'ensemble de la frontière Pareto. Ces approches appartiennent également aux approches de type à « posteriori ». [10]

1-Les méthodes non-élitistes

Nondominated Sorting Genetic Algorithm (NSGA) : Proposée par Srinivas et Deb en 1994, dans cette méthode, avant que la sélection soit entamée, les solutions sont classifiées à base de non-dominance. Tous les individus non-dominés sont classés dans une catégorie avec une valeur de fitness factice proportionnelle à la taille de la population afin de fournir une possibilité de reproduction égale pour tous les individus.[44]

Multi-Objective Genetic Algorithm (MOGA) : Fonseca et Fleming ont proposé cette méthode qui classifie les individus selon le nombre d'individus qui les dominent, tous les individus non-dominés dans la population obtiennent le même rang et la même valeur de fitness pour qu'ils aient tous la même possibilité d'être sélectionnés. Le MOGA utilise la méthode de formation de niches afin de diversifier la population. [31]

2- Les méthodes élitistes

On appelle élitistes, les méthodes qui conservent les solutions Pareto-optimales pour une réintroduction dans des populations nouvelles.

Strength Pareto Evolutionary Algorithm (SPEA) : Proposée par Zitzler et Thiele en 1998 [52], cette méthode actualise l'archive (population externe) avec les solutions non-dominées, chaque individu de la population se fait attribuer une valeur de fitness proportionnelle au nombre de solutions qu'il domine. La diversité est préservée en utilisant la technique de clustring qui permet de réguler le nombre d'individus tout en gardant la diversité des solutions.

Nondominated Sorting Genetic Algorithm (NSGA II) : Proposée par Deb et al en 2002, dans cette méthode, la population est classifiée selon la dominance de Pareto, elle attribue à chaque individu un rang de non-dominance et une distance de crowding qui permet d'évaluer de la densité de la région autour de cet individu. Cette méthode est bien moins complexe que sa devancière [11].

II.3.2.3 Approches non Pareto :

Ne traitent pas le problème comme un véritable problème multiobjectif. Elles cherchent à ramener le problème initial à un ou plusieurs problèmes mono objectifs. Les approches non Pareto sont classées en deux catégories : les approches scalaires, qui transforment le problème multiobjectif en problème mono-objectif et les approches non scalaires, qui gardent l'approche multiobjectif, mais en traitant séparément chacun des objectifs.

Approches non Pareto scalaires :

Elles consistent à transformer le vecteur objectif en un scalaire, il en existe plusieurs types :

1. **La moyenne pondérée :**
   Dans cette approche, le but consiste à ramener le problème multicritère à un problème monocritère plus simple à traiter. Cette méthode est la plus simple des méthodes d'optimisation multiobjectif. La transformation que l'on effectue est la suivante :

   $$\text{Minimiser} \qquad \sum_{i=1}^{m} w_i.f_i(x)$$

   $$\text{avec} \qquad x \in X, w_i \in [0,1] \ \text{ et } \sum_{k=1}^{m} w_i = 1$$

   X : représente le domaine réalisable.
   w_i : Appelé le poids, est une pondération associée au critère, cette pondération permet d'exprimer des préférences sur les critères de décision [3].
2. **Méthode ε-contraintes :**
   Dans cette approche, le problème consiste à optimiser une seule fonction objective f_k sujette à des contraintes sur les autres fonctions objectives (Convertir p-1 des p objectifs du problème en contraintes) [32]

   $$\begin{aligned} &\text{minimiser } f_1(x) \\ &\text{Tel que } f_2(x) \leq \varepsilon_2, \ldots\ldots, f_m(x) \leq \varepsilon_m \\ &x \in X \end{aligned}$$

3. **Méthode de but à atteindre :**
   On définit un ensemble de buts qu'on espère atteindre pour chaque fonction objective . L'algorithme tente de minimiser l'écart entre la solution courante et ses buts. : [32]

$$\begin{aligned} &\text{minimiser } y \\ &\text{Tel que } f_i(x) - w_i.y \leq B_i \\ &Avec \\ &g - q(x) \leq 0 \\ &Avec\ x \in R^n, g_q(x) \in R^q, F(x) \in R^n \end{aligned}$$

Approches non Pareto non scalaires :

1. **Sélection parallèle :** Cette approche a été la première proposant un algorithme génétique pour la résolution de problèmes multiobjectifs . L'algorithme proposé, VEGA (Vector Evaluated Genetic Algorithm), sélectionne les individus selon chaque objectif de manière indépendante. L'idée est simple : Pour k objectifs et une population de n individus, une sélection de n/k meilleurs individus est effectuée pour chaque objectif. Ainsi k sous-populations vont être créées et ensuite mélangées afin d'obtenir une nouvelle population de taille n. le processus se termine par l'application des opérateurs génétiques (croisement et mutation). [42]
2. **Sélection lexicographique :** Cette approche, proposée par Fourman en1985 , elles classe les objectifs en fonction d'un ordre d'importance proposé par le décideur. Ensuite l'optimum est obtenu en optimisant tout d'abord la fonction objectif la plus importante puis la deuxième en intégrant les valeurs obtenues comme contraintes pour la résolution sur des objectifs moins prioritaires et ainsi de suite. La solution obtenue à l'étape k sera la solution du problème. Le risque essentiel de cette méthode est la grande importance attribuée aux objectifs classés en premier. La meilleure solution trouvée pour l'objectif le plus important va faire converger l'algorithme vers une zone restreinte de l'espace d'état et enfermer les points dans une niche. [6]

II.3.3 Classification méthodes d'ordonnancement des tâches dans le Cloud computing

Sur la base des travaux pertinents dans la littérature [33], nous classons les méthodes d'ordonnancement des tâches généralement en trois groupes (figure II.2) :

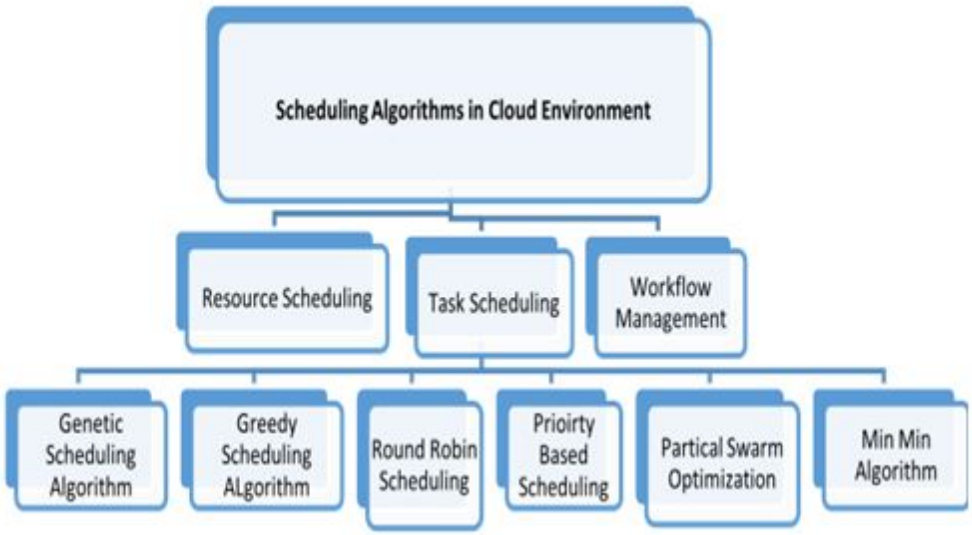

Figure II.2 – Classification des techniques d'ordonnancement [33]

Dans notre thèse on s'intéresse juste à l'ordonnancement des tâches (task scheduling).

II.3.4 Les algorithmes d'ordonnancement des tâches basiques

L'ordonnancement des tâches est un problème NP-complet qui joue un rôle important dans le Cloud Computing. Les algorithmes d'ordonnancement deviennent de plus en plus compliqués, car ils doivent gérer un grand nombre de paramètres contradictoires et qui concernent différents acteurs, dont les intérêts ne sont pas les mêmes. Le problème d'ordonnancement de ces tâches comporte un groupe d'objectifs contradictoires (QoS) à optimiser. Par exemple (minimisation du coût, de temps réponse du service et la consommation d'énergie et autres). La motivation principale de ces algorithmes d'ordonnancement est de minimiser le temps et le coût d'exécution.

Dans cette section on détaille les algorithmes d'ordonnancement basiques :

1. **Algorithme génétique (Genetic Algorithm)**

Les différentes étapes de l'algorithme GA que les chercheurs introduisent pour résoudre un problème d'ordonnancement des tâches sont détaillées d'une façon générale dans cette section, L'algorithme commence par l'initialisation de la génération initiale de manière aléatoire. Après, Chaque solution sera évaluée en utilisant une fonction objective qui sera calculée en fonction des métriques de qualité service. L'évolution de la population est valide par opérateurs génétiques, la sélection, le croisement et la mutation et le remplacement. L'algorithme itère le processus jusqu'à atteindre un nombre maximum d'itérations ou jusqu'à ce que la valeur globale de l'objectif ne s'améliore plus.[49]

1.1 Représentation d'une solution :

Pour résoudre un problème d'ordonnancement des tâches avec l'algorithme génétique , une solution valable doit respecter les conditions suivantes :

- Une machine virtuelle peut exécuter une seule ou plusieurs tâches successivement.
- Chaque tâche est exécutée une seule fois sur une seule machine. [49].

Un exemple de représentation des tâches :

Tâches	T1	T2	T3	T4	T5	.....	Tn
Id VM	VM3	VM2	VM1	VM3	VM2	.....	VM4

Tableau II.1 – Représentation d'une solution pour l'ordonnancement des tâches

1.2 Fonction objective

Les algorithmes génétiques sont généralement efficaces pour résoudre des problèmes d'optimisation multiobjectifs. Donc la conception de la fonction objective dépend de problème étudié et la nature des métriques de qualité service . [49]

1.3 Opérateurs génétiques

1.3.1 L'opérateur de sélection

Le but de l'opérateur de sélection est d'éviter de conserver les meilleures solutions, il y'a plusieurs méthodes de sélection telles que :

Sélection par la roulette : proposé par J. Holland .Elle consiste à sélectionner les solutions proportionnellement à leur performance, donc plus les solutions sont adaptées au problème, plus elles ont de chances d'être sélectionnées. La probabilité de sélection est calculée à partir de la valeur de " fonction objective " du chromosome dans la population. Ainsi une solution x a la probabilité suivante d'être sélectionnée [28] :

$$P_{sel}(X) = \frac{Fitness(x_i)}{\sum_{i=1}^{PopSz} Fitness(x_i)}$$

Où **PopSz** représente le nombre de solutions d'une population.

Sélection par rang : Cette méthode consiste à ranger d'abord les solutions selon un ordre (croissant ou décroissant de leurs performances) puis à attribuer (à l'aide d'une procédure) une probabilité de sélection en fonction de rang. Cette probabilité est proportionnelle à la performance et est donnée par la formule suivante [34] :

$$P_{sel}(X) = \frac{Rang(x_i)}{\sum_{i=1}^{PopSz} Rang(x_i)}$$

Où

PopSz représente le nombre de solutions d'une population.

Sélection par Tournoi : Cette méthode de sélection augmente les chances des individus de "mauvaise qualité" par rapport à leur fitness, de participer à l'amélioration de la population. En effet, c'est une compétition entre les individus d'une sous-population de taille M (M < N) prise au hasard dans la population. Le paramètre M est fixé à priori par l'utilisateur. L'individu de meilleure qualité par rapport à la sous-population sera considéré comme vainqueur et sera sélectionné pour l'application de l'opérateur de croisement. Le paramètre M joue un rôle important dans la méthode du tournoi. Par conséquent le choix de M permet de faire varier la pression sélective. De cette manière, on contrôle les chances de sélection des individus les plus performants par rapport aux plus faibles. [34]

1.3.2 Opérateur de croisement (crossover)

L'opérateur de croisement a pour but d'enrichir la diversité de la population en manipulant les composantes des chromosomes. Le principe de cet opérateur est de produire deux nouveaux individus enfants, En échangeant des informations entre deux autres individus

Pour clôturer la partie sur les opérateurs de croisement, nous en présentons quelques types :

Croisement à un point (One-point Crossover) : Cet opérateur consiste à choisir aléatoirement un point de coupure pour partager chaque parent en deux parties. Le premier enfant est construit en utilisant la première partie du premier parent et la deuxième partie du deuxième parent. A l'inverse, le deuxième enfant est une combinaison de la seconde partie du premier parent et de la première partie du second parent [49]

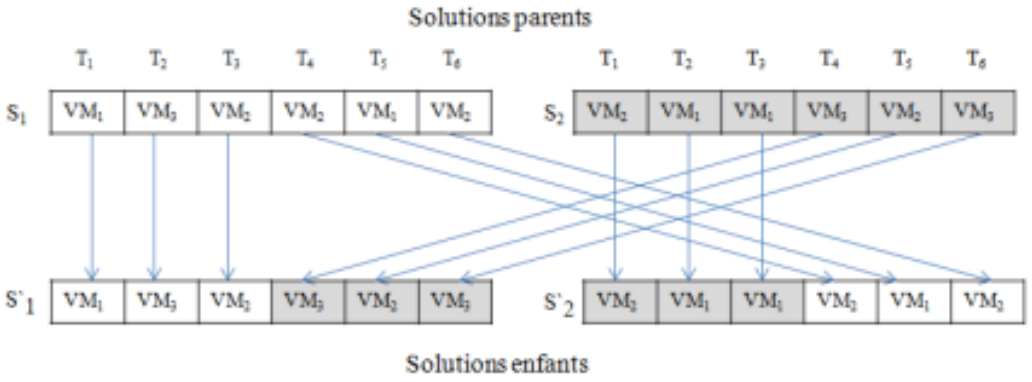

Figure II.3 – Exemple du « One-point Crossover » pour l'ordonnancement des tâches

Croisement à deux points (Two-point Crossover) : Cet opérateur consiste à choisir deux points aléatoires pour former une fenêtre de croisement. Puis les segments de VM se trouvant dans la fenêtre de croisement sont échangés pour former deux solutions enfants, comme illustré sur la figure II.4 [49]

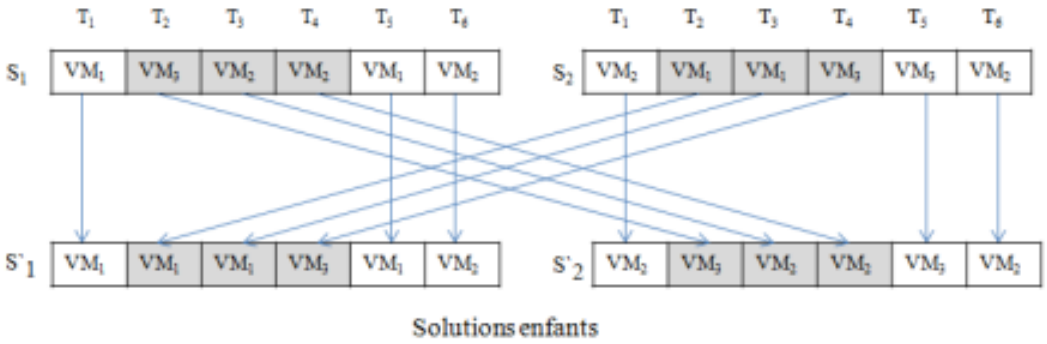

Figure II.4 – Exemple du « Two-point Crossover » pour l'ordonnancement des tâches

Croisement aléatoire (Random Crossover) La solution enfant hérite de façon aléatoire des gènes de ses parents, la figure II.5 illustre ce type de croisement.[14]

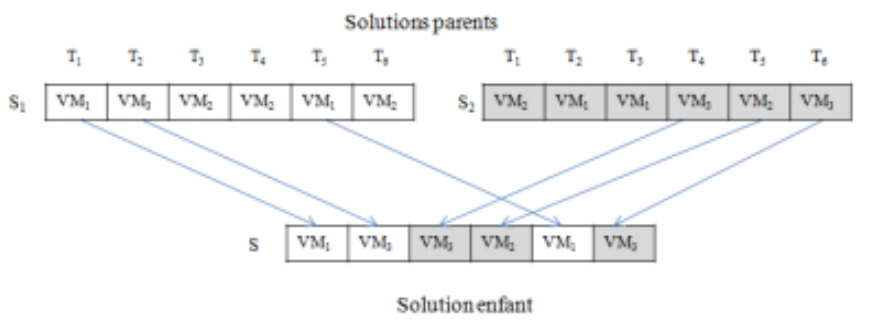

Figure II.5 – Exemple du « Random Crossover » pour l'ordonnancement des tâches

Opérateur de mutation

L'objectif de l'opérateur de mutation est d'éviter une convergence vers une solution optimale locale, en réintroduisant des caractéristiques aléatoires qui n'appartiennent à aucune des solutions parentes, nous en présentons quelques types :

La mutation locale : Ce type de mutation a besoin d'un opérateur local, connu spécifiquement pour les problèmes d'optimisation de "forme" : Le principe consiste à faire des petites médications localisées sur les contours du domaine de conception. Cependant, ce type de mutation ne permet pas d'explorer tout le domaine de conception. [34]

La mutation uniforme : Est appliquée sur les individus moins performants. Elle consiste à choisir le point de mutation aléatoire. Dans ce type de mutation, tous les points ont la même probabilité d'être mutés. Mais l'aspect aléatoire de ce dernier type de mutation ne peut garantir une diversité génétique qui représente un facteur important dans un algorithme génétique. [34]

Remplacement

Cette étape consiste à construire la nouvelle population. Pour conserver les meilleures solutions. [49]

Les critères d'arrêt

L'algorithme termine son exécution lorsque l'une des conditions d'arrêt est validée. Les conditions d'arrêt du processus sont les suivantes :

- Le nombre de générations atteint une limite maximale ;
- Stabilité de la fonction objective (la fonction objective se stabilise lorsque cette fonction ne s'améliore pas avec une nouvelle itération de l'algorithme).[49]

2. **Algorithme de planification des tâches basé sur la priorité**

   Une approche innovante par Shamsollah Ghanbari et Mohamed Othman pour gérer l'ordonnancement des tâches dans le cloud [40]. L'algorithme donne une priorité élevée à la tâche qui nécessite une puissance de calcul élevée. Une tâche qui nécessite un peu de puissance de calcul on lui donne une faible priorité [41].

3. **Algorithme Round Robin**

   L'algorithme d'ordonnancement des tâches Round Robin affecte les tâches sélectionnées aux machines virtuelles disponibles à tour de rôle, comme présenté à la figure II.6 l'algorithme envoie les tâches reçues aux machines virtuelles disponibles en suivant un cercle fermé. De manière plus précise, la première tâche est affectée à la première machine virtuelle du serveur 1, la deuxième tâche est affectée à la première machine virtuelle du serveur 2, la troisième tâche est affectée à la première machine virtuelle du troisième serveur, alors que la quatrième tâche, la cinquième tâche et la sixième tâche sont affectées respectivement à la deuxième machine virtuelle du premier, du deuxième et du troisième serveur. [38]

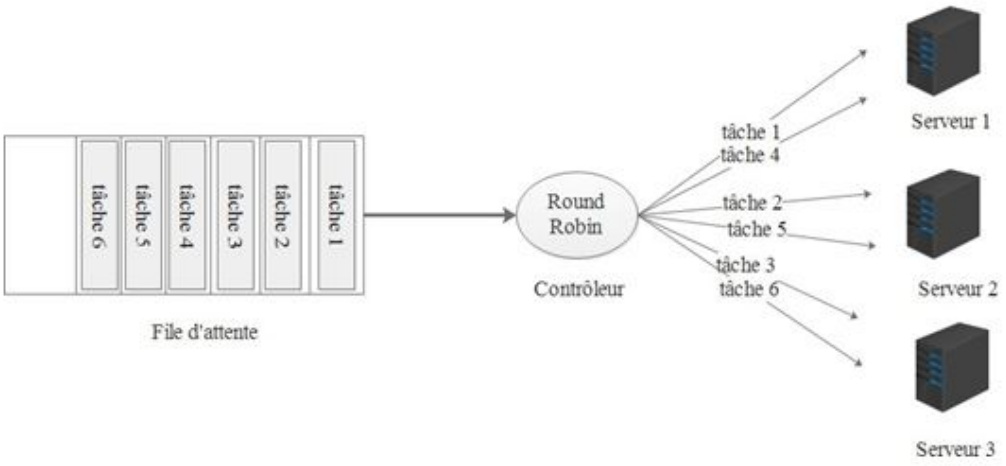

Figure II.6 – Ordonnancement des tâches avec Round Robin .

4. **Algorithme Min-Min**
   L'algorithme commencer par calculer le temps d'exécution minimal pour toutes les tâches , la valeur minimale est choisie et qui représente le temps d'exécution minimal parmi toutes les tâches sur les ressources ensuite en fonction de ce temps minimum la tâche est ordonnancée sur la machine correspondante puis le temps d'exécution des autres tâches se mis à jour sur cette machine en ajoutant le temps d'exécution de la tâche assignée au temps d'exécution des autres tâches sur cette machine/ressource et la tâche assignée est supprimée de la liste des tâches. Ensuite la même procédure est répétée jusqu'à ce que toutes les tâches soient assignées sur les ressources .[36]
5. **Algorithme d'optimisation par l'essaim de particules (PSO)**
   L'optimisation par Essaim Particulaire (OEP), ou Particle Swarm Optimization (PSO) en anglais, est une méthode d'optimisation stochastique basée sur une population de solutions. Elle a été développée par Kennedy et Eberhart en 1995 (Kennedy, 1985).).Elle est inspirée du comportement social des colonies d'insectes, des nuées d'oiseaux, des bancs de poissons et bien d'autres sociétés animales évoluant en essaim. L'algorithme PSO prend en considération les tâches et les ressources disponibles dans l'environnement Cloud, En même temps pour garder la ressource aussi occupée que possible et à minimiser le temps de traitement de la tâche.[15]
6. **Algorithme Premier arrivé, premier servi (FCFS)**
   FCFS est un algorithme simple, l'idée est d'ajouter tâche et ressource disponible dans une file et d'exécuter chaque tâche et ressource par l'ordre d'arrivée. [33]
7. **Algorithme glouton**
   Les problèmes d'optimisation peuvent être facilement résolus à l'aide d'algorithme glouton. L'objectif fondamental de cet algorithme était de diviser le problème en sous problèmes à chaque étape et faire la meilleure décision. Son objectif principal était d'obtenir la meilleure solution. [33]

II.3.5 Comparaison entre les algorithmes d'ordonnancement des tâches

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 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 du 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.

Tableau II.2: Comparaison entre les algorithme d'ordonnancement basiques.[33]

II.3.6 Travaux connexes

Dans cette section nous présentons quelques travaux liés à la planification des tâches indépendantes dans le Cloud Computing :

Zhao, Zhang et Hu (Zhao, 2009) ont proposé un algorithme basé sur approche génétique pour planifier les tâches indépendamment. Ils ont implémenté l'algorithme dans les systèmes hétérogènes, leur but est minimiser le temps d'achèvement et augmenter la productivité du système,

Guzek et al. (Guzek, 2012) examinent l'influence de la DVFS, en utilisant l'algorithme évolutionnaire multiobjectif (NSGA-II) pour l'ordonnancement biobjectif du makespan et de la consommation énergétique d'un graphe (DAG, Direct Acyclic Graph) de tâches sur une plateforme multiprocesseur hétérogène.

Jang et al. (Jang, 2012) ont proposé un modèle d'ordonnancement des tâches dans lequel l'ordonnanceur invoque la fonction de l'algorithme génétique pour réaliser l'ordonnancement de tâches, en considérant la satisfaction des utilisateurs et les disponibilités des machines virtuelles (VMs)

Liu et al. (Liu, 2013) ont proposé un algorithme génétique multiobjectif MOGA pour l'ordonnancement des tâches sur les machines virtuelles du cloud. L'algorithme tente de minimiser la consommation d'énergie et de maximiser le profit du fournisseur, sous la contrainte de délai. Néanmoins, ils ne prennent pas en compte les contraintes de précédence entres les tâches.

S. Singh et M. Kalra (Singh, 2014) ont fourni plusieurs variantes de GA pour l'ordonnancement des tâches dans le Cloud. Ils ont introduit un algorithme pour résoudre le problème de planification des tâches en modifiant GA dans lequel la population initiale est générée par l'approche Max-Min pour obtenir des résultats plus optimaux en termes de «makespan»

S.A. Hamad et F.A. Omara (Omara,2016) ont proposé un algorithme génétique sous le nom de «Tournament Selection Genetic Algorithm (TS-GA) » ou ils introduisent une nouvelle méthode de croissement pour obtenir un meilleur ordonnancement des tâches dans le cloud Computing.

Chang-Tian et Jiong (Chang-Tian, 2012) ont utilisé la technique DVS et ont proposé deux algorithmes basés sur un algorithme génétique, à savoir : ETU-GA (Energy consumption Time Unify Genetic Algorithm) et ETDF-GA (Energy consumption Time Double Fitness Genetic Algorithm), pour l'ordonnancement de tâches indépendantes dans le Cloud Computing. Ces algorithmes visent à trouver l'équilibre entre le makespan et l'énergie.

Le tableau II.3 recense les différents travaux cités précédemment où y est ajoutée une dernière ligne concernant l'algorithme de notre contribution afin de se placer dans un contexte de comparaison avec les méthodes qui y sont énumérées.

Articles	**Algorithme**	**Méta heuristique**	**Makespan**	**Coût**	**Fiabilité**	**disponibilité**	**Énergie**	**Pareto**	**Non Pareto**
Zhao,2009	GA	oui	oui	non	non	non	non	non	oui
Guzek, 2012	NSGA-II	oui	oui	oui	non	non	non	oui	non
Jang, 2012	GA	oui	oui	oui	non	oui	non	non	oui
Liu, 2013	MOGA	oui	non	non	non	non	oui	non	oui
S. Singh, 2014	MGA	oui	oui	non	non	non	non	non	oui
Omara, 2016	TS-GA	oui	oui	non	non	non	non	non	oui
Chang-Tian, 2012	$ETU-GA, ETDF-GA$	oui	oui	non	non	non	oui	non	oui
~~Notre travail~~	$TS-GA$	oui	oui	oui	oui	oui	oui	oui	non
	$TS-GA$	oui	oui	oui	oui	oui	oui	non	oui

Tableau II.3 – Les travaux d'ordonnancement des tâches utilisant l'approche génétique comparés à notre approche [16]

D'après le tableau II.3 , nous constatons que plupart de travaux présentés optimisent une seule ou deux métriques de QoS à la fois. De plus, très peu de travaux considèrent la fiabilité des services. Pour répondre à ce manque, nous développons un algorithme génétique (GA), qui prend en compte plusieurs métriques de QoS, à savoir le makespan, le coût, la fiabilité et disponibilité des ressources et consommation d'énergie pour l'ordonnancement de tâches sur l'IaaS Cloud.

II.3.7 Conclusion

Dans ce chapitre, nous avons présenté les méthodes de résolution de problèmes multiobjectifs, à savoir les méthodes Pareto et les méthodes non pareto en se focalisant sur les métaheuristiques. Ensuite nous avons présenté quelques algorithmes d'ordonnancement des tâches dans le Cloud Computing.

Dans le cadre de notre projet, nous intéressons dans le chapitre suivant à l'application de l'algorithme génétique GA pour l'ordonnancement de tâches multiobjectifs dans le domaine du Cloud Computing par rapport à plusieurs critères qui sont le coût, la fiabilité, la disponibilité, le makespan et la consommation énergétique.

Chapitre III

Implémentation de l'algorithme génétique TS-GA pour l'ordonnancement des tâches et évaluation des résultats

III.1 Introduction

Étant donné que l'ordonnancement de tâches est un problème NP-complet, il devient difficile de le résoudre en utilisant des heuristiques classiques. L'utilisation d'algorithmes génétiques est généralement efficace pour résoudre un tel problème . . Dans ce contexte, l'objectif de l'ordonnancement consiste à une répartition des tâches sur des ressources du Cloud de façon à optimiser plusieurs métriques de qualité de services (QoS). Notons que la plupart des travaux sont concentrés souvent sur l'optimisation d'une seule métrique qui est le makespan.

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 GA pour l'ordonnancement des tâches multiobjectifs dans le Cloud Computing par rapport à cinq critères qui sont : le makespan, le coût, la consommation d'énergie, la disponibilité et la fiabilité.

Ainsi, nous présentons les principes de base de l'algorithme génétique GA pour l'ordonnancement des tâches multiobjectif, notamment, les méthodes pour mesurer la qualité d'un ensemble de solutions, Nous présentons également les outils de simulations utilisés dans le cadre de ce PFE, à savoir, le langage java, l'IDE NetBeans et le simulateur CloudSim. Nous terminons par une évaluation comparative par rapport aux résultats obtenus.

III.2 Modèle d'ordonnancement

Étant donné : un fournisseur d'une infrastructure IaaS offrant un ensemble de machines virtuelles VM, une tâche d'un utilisateur composée d'un ensemble T de n tâches qui doivent être exécutées sur ces VM. Le problème d'ordonnancement des tâches sur l'IaaS revient à construire un mapping M des tâches aux VM , qui optimise les métriques de QoS suivantes : le makespan, le coût, la fiabilité et la disponibilité et la consommation d'énergie .
Ces QoS sont souvent conflictuelles, par conséquent, le problème d'ordonnancement de tâches peut être formulé comme le problème d'optimisation multi objectif suivant :[49]

Minimiser $Makespan(c)$
Minimiser $Coût\ de\ traitement(c)$
Minimiser $Consommation\ d'énergie(c)$
Maxismiser $Fiabilité(c)$
Maxismiser $Disponibilté(c)$

III.3 Méthodes d'évaluation de quelques métriques de QoS

- **Makespan** L'une des mesures les plus utilisées dans l'évaluation des algorithmes d'ordonnancement de tâches 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 .[1]
  Le makespan est donné dans l'équation suivante :

$$\text{Makespan} = \max(\text{temps d'excution(i)})$$

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

$$\text{Le Temps d'exécution} = \frac{\text{Cloudletlength}}{(VMMIPS * VMPEsNumber)}$$

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

- **Fiabilité :** La fiabilité représente la probabilité que toutes les tâche soient entièrement exécutées avec succès, sans aucune faute de ressources. [43]

$$Fibabilté = e^{-\sum_{i=0}^{n} Temps\ d'exécution(i)*\lambda}$$

- n : nombre de tâche
- λ : est le taux d'échec de la machine qui exécute la tâche T.

- **Disponibilité :** La disponibilité peut être utilisée pour déterminer dans quelle mesure l'ensemble de machines virtuelles louées sont utilisées. Elle est calculée dans la formule .[43]

$$Disponibilité = \frac{1}{m} * (1 - \sum_{j=1}^{m} \frac{Temps\ d'exécution}{makespan})$$

- m est le nombre total de ressources au niveau de l'infrastructure IaaS ;

- **Coût de traitement :** Représente le coût nécessaire pour qu'une tâche s'exécute au niveau de la machine virtuelle. Nous utilisons cette formule : [43]

$$\text{Coût de traitement} = (\text{Prix} * \text{Temps d'exécution(j)})$$

Où :

- Prix :Le prix d'utilisation de Vm par second.

- **La consommation d'énergie :** Représente l'énergie consommée au niveau de la machine virtuelle pour faire un traitement d'une tâche. [49]

consommation d'énergie = maxpower*(0.7+(0.3*Temps d'exécution(j)))

- maxpower : la puissance maximale des serveurs qui se produit lorsque la charge de travail du centre de données est au maximum.

III.4 Présentation de l'algorithme génétique (TS-GA)

Dans notre projet on a utilisé l'algorithme génétique nommé « Tournament Selection Genetic Algorithm », cet algorithme est proposé par Safwat A. Hamad et Fatma A. Omara dans l'article nomme « Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment » pour résoudre le problème d' ordonnancement des tâches dans l'environnement Cloud Computing
Les principales améliorations de l' algorithme génétique TS-GA sont :

La différence majoritaire entre TS-GA et algorithme génétique standard est :

• La sélection par tournoi est utilisée à la place de La sélection par roulette dans le processus de sélection pour sélectionner la meilleure solution.

• Les solutions non retenues lors du processus de sélection sont considérées et ajoutées à la nouvelle population. Cela pourrait aider à générer la meilleure solution dans la prochaine génération.

• Le nouveau crossover est introduit en considérant les solutions parentes comme nouveau fils (voir figure III.2) [1]

III.5 Étapes de l'algorithme génétique TS-GA

Les différentes étapes de l'algorithme TS-GA, que nous proposons sont détaillées dans les sous-sections suivantes, L'algorithme commence par l'initialisation de la génération initiale de manière aléatoire . Après, chaque solution sera évaluée en utilisant une fonction objective qui sera calculée en fonction des préférences de QoS .

Le processus d'évolution de la population est assuré par différents opérateurs génétiques, notamment, la sélection, le croisement, et le remplacement. L'algorithme itère le processus jusqu'à atteindre un nombre d'itérations maximal.

III.5.1 Pseudo code

Algorithm 1 L'algorithme génétique TS-GA

Entrée T : liste des tâches, V : liste des Vms,
Sortie liste des affectations des tâches aux machines virtuelles.
Inisialisation p,K = 0.5,Nbgénération=20,taillePopulation=10,nbSolution : entier
S1,S2,S3,S4 :chromosome
intialPopulation,listeSolution,listeResCrossover,listeCrossover : liste des chromosomes
Générer la population P0 de taille N
pour i de 0 a Nbgénération faire
listeSolution =**FonctionObjectif**(intialPopulation)
p= **aléatoire**(0,1)
Si p<K **alors**
pour i de 0 a taillePopulation faire
pour i de 0 a nbSolution faire
(S1,S2)=**SélectionparTournoi**(intialPopulation)
fin pour
fin pour
(S3,S4)=**CroisementUnPoint**(S1,S2)
listeCrossover = **Ajouter**(S1)
listeCrossover = **Ajouter**(S2)
listeCrossover = **Ajouter**(S3)
listeCrossover = **Ajouter**(S4)
listeResCrossover=**FonctionObjectif**(listeCrossover)
Si listeResCrossover>1 **alors**
intialPopulation = **Remplacer**(listeResCrossover(0),S1)
listeCrossover = **Remplacer**(listeResCrossover(1),S2)
Sinon Si listeResCrossover == 1 **alors**
intialPopulation = **Remplacer**(listeResCrossover(0),S1)
intialPopulation = **Remplacer**(listeCrossover(3),S2)
Sinon
intialPopulation = **Remplacer**(listeCrossover(2),S1)
intialPopulation = **Remplacer**(listeCrossover(3),S2)
finsi
finsi
finsi
fin pour

III.5.2 Représentation d'une solution

Pour le problème d'ordonnancement des tâches indépendantes, une solution valable doit respecter les conditions suivantes :

• Chaque tâche est exécutée une seule fois sur une seule machine.

• Une machine virtuelle peut exécuter plusieurs tâches successivement . [1].

• Une population est un ensemble des solutions (chromosomes) générées initialement par l'algorithme TS-GA.

• La taille de solution égale au nombre des tâches.

• Chaque gène contient ID de tâche et ID de VM.[1].

Notre codage d'une solution (chromosome) est représenté dans la figure III.1 .

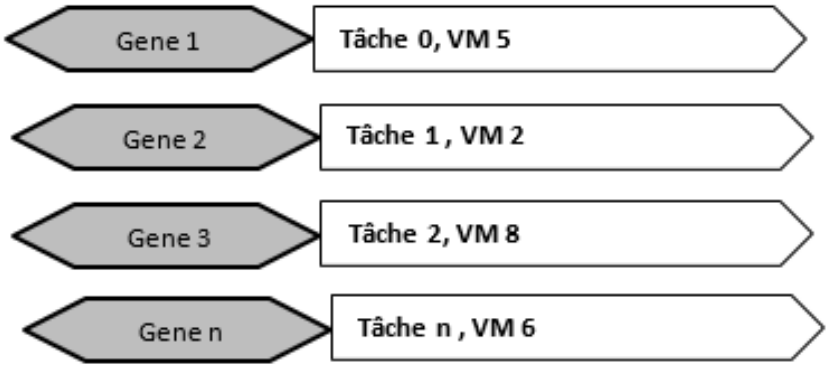

Figure III.1 – Représentation graphique d'une solution

III.5.3 Fonction objective

Nous allons utiliser trois approches différentes :

• La première approche : consiste à évaluer les chromosomes avec la dominance forte de Pareto (détaillé dans chapitre 2 sous la section II.2.2) où chaque chromosome dominant doit avoir des valeurs de QOS meilleures que d'autres chromosomes. Cette approche donne un ensemble des solutions où le choix de solution finale est fait avec le choix aléatoire, nous avons nommé cette version d'algorithme « TS-GA Pareto Random Solution »

- fonction objective : Pareto

Algorithm 2 Fonction objective 1

1: **Input :**F1,ch1,ch2 : Chromosome
2: **Output :**F1 : Chromosome

$$F1 = \left\{ \begin{array}{l} critère1(ch1) < critère1(ch2)\ et \\ critère2(ch1) > critère2(ch2)\ et \\ critère3(ch1) < critère3(ch2)\ et \\ critère4(ch1) < critère4(ch2)\ et \\ critère5(ch1) > critère5(ch2) \end{array} \right\}$$

3:

• La deuxième approche : consiste à faire une somme pondérée (détaillée dans le chapitre 2 la sous la section I.4.2) des fonctions présentées précédemment où chaque fonction est associée à un poids selon son importance. La somme des poids doit être égale à 1. Dans nos simulations, nous avons des poids équitables pour chaque métrique, on a nommé cette version d'algorithme « TS-GA Agregation Solution »

- fonction objective : La somme pondérée

Algorithm 3 Fonction objective 2

1: **Input :**F2,poids1,poids2,poids3,poids4,poids5 : Réel
2: **Output :**F2 : Réel

$$F2 = \left\{ \begin{array}{l} poids1 * critère1+ \\ poids2 * critère2+ \\ poids3 * critère3+ \\ poids4 * critère4+ \\ poids5 * critère5 \end{array} \right\}$$

3:

• La troisième approche : consiste à hybrider les deux approches détaillées au-dessus, on a utilisé la fonction Pareto pour trouver l'ensemble des solutions après on applique la somme pondérée pour trouver la solution finale ,nous avons nommé cette version d'algorithme « TS-GA Pareto Agregation Solution »

III.5.4 Opérateurs génétiques

Les opérateurs génétiques représentent le cœur d'un algorithme génétique. L'utilisation de tels opérateurs permet de générer de nouvelles solutions à partir de celles déjà existantes.

III.5.4.1 Processus de sélection

L'algorithme $TS-GA$, utilise la sélection par tournoi qui consiste à comparer une paire des chromosomes choisis au hasard pour la reproduction (crossover) , donc un nombre aléatoire r est choisi entre 0 et 1.
Si $r < k$ (où k est un paramètre par exemple $0,75$) on sélectionne deux chromosomes aléatoirement pour avoir des parents et on ajoute les chromosomes non sélectionnés à la nouvelle population. L'avantage de cette méthode est d'éviter d'avoir le super chromosome qu'on peut rencontrer dans les autres méthodes de sélection.[1]

III.5.4.2 Processus de croisement

Dans l'algorithme $TS-GA$ deux chromosomes sélectionnés pour appliquer le processus de croisement à un seul point (détaillé dans le chapitre 2, section 5) pour générer deux descendants en utilisant un croissement à un point Ainsi, le croisement produit quatre enfants (voir figure III.2). Après cela, les deux meilleurs enfants sont choisis parmi ceux-ci. .[1]

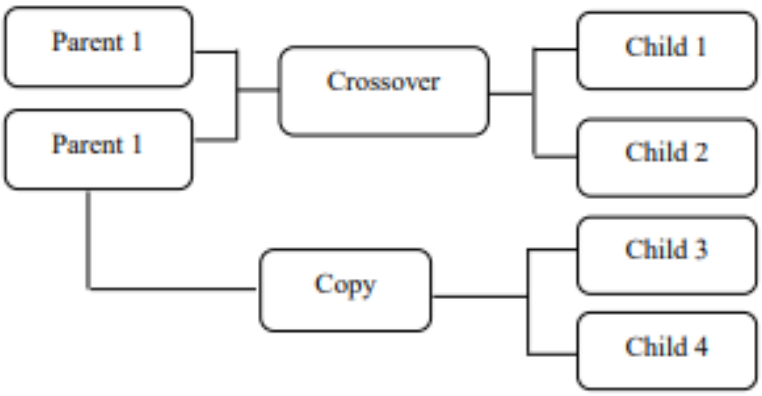

Figure III.2 – Processus de croisement de l'algorithme $TS-GA$

III.5.5 Initialiser la sous-population

Après chaque itération, les sous-populations (c'est-à-dire les nouvelles populations après le croisement) sont ajoutées aux anciennes populations (c'est-à-dire les parents). Cette étape peut améliorer la diversité de la population. Cette idée est introduite par l'algorithme $TS-GA$.[1]

III.5.6 Garder la meilleure solution

Dans l'algorithme génétique, Il existe une solution qui pourrait satisfaire la fonction objective, mais elle n'est pas sélectionnée pendant le processus de croisement. Par l'algorithme $TS-GA$, cette solution n'est pas supprimée de la population, mais elle est choisie et ajoutée à la population au démarrage de la prochaine itération. Cette étape est considérée comme bonne étape car certaines des itérations peuvent générer la meilleure solution.[1]

III.5.7 Critère d'arrêt

L'algorithme termine son exécution lorsque Le nombre de générations atteint une limite maximale [1]

III.6 Langage et environnement de développement

Le travail proposé dans ce mémoire a été implémenté et testé dans un environnement possédant les caractéristiques suivantes :

- Une machine avec un processeur Intel (R) Core (TM) i3-4030U CPU@ 1.80GHz, une vitesse de 2.40 Ghz et une capacité mémoire de 4GB. Le simulateur CloudSim est sous Windows 10 de 64 bits.
- Le simulateur CloudSim version 3.0.3.
- Le langage de programmation Java.
- L'IDE NetBeans.

III.6.1 Langage de programmation Java

C'est un langage de programmation informatique orienté objet, et en même temps un environnement d'exécution portable [66], développé par Sun Microsystems. Il est simple, robuste, dynamique et sécurisé, indépendant de la plateforme matérielle. [17]

III.6.2 Environnements de développement

L'éditeur que nous avons utilisé est NetBeans, qui est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL. En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend

toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web). NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X et Open VMS.[14]

III.6.3 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.[29]

III.7 Diagramme de classe cloudsim

Le simulateur ClouSim est composé de plusieurs classes (figure III.3). Parmi les classes fondamentales qui forment les blocs constitutifs du simulateur CloudSim, nous pouvons citer :

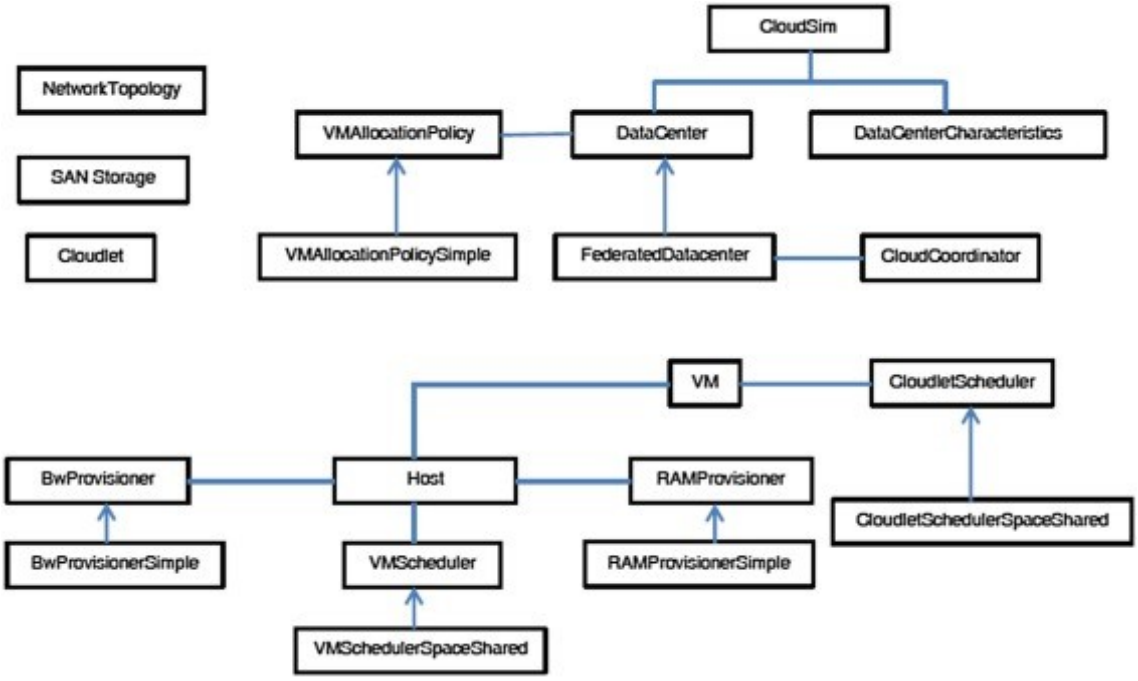

Figure III.3 – Diagramme de classe toolkit CloudSim [7]

A DataCenter :

Cette classe modélise l'infrastructure du noyau du service (matériel, logiciel) offert par des fournisseurs de ressources dans un environnement de Cloud Computing. Il encapsule un ensemble de machines de calcul qui peuvent être homogènes ou hétérogènes en ce qui concerne leurs configurations de ressources (mémoire, noyau, capacité, et stockage). En outre, chaque composant de Data Center instancie un composant généralisé d'approvisionnement de ressource qui implémente un ensemble de politiques d'allocation de bande passante, de mémoire, et de dispositifs de stockage .[7]

B DatacenterBroker :

Cette classe modélise le courtier (Broker) qui est responsable de la médiation entre les utilisateurs et les prestataires de service selon les conditions de QoS des utilisateurs et déploie les tâches de service à travers les Clouds. Le Broker agissant au nom des utilisateurs et identifie les prestataires de service appropriés du Cloud par le service d'information du Cloud noté par CIS (Cloud Information Services) et négocie avec eux pour une allocation des ressources qui répond aux besoins de QoS des utilisateurs.[7]

C SANStorage : Cette classe modélise un réseau de zone de stockage qui est généralement capable, pour les Data Center, de stocker une large quantité des données. SANStorage met en application une interface simple qui peut être employée pour simuler le stockage et la recherche de n'importe quelle quantité de données, à tout moment.[7]

D Host :

Cette classe modélise une ressource physique comme le serveur de stockage ou de calcul. Elle encapsule des informations importantes, telles que la quantité de mémoire et de stockage, le type de cœurs de traitement (pour représenter une machine multi-core), une politique d'allocation pour le partage de la puissance du traitement entre les machines virtuelles et les politiques d'attribution de mémoire et de bande passante aux machines virtuelles.[7]

E VirtualMachine :

Cette classe modélise une instance de machine virtuelle (VM), dont la gestion pendant son cycle de vie est une responsabilité de la machine (Host). Un Host peut simultanément instancier de multiples VMs et assigner des cœurs à base de politiques prédéfinies de partage de processeur (espace partagé, temps partagé).[7]

F Cloudlet :

Cette classe modélise les services d'application de base du Cloud (la livraison, la gestion de réseau social, le déroulement des opérations d'affaires), qui sont généralement déployés dans des DataCenter.[7]

G BWProvisioner :

C'est une classe abstraite qui modélise la politique d'approvisionnement de la bande passante aux VMs qui sont déployés sur un composant Host. La fonction de ce composant est d'entreprendre l'attribution de la bande passante du réseau à l'ensemble des VMs déployées à travers le DataCenter. Les chercheurs et les développeurs système de Cloud peuvent étendre cette classe avec leurs propres politiques (priorité, QoS) pour refléter les besoins de leurs applications.[7]

H VMProvisioner :

Cette classe abstraite représente la politique de provisionnement qu'un Moniteur de VM utilise pour allouer les VMs aux Hôtes. La fonctionnalité chef de la VMProvisioner est de sélectionner l'hôte disponible dans un DataCenter, qui répond à la mémoire, stockage, et l'exigence de disponibilité pour le déploiement de VM. L'implémentation par défaut du SimpleVMProvisioner fournie avec le paquet CloudSim alloue des VM au premier hôte disponible qui répond aux exigences précitées.[7]

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).[25]

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).[25]

III.8 Architecture de CloudSim

La structure logicielle de CloudSim et ses composants sont représentés par une architecture en couches comme indiqué dans la figure III.4 Les premières versions de CloudSim utilisent SimJava, un moteur de simulation d'événement discret qui met en œuvre 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.[7]

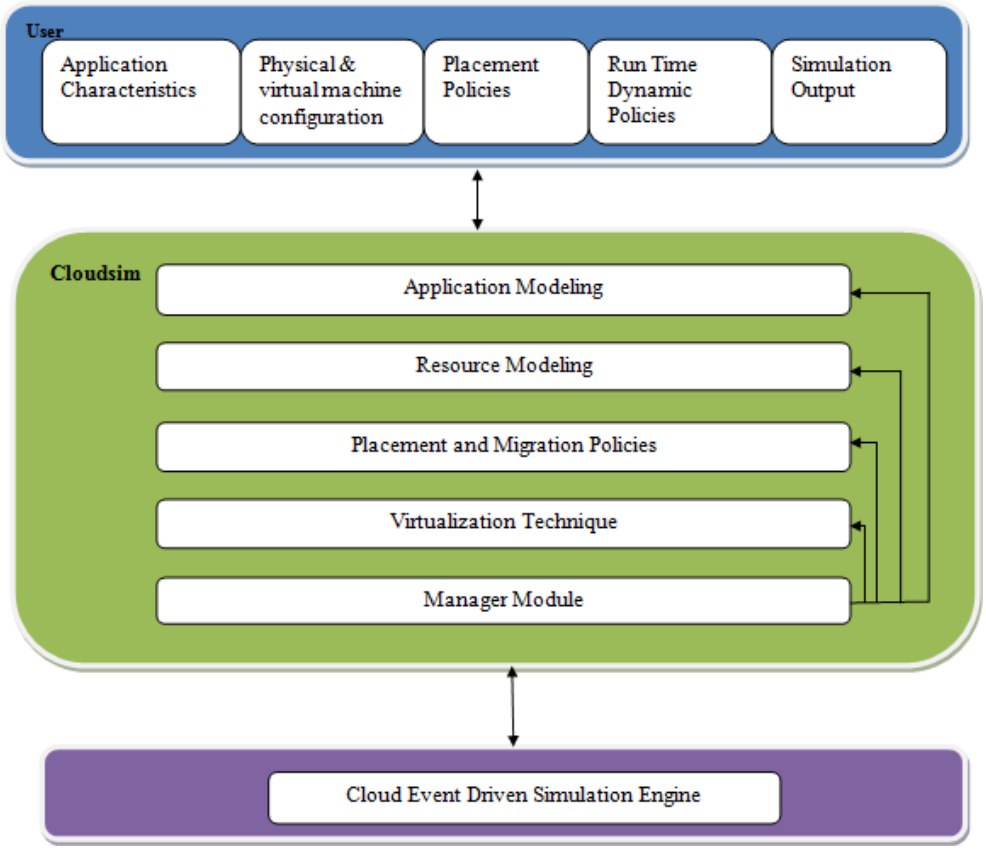

Figure III.4 – Architecture de toolkit CloudSim

III.9 Implémentation

1. **La configuration des datacenters :** La figure III.5 illustre la page d'accueil de notre application, l'utilisateur se retrouvera devant la fenêtre de configuration des datacenters, l'utilisateur doit remplir les informations nécessaires pour la création de datacenter, le nombre de datacenter , l'utilisateur doit passer par les étapes suivantes :

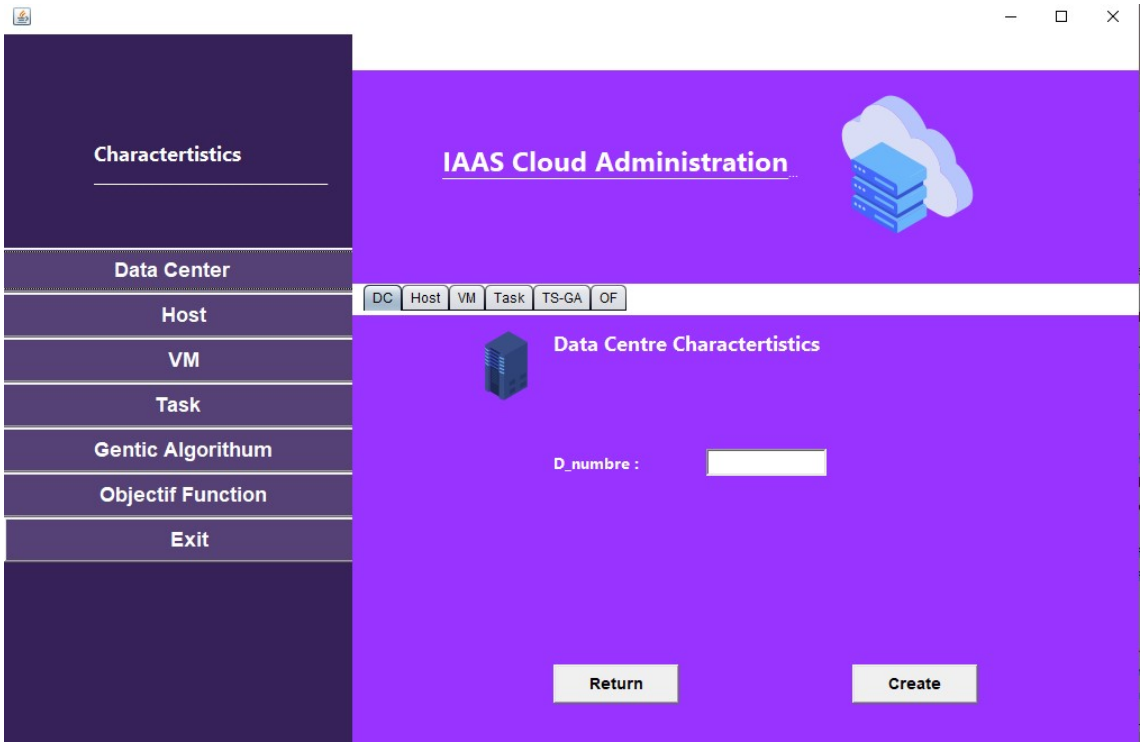

Figure III.5 – Fenêtre de configuration des data centres

2. **La configuration des machines physiques :** nous commençons par saisir les paramètres nécessaires aux machines physiques comme le nombre des hôtes qui se trouvent dans chaque (MIPS), la RAM en MB, la bande passante Mbit/s et aussi la capacité de stockage MB.

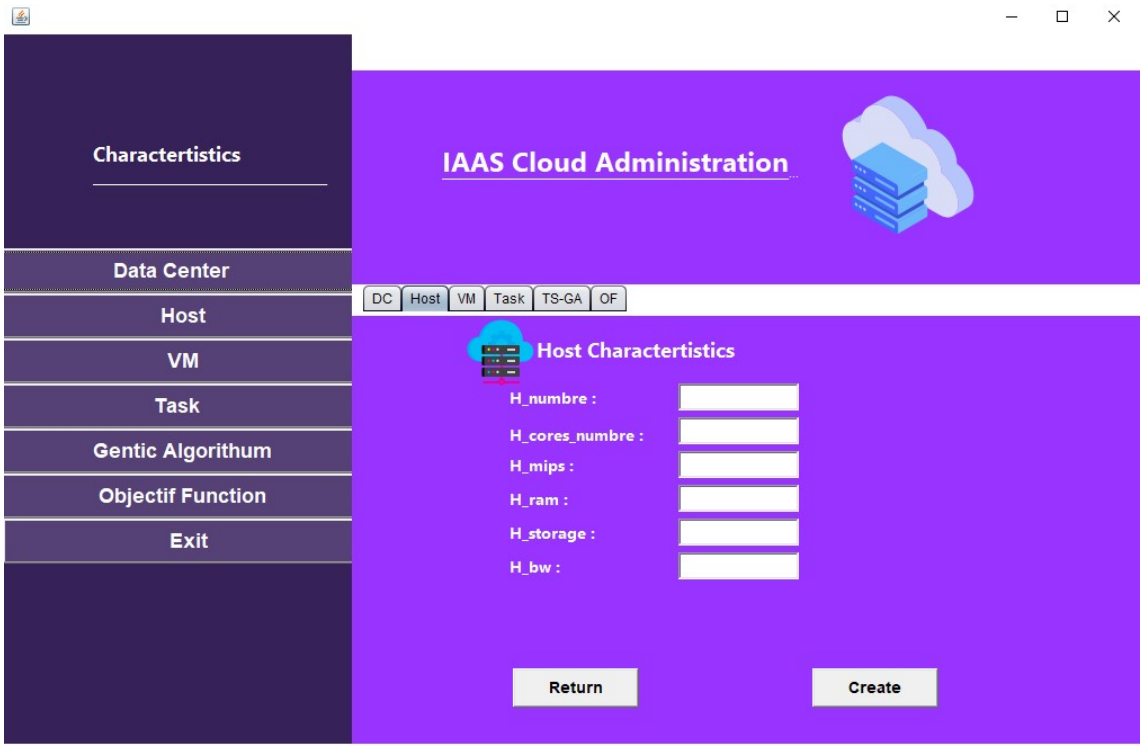

Figure III.6 – Fenêtre de configuration des machines physiques

3. **La configuration des machines virtuelles :** Une fois que nous avons configuré les différentes caractéristiques des machines physiques, nous devrons configurer les machines virtuelles VMs. Une machine virtuelle est caractérisée par sa vitesse de traitement (MIPS), son coût d'utilisation, son taux d'échec, sa capacité de RAM, sa bande passante ainsi que son stockage comme indiqué dans la figure III.7 .

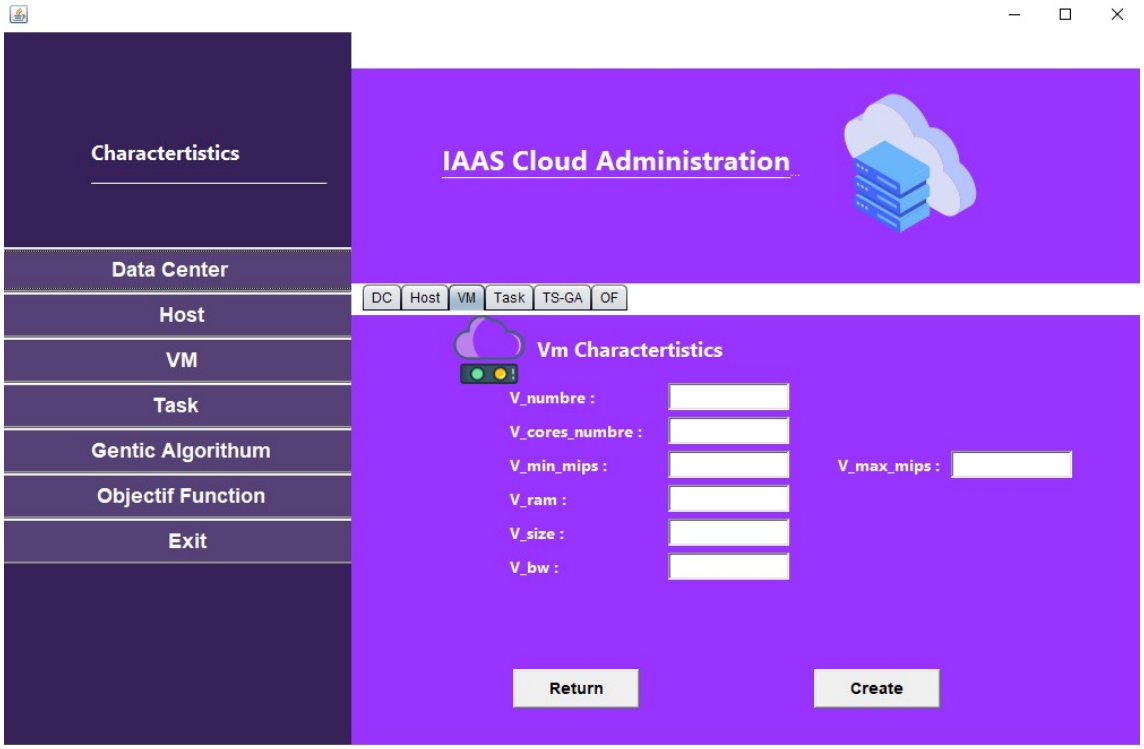

Figure III.7 – Fenêtre de configuration des machines virtuelles

4. **La configuration des tâches :** Cette fenêtre permet de définir les caractéristiques des tâches tells que le nombre de tâches, le nombre de cœurs (PE, la taille de la tâche, fichier de la tâche en entrée, fichier de la tâche en sortie.

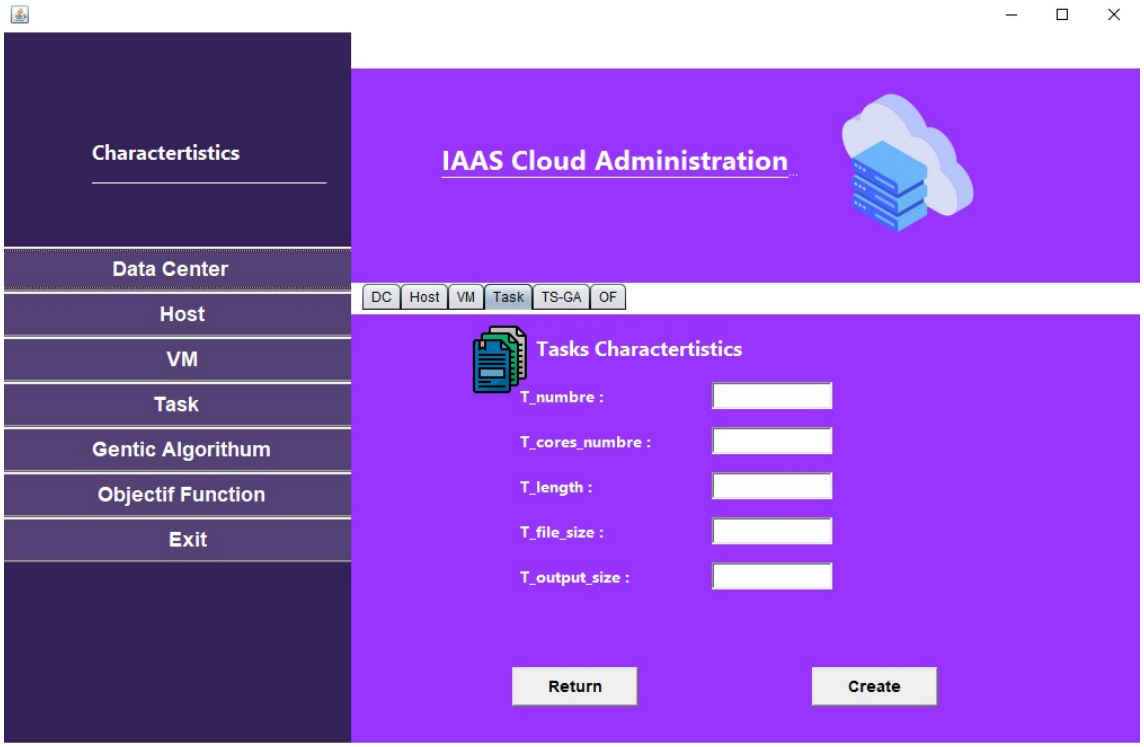

Figure III.8 – Fenêtre de configuration des tâches

5. **La configuration d'algorithme TS-GA :** cette fenêtre permet la configuration d'algorithme génétique comme illustré dans la figure III.9 , en remplissant les champs suivants : taille de population, nombre d'itérations et paramètres de sélection

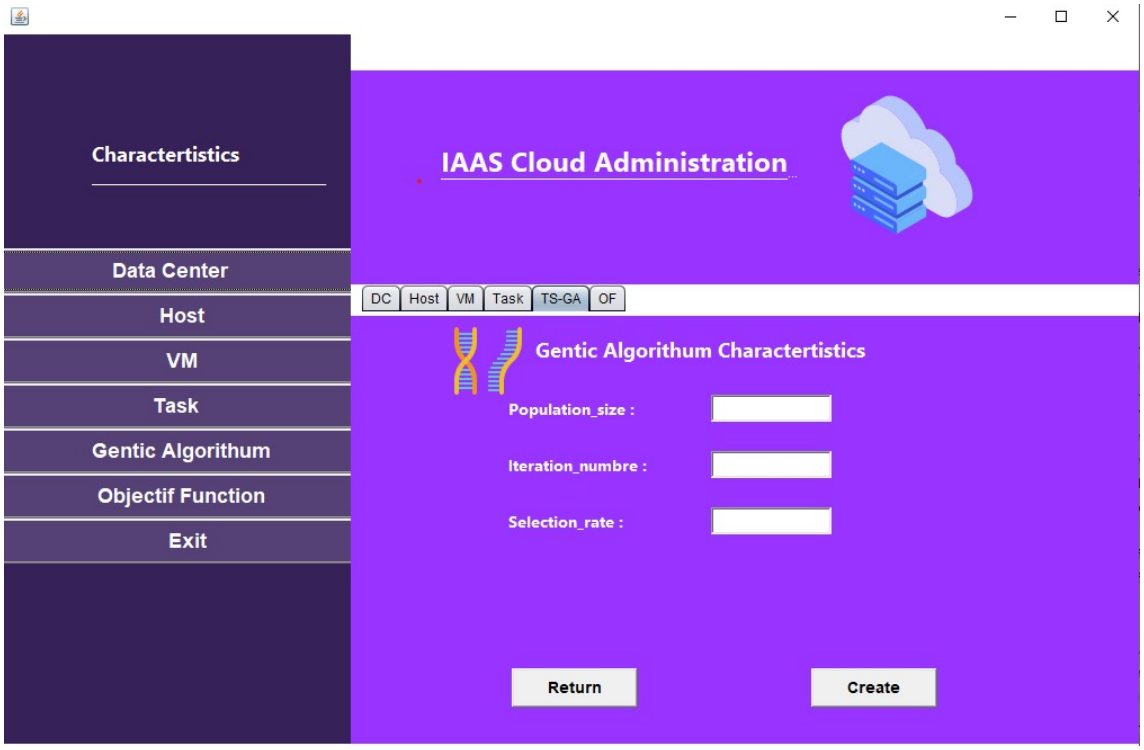

Figure III.9 – Fenêtre de configuration les paramètres de l'algorithme $TS - GA$

6. **La configuration de la fonction objective :** cette fenêtre permet la configuration de la fonction objective somme pondérée comme illustré dans la figure III.10 .

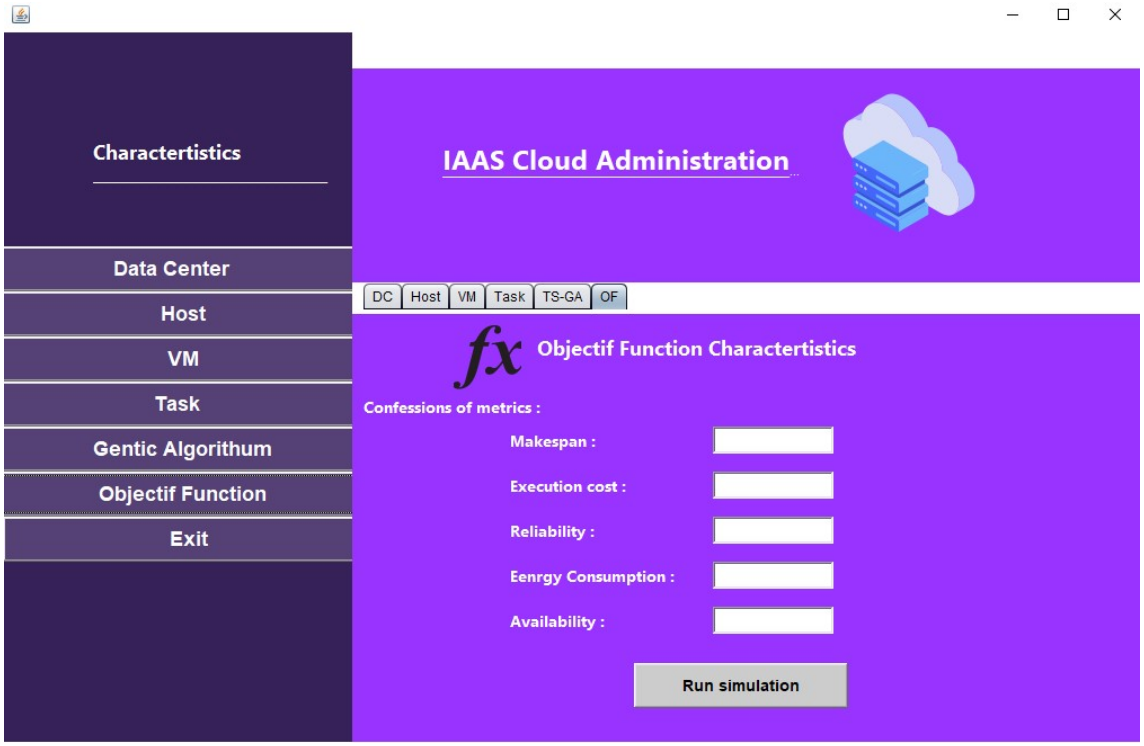

Figure III.10 – Fenêtre de configuration les coefficients de la fonction objectif

III.10 Paramètres expérimentaux

Les paramètres expérimentaux concernent à la fois le modèle d'infrastructure IaaS du fournisseur, ainsi que les paramètres de l'algorithme $TS - GA$.

III.10.1 Paramètres de l'environnement IAAS

— **Les Caractéristiques des Data centres :**

Dc	Nombre d'hôtes	Prix Cpu ($/seq)	Prix BW ($/seq)	Prix Ram ($/seq)	Prix stockage ($/seq)
Dc1	4	3	0.1	0.05	0.1
DC2	4	3	0.1	0.05	0.1

Tableau III.1 – Caractéristiques des data centres

— **Les Caractéristiques des machines physiques :**

Nombre d'Hôtes	Cœurs/cpu	Mips	Ram	Bande passante	Stockage	Politique
8 hôtes	1	10000	204800 MB	10000 Mb/s	10000 MB	TimeShared

Tableau III.2 – Caractéristiques des machines physiques

— **Les caractéristiques des machines virtuelles :**

Type Vm	Cœurs/cpu	Mips	Ram	Bande passante	Size
Type 1	1	100	128 MB	10 Mb/s	10000 MB
Type 2	1	500	512 MB	50 Mb/s	10000 MB
Type 3	1	1000	1024 MB	100 Mb/s	10000 MB

Tableau III.3 – Caractéristiques des machines virtuelles

— **Les caractéristiques des tâches :**

Longueur	Cœurs/cpu	Taille fiche	Taille sortie	Model d'utilisation
$1000 + R$	1	300	300	full

Tableau III.4 – Caractéristiques des tâches :

— **Les paramètres d'algorithme génétique TS-GA :**

Les différents paramètres de l'algorithme TS-GA sont décrits dans le Tableau III.5

Paramètre	Valeur
Taille de population	10
Nombre de générations	20
Taux de sélection	0.5

Tableau III.5 – Paramètres de l'algorithme $TS - GA$

— **Les tests de simulation :**

Pour chaque comparaison entre algorithmes on a établie 6 tests :

Test	Nombre de tâches	Nombre de vm
1	100	20
2	500	20
3	1000	20
4	100	30
5	500	30
6	1000	30

Tableau III.6 – Les tests de simulation

III.11 Résultats et discussion

III.11.1 Intérêt de la fonction objective agrégeant plusieurs métriques de QoS

Notre travail de recherche traitant l'ordonnancement des tâches , qui prend en compte, à la fois, les cinq métriques de qualités de services (makespan, coûts, fiabilité et disponibilité,consommation d'énergie) , et la configuration hétérogène de l'IaaS considérée. Dans ce chapitre nous comparons les algorithmes, nous proposons le processus d'évaluation suivant :

— Nous appliquons l'algorithme $TS-GA$, qui utilise la fonction objective agrégeant les différentes métriques de QoS et nous extrayons la valeur de chaque métrique de QoS dans la solution trouvée.

— Enfin, nous comparons les résultats obtenus pour chaque métrique de QoS par l'algorithme $TS-GA$ avec son meilleur résultat obtenu par l'optimisation multiobjectif.

Nous effectuons également plusieurs simulations, en faisant varier les poids des différentes QoS.

Optimisation Multiobjectif	Makespan (s)	Fiabilité (%)	Coût ($)	Consomation d'énergie (w/h)	Disponibilité (%)
TS-GA(0.2 0.2 0.2 0.2 02)	238,95	**0,999969775**	**9082,9425**	**235987,9375**	12,64970705
TS-GA(0.6 0.1 0.1 0.1 0.1)	215,88	0,999950522	14859,3315	381262,7875	22,92065268
TS-GA(0.7 0.075 0.075 0.075 0.075)	**154,6**	0,999963474	10973,166	286406,65	**23,62692109**
Meilleures valeurs	154,6	0,999969775	9082,9425	235987,9375	23,62692109

Tableau III.7 – Les valeurs des QoS pour la fonction objective somme pondérée

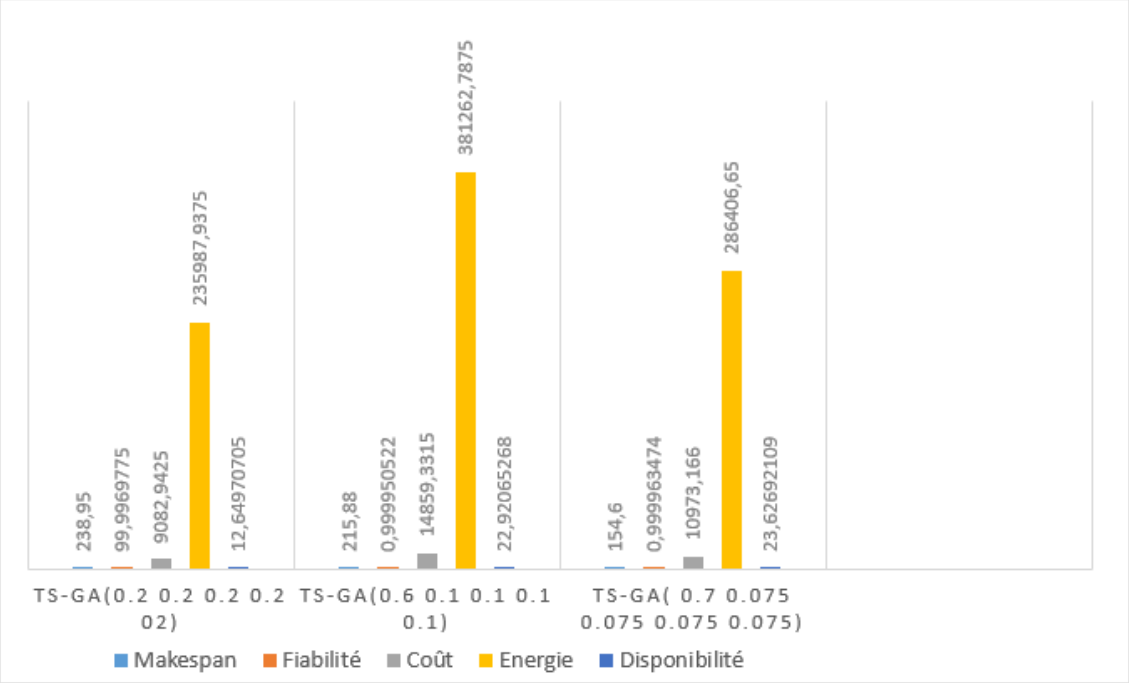

Figure III.11 – Résultats de la variation des poids de la fonction somme pondérée

III.11.2 Discussion

D'après la figure III.11 , nous pouvons constater que l'algorithme $TS-GA$ donne de bons résultats d'ensemble en utilisant les valeurs de poids de 0,2 pour chaque métrique de QoS sauf makespan et disponibilité (ceci est représenté par la configuration $TS-GA$ (0.2 0.2 0.2 0.2 0.2).

Un poids important est donné au makespan afin de voir son impact sur les autres métriques de QoS. Les résultats montrent que la réalisation d'un bon makespan, conduit à une dégradation significative des autres métriques de QoS. Renforcer le poids du makespan dans la configuration ($TS-GA$ (0.6 0.1 0.1 0.1 0.1)) et la configuration ($TS-GA$ (0.7 0.075 0.075 0.075 0.075)) donne à nouveau de bons résultats .

Pour résumer, les résultats des expériences (avec les valeurs des poids équitables) donnent des bons résultats avec un score supérieur ou égal à 60 % . pour l'ensemble des métriques de QoS. Pour le reste de simulations on a utilisé des poids équitables (0.2 pour chaque métrique utilisée)

III.12 Comparaison entre les différentes versions de $TS - GA$

Pour la validation de l'ordonnancement multiobjectif, une comparaison a été réalisée en utilisant trois versions pour l'évaluation des solutions dans l'algorithme $TS - GA$, à savoir le principe de Pareto, la méthode de la somme pondérée en agrégeant les différentes métriques de QoS et la méthode hybride.

1. Makespan : Le tableau (III.8) et la Figure (III.12) et (III.13) représentent le makespan pour chaque version d'algorithme TS-GA proposé en utilisant 20 et 30 VM.

No of task	TS-GA Pareto random	TS-GA Pareto Agregation	TS- GA Agrégation	No Vm
100	116,064	128,96	133,58	20
500	507,33	519,04	705,34	
1000	969,76	1031,8	1021,11	
100	84,72	94,59	200,052	30
500	354,15	375,83	378,97	
1000	700,11	770,01	1102,38	

Tableau III.8 – Résultats d'optimisation de la métrique makespan

pour 20 vm :

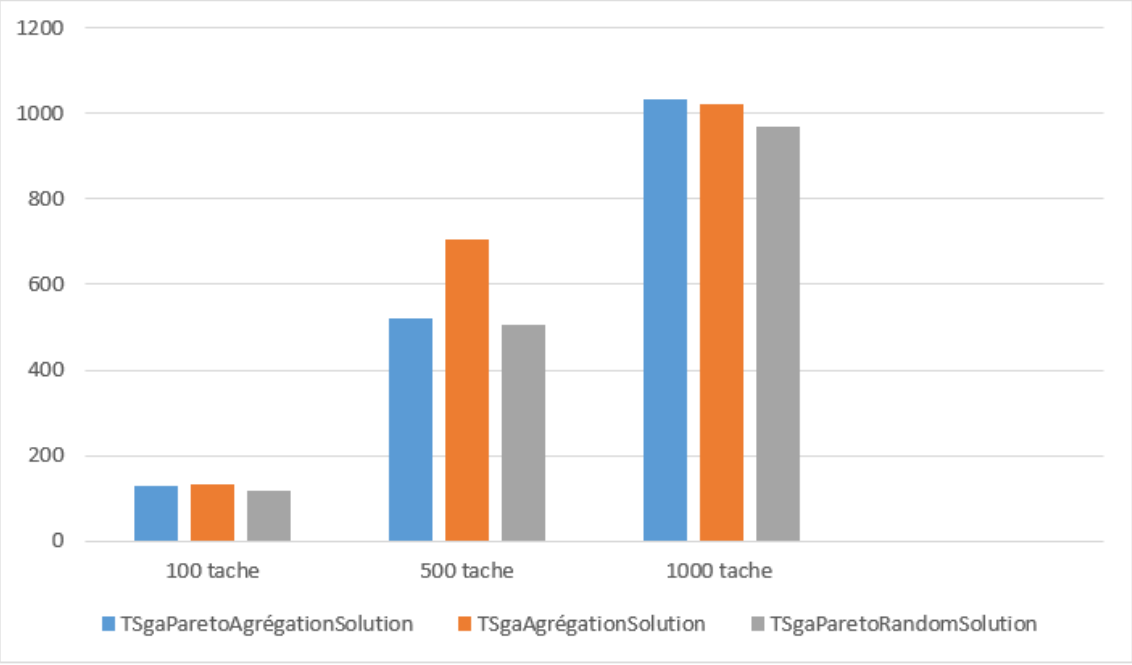

Figure III.12 – Résultats d'optimisation de la métrique makespan pour 20 Vm

pour 30 vm :

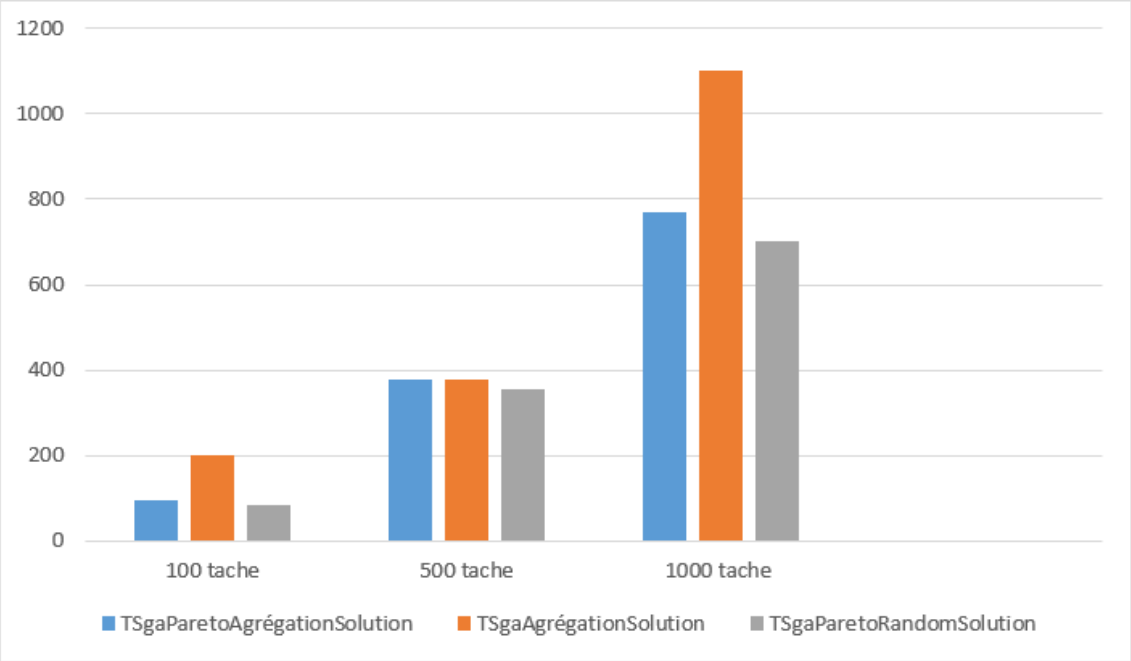

Figure III.13 – Résultats d'optimisation de la métrique makespan pour 30 Vm

Dans la Figure (III.12) (20 VM) et (III.13) (30 VM), nous pouvons voir les résultats de l'expérience sur l'axe des abscisses, est indiqué le makespan et l'axe des ordonnées indique le « nombre de tâches ».

Peu importe le nombre de VMs et le nombre de tâches, la meilleure façon d'optimiser le makespan est d'utiliser le $TS - GA$ Pareto Random.

2. Fiabilité :

   Le tableau (III.9) et la Figure (III.14) et (III.15) représentent la Fiabilité ou la probabilité que toutes les tâches soient entièrement exécutées avec succès en utilisant 20 et 30 VM.

No of task	TS-GA Pareto random	TS-GA Pareto Agregation	TS- GA Agrégation	No Vm
100	99,9971	99,9968	99,9965	20
500	99,9863	99,9859	99,9850	
1000	99,9748	99,9742	99,9728	
100	99,9982	99,9980	99,9967	30
500	99,9818	99,98157	99,98862	
1000	99,99798	99,99791	99,9787	

Tableau III.9 – Résultats d'optimisation de la métrique fiabilité

pour 20 vm :

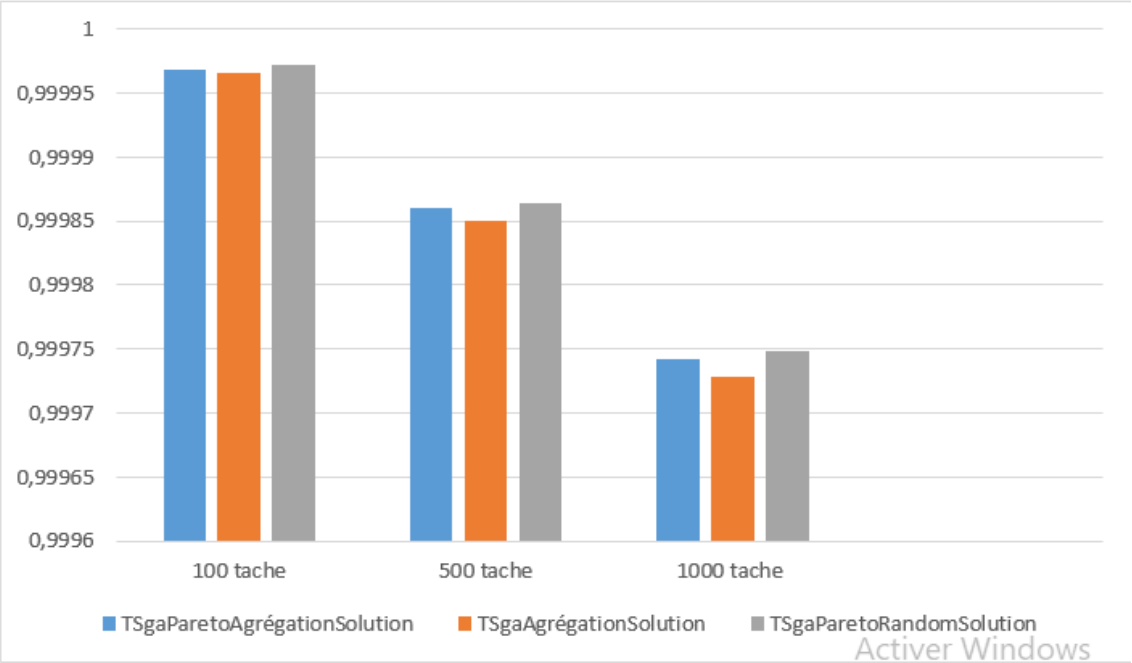

Figure III.14 – Résultats d'optimisation de la métrique fiabilité pour 20 Vm

pour 30 vm :

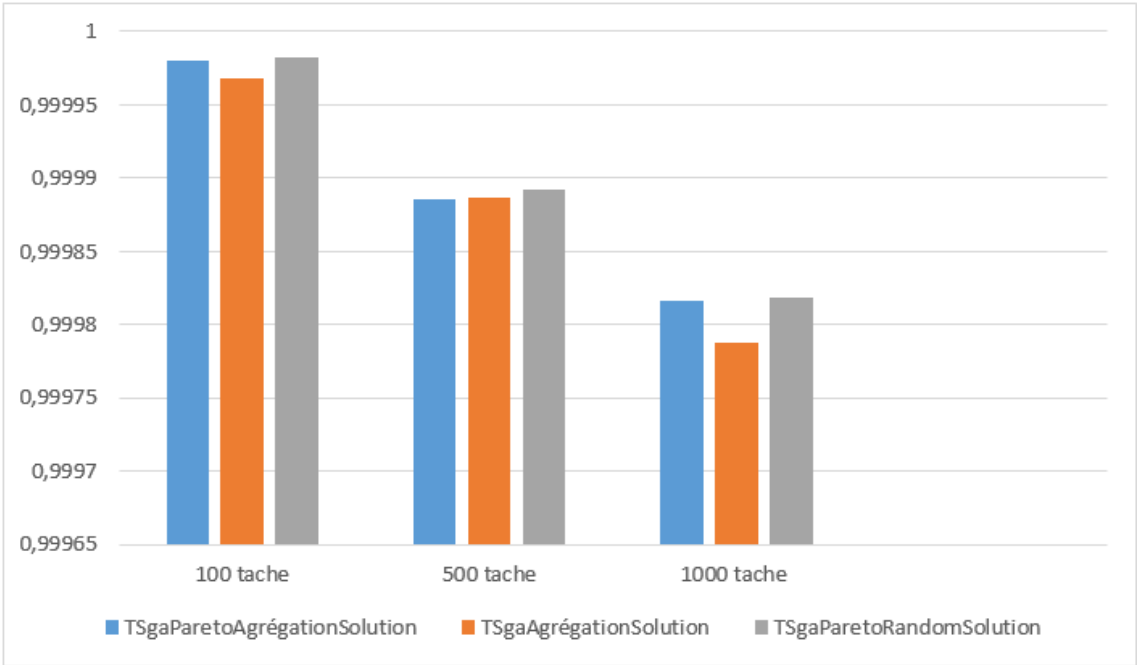

Figure III.15 – Résultats d'optimisation de la métrique fiabilité pour 30 Vm

Peu importe le nombre de VMs et le nombre de tâches, nous apercevons que la différence entre les méthodes en termes de fiabilité requise n'est pas flagrante, un léger avantage de la méthode TS-GA Pareto Random Les machines virtuelles sont considérées comme fiables puisque les fournisseurs Cloud existant proposent des services avec un taux d'échec trop petit.

3. Coût d'exécution :

   Le tableau (III.10) et la figure (III.16) et (III.17) représentent le Coût d'exécution de toutes les tâches sur les machines virtuelles disponibles en utilisant 20 et 30 VM.

No of task	TS-GA Pareto random	TS-GA Pareto Agregation	TS- GA Agrégation	No Vm
100	8519,37	9495,3615	10260,9585	20
500	204857,12	210469,78	223831,49	
1000	754278,94	774091,97	816012,39	
100	5311,80	5945,40	9615,95	30
500	162730,36	171001,13	170726,32	
1000	544494,71	553139,83	638879,71	

Tableau III.10 – Résultats d'optimisation de la métrique coût d'exécution

pour 20 vm :

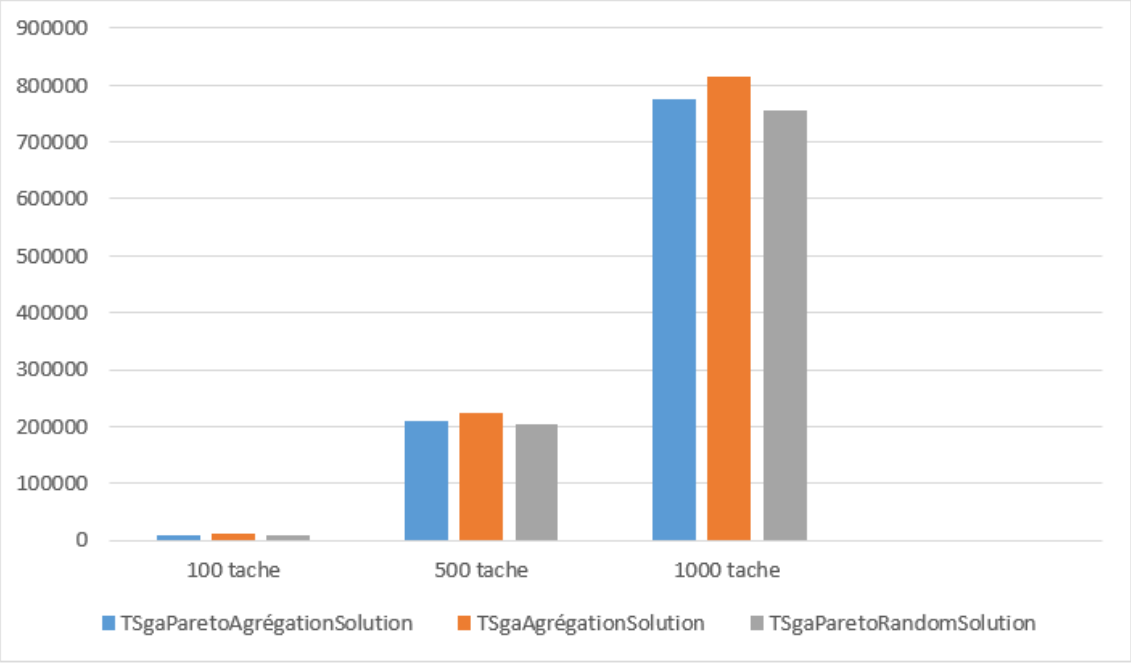

Figure III.16 – Résultats d'optimisation de la métrique coût d'exécution pour 20 Vm

pour 30 vm :

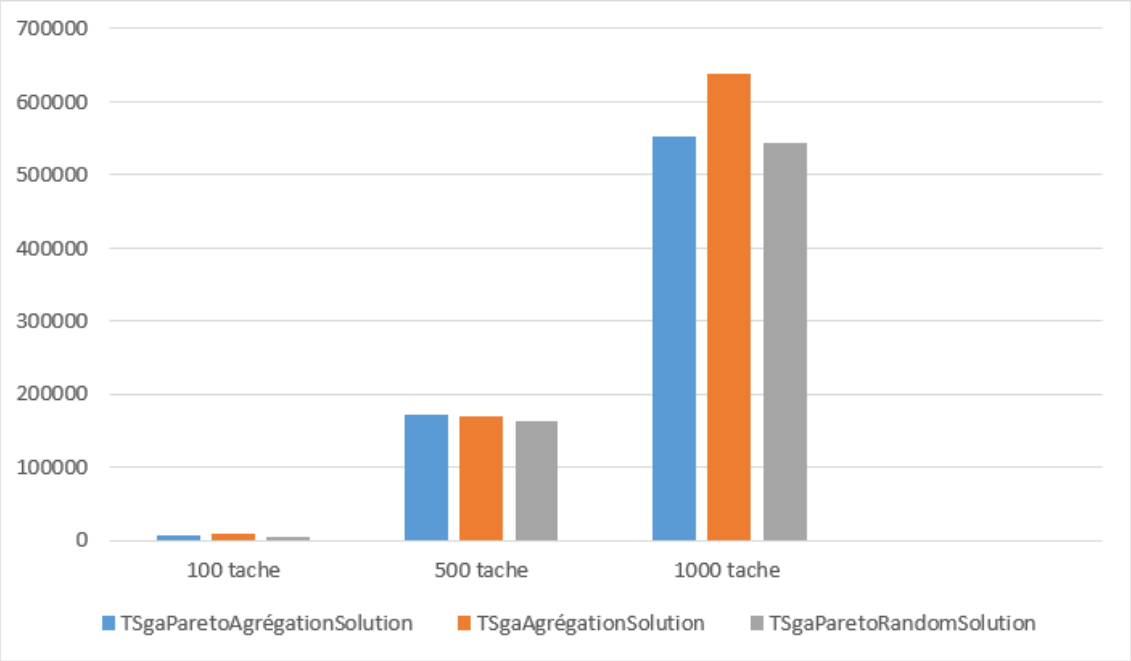

Figure III.17 – Résultats d'optimisation de la métrique Coût d'exécution pour 30 Vm

Peu importe le nombre de VMs et le nombre de tâches, la meilleure façon d'optimiser le coût est d'utiliser la méthode TS-GA Pareto Random.

4. Disponibilité

   Le tableau (III.11) et la figure (III.18) et (III.19) représentent la disponibilité des ressources pendent l'exécution des tâches en utilisant 20 et 30 VM.

No of task	TS-GA Pareto random	TS-GA Pareto Agregation	TS- GA Agrégation	No Vm
100	24,42	24,50	25,56	20
500	13,45	13,51	10,57	
1000	25,92	25,00	26,63	
100	20,84	20,89	15,99	30
500	15,30	15,15	15,01	
1000	25,91	23,93	19,31	

Tableau III.11 – Résultats d'optimisation de la métrique disponibilité

pour 20 vm :

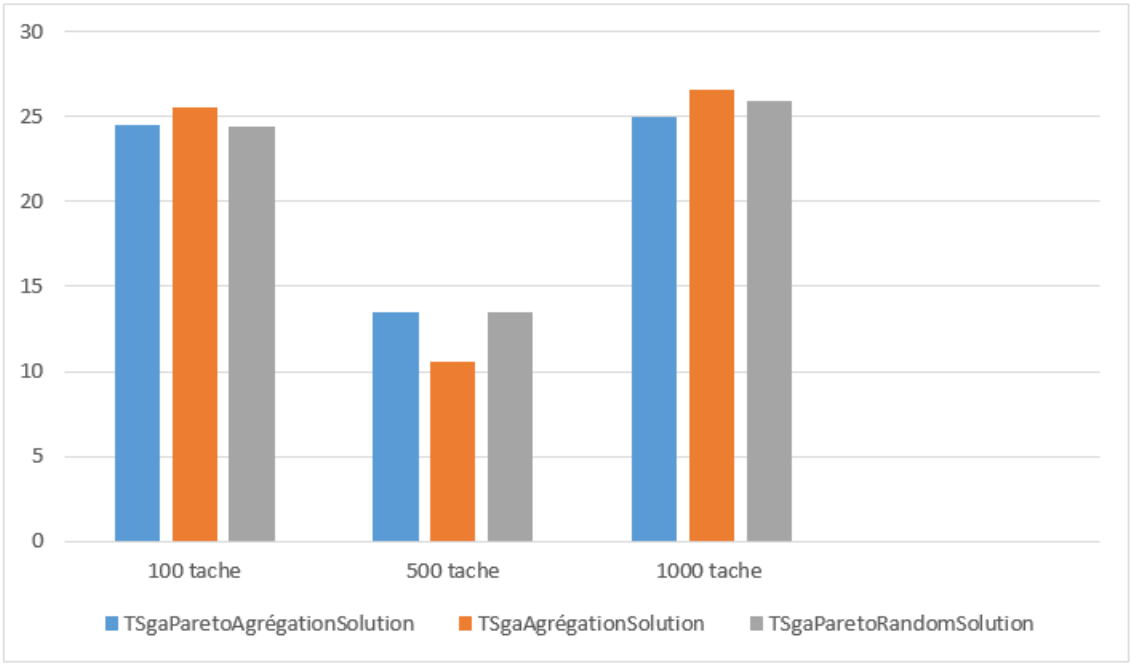

Figure III.18 – Résultats d'optimisation de la métrique disponibilité pour 20 Vm

pour 30 vm :

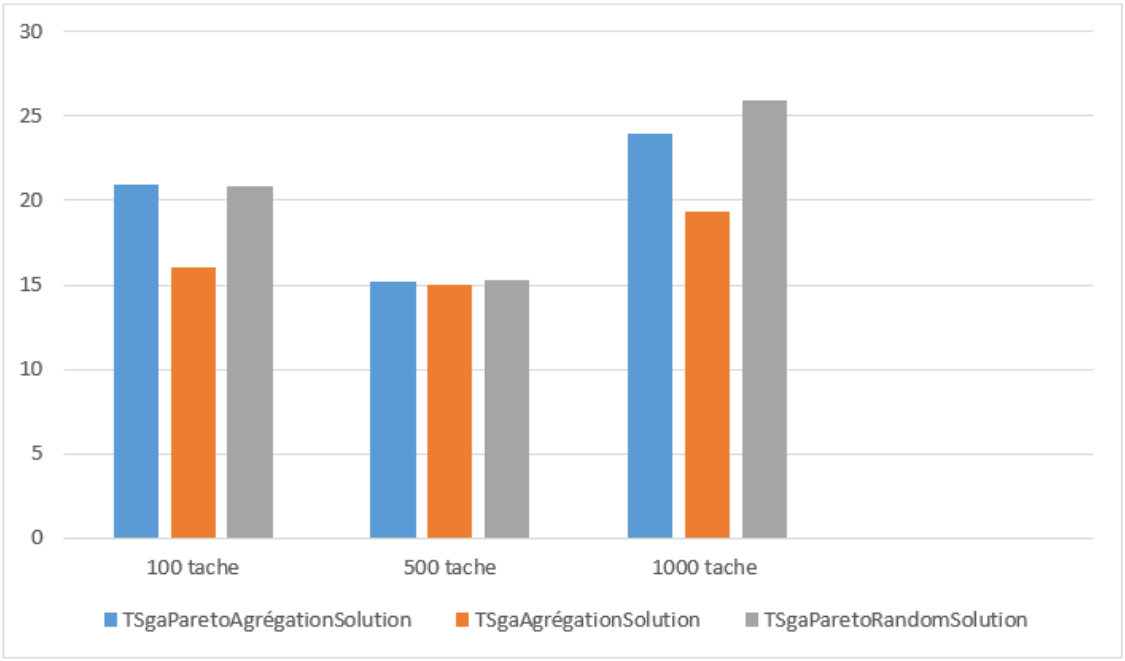

Figure III.19 – Résultats d'optimisation de la métrique disponibilité pour 30 Vm

Peu importe le nombre de VMs et le nombre de tâches, la meilleure façon d'optimiser la disponibilité est d'utiliser la méthode TS-GA Pareto Agreggation.

5. Consommation d'énergie :

   Le tableau (III.12) et la figure (III.20) et (III.21) représentent la consommation d'énergie nécessaire pour exécuter chaque ensemble des tâches en utilisant 20 et 30 VM.

No of task	TS-GA Pareto random	TS-GA Pareto Agregation	TS- GA Agrégation	No Vm
100	250423,03	226506,85	269389,71	20
500	5253799,7	5115678,76	5552911	
1000	19144124,38	18672063,7	20196143,5	
100	164145,43	148552,22	253272,47	30
500	4317424,67	4113365,33	4310161,91	
1000	13814743,3	13616090,42	15875147,91	

Tableau III.12 – Résultats d'optimisation de la métrique consommation d'énergie

pour 20 vm :

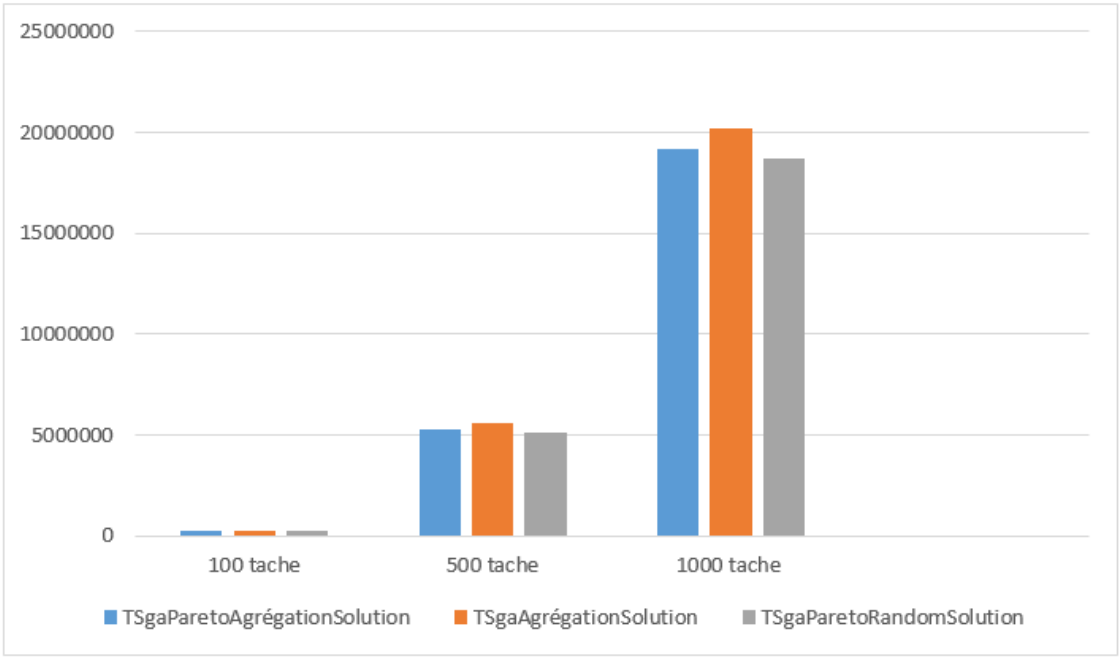

Figure III.20 – Résultats d'optimisation de la métrique consommation d'énergie pour 20 Vm

pour 30 vm :

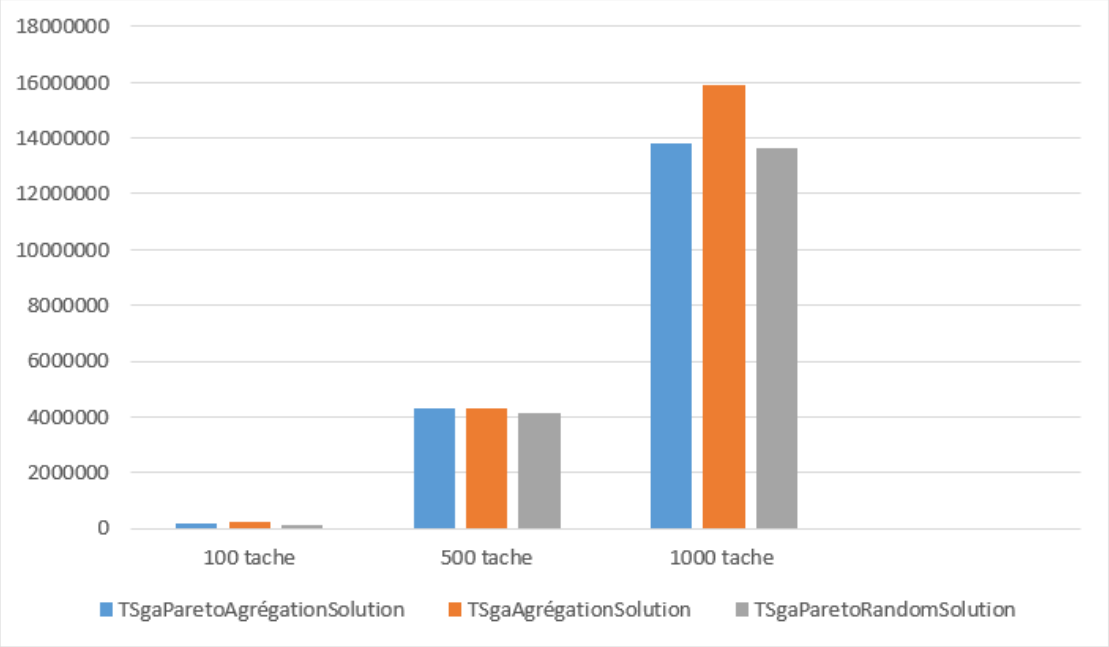

Figure III.21 – Résultats d'optimisation de la métrique consommation d'énergie pour 30 Vm

Dans cette dernière expérience et d'après Le tableau (III.12) et les Figures (III.20) et (III.21) nous remarquons que la version « TS-GA Pareto Random » est meilleur par rapport aux autres versions de l'algorithme TS-GA.

III.12.1 Discussion

Dans cette section, nous avons présenté les résultats expérimentaux d'algorithme génétique TS-GA pour traiter le problème d'ordonnancement des tâches utilisateur sur une infrastructure virtualisée de services IaaS.
Les résultats expérimentaux montrent qu'une approche purement multiobjectif comme le Pareto Random est plus recommandée dans ce contexte.

III.13 Conclusion

Ce chapitre a été consacré à la présentation de notre contribution dans le cadre de ce PFE. Nous avons présenté les outils de simulations utilisés, à savoir, le langage JAVA, l'IDE NetBeans et le simulateur CloudSim. Nous avons présenté également l'algorithme TS-GA avec les méthodes de calcul des différentes métriques de QoS notamment, le makespan, le coût, la fiabilité et la disponibilité et la consommation d'énergie.

Les résultats obtenus des différentes simulations effectuées montrent que notre algorithme TS-GA avec la fonction objective Pareto Random surpasse les deux autres versions de TS-GA Pareto agrégation et TS-GA agrégation et ceci, Peu importe le nombre de VM et le nombre de tâches.

Conclusion générale et perspectives

Le Cloud Computing fournit un modèle informatique, qui permet d'accéder d'une façon transparente et à la demande, à un pool de ressources hétérogènes physiques ou virtualisées (serveurs, stockage, applications et services), à travers le réseau. Ces ressources sont délivrées sous forme de services reconfigurables et élastiques, à base d'un modèle de paiement à l'usage dont les garanties sont offertes par le fournisseur, via des contrats de niveau de service SLA (Service Level Agreement).
L'exploitation efficace de ces services dans un environnement élastique sous contraintes souvent contradictoires pour les exploitants et les utilisateurs soulève plusieurs défis scientifiques.

Ce rapport s'intéresse particulièrement à la problématique d'ordonnancement des tâches indépendante dans le Cloud Computing. Cette problématique est considérée comme étant un problème d'optimisation multiobjectifs en raison de la nature conflictuelle des métriques de QoS à prendre en compte.
Nous avons développé des méthodes à base de métaheuristiques pour résoudre cette problématique, en nous focalisons essentiellement sur un modèle de service de Cloud Computing , notamment l'Infrastructure as a Service IaaS, nous avons utilisé l'algorithme génétique (TS-GA) pour le problème d'ordonnancement afin d'optimiser le makespan, le coût, la fiabilité, la disponibilité et la consommation d'énergie , en mettant en œuvre trois versions de la fonction objective à savoir Pareto Random, Pareto Agrégation et agrégation.

Les résultats expérimentaux obtenus montrent que l'algorithme TS-GA donne de bons résultats d'ensemble pour toutes les métriques de QoS. Les résultats obtenus sont plus intéressants pour TS-GA et la fonction objective Pareto Random

Les perspectives d'évolution de ce travail concernent différents points :

Les algorithmes d'ordonnancement basés sur l'approche de Pareto que nous avons proposés génèrent des ensembles de solutions non dominées de grande taille. La prochaine étape consiste à réduire cet ensemble, afin de permettre au décideur de mettre en place des méthodes interactives pour l'aide à la décision multicritère.

L'hybridation de nos algorithmes avec d'autres métaheuristiques ou des recherches locales, afin d'accélérer les convergences des algorithmes et d'améliorer la qualité des solutions obtenues.

L'exploitation des puissances de calcul fournies et le passage à l'échelle réelle dans un vrai Cloud permet d'envisager une parallélisation de nos algorithmes, afin d'améliorer la robustesse et de résoudre le problème d'ordonnancement multiobjectifs des tâches à très grande échelle.

Dans ce rapport, nous nous sommes concentrés sur l'ordonnancement statique des tâches. Il serait intéressant de prendre aussi en compte l'aspect dynamique de l'environnement cloud, afin d'établir le plan d'ordonnancement, à l'aide des informations du cloud mises à jour en temps réel.

Bibliographie

[1] Hamad A et Omara. « Genetic-based task scheduling algorithm in cloud computing environment ». In : *International Journal of Advanced Computer Science and Applications* (2016), p. 550-556. URL : https://www.researchgate.net/publication/301945977_Genetic-Based_Task_Scheduling_Algorithm_in_Cloud_Computing_Environment.

[2] Asli. « Approche hybride pour les problèmes d'optimisation combinatoire multiobjectif ». Thèse de doct. 2010, p. 44. URL : http://repository.usthb.dz/bitstream/handle/123456789/3073/TH5924.pdf?sequence=3&isAllowed=y.

[3] Avila. « Optimisation multiobjectif et analyse de sensibilité appliquées à la conception de dispositifs ». Thèse de doct. Ecully, Ecole centrale de Lyon, 2006. URL : https://tel.archives-ouvertes.fr/tel-00012065/document.

[4] Microsoft Azure. *« Qu'est-ce qu'une machine virtuelle ? »* URL : https://azure.microsoft.com/fr-fr/overview/what-is-a-virtual-machine/.

[5] Alain Berro. « Algorithmes évolutionnaires pour l'optimisation multi-objectif ». In : *Technique et Science Informatiques* 25.8-9 (2006), p. 991-1021. URL : http://www.laas.fr/files/MOGISA/sem_Alain-Berro.pdf.

[6] Alain Berro. « Optimisation multiobjectifs et stratégies d'évolution en environnement dynamique ». Thèse de doct. ANRT [diff.], 2001, p. 31-32. URL : http://www.univ-tebessa.dz/fichiers/laghouat/memoireb.pdf.

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

[8] Bouaynaya. « Usage du cloud computing dans les PME: Quelle imbrication acteurs / données? » In : 2017, p. 1-8. URL : https://www.researchgate.net/publication/317381293_Usage_du_cloud_computing_dans_les_PME%20_Quelle_imbrication_acteurs_donnees.

[9] Chinthagunta et Vidyamadhuri. « Cloud computing models: a survey ». In : *Adv. Comput. Sci. Technol.* (2017), p. 747-761. URL : https://www.ripublication.com/acst17/acstv10n5_09.pdf.

[10] Collette et Siarry. *Optimisation multiobjectif*. Editions Eyrolles, 2002. URL : https://www.researchgate.net/publication/276275719_Optimisation_Multiobjectif.

[11] Deb et al. « A fast and elitist multiobjective genetic algorithm: NSGA-II ». In : *IEEE transactions on evolutionary computation* (2002), p. 182-197. URL : https://sop.tik.ee.ethz.ch/publicationListFiles/zt1998a.pdf.

[12] Delinchamp. *« Définition Cloud Computing Documentation technique »*. 2017. URL : https://www.itnation.lu/content/uploads/2017/10/cloud_computing_-_definitions.pdf/.

[13] CGET DGE. *« Guide sur le Cloud Computing et les Datacenters à l'attention des collectivités locales »*. 2015. URL : https://www.entreprises.gouv.fr/files/files/directions_services/secteurs-professionnels/numerique/guide-cloud-computing-et-datacenters-2015.pdf/.

[14] Dad Djouhra. « Optimisation des performances des Data Centers des Cloud sous contrainte d'énergie consommée ». Thèse de doct. Université Ahmed Benbela, 2016-2017.

[15] El Dor. « Perfectionnement des algorithmes d'optimisation par essaim particulaire : applications en segmentation d'images et en électronique ». Thèse de doct. 2013, p. 21-31. URL : https://hal.archives-ouvertes.fr/tel-01167131/.

[16] Femmam. « Une approche formelle pour la planification des tâches pour la QoS dans le cloud-computing ». Thèse de doct. 2017, p. 40-45. URL : http://thesis.univ-biskra.dz/3654/1/th%C3%A9se.pdf.

[17] Forax. « Le langage Java ». In : (), p. 1-47. URL : http://www-igm.univ-mlv.fr/~forax/ens/java-avance/cours/pdf/old/I-%20Le%20langage%20Java.pdf.

[18] FRED. « Future paths for integer programming and links to ar tifi cial intelli g en ce ». In : *Computers operations research* (1986), p. 533-549. URL : ehttp://leeds-faculty.colorado.edu/.

[19] FRED. « Tabu search—part I ». In : *ORSA Journal on computing* (1989), p. 1-17. URL : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.302.4060&rep=rep1&type=pdf.

[20] FRED. « Tabu search—part II ». In : *ORSA Journal on computing* (1990), p. 1-29. URL : http://leeds-faculty.colorado.edu/glover/.

[21] GOLDBERG et HOLLAND. « Genetic algorithms and machine learning ». In : (1988), p. 95-99. URL : https://deepblue.lib.umich.edu/bitstream/handle/2027.42/46947/10994_2005_Article_422926.pdf.

[22] GREVET. « Le cloudcomputing: évolution ou révolution ». Thèse de doct. 2009, p. 15. URL : http://www.nicolasgrevet.com/files/mr09_ngrevet_cloudcom.pdf.

[23] Romain HENNION, Hubert TOURNIER et Eric BOURGEOIS. *« Cloud computing: décider, concevoir, piloter, améliorer »*. Editions Eyrolles, 2012. URL : https://www.eyrolles.com/Informatique/Livre/cloud-computing-9782212134049/.

[24] HOLLAND et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992, p. 89-112. URL : https://books.google.dz/books?hl=fr&lr=&id=5EgGaBkwvWcC&oi=fnd&pg=PR7&dq=Adaptation+in+%20natural+and+artificial+systems+&ots=mInr3YJsyq&sig=AxYrjgRHNeP1tRWOBcybr_HkcE8&redir_esc=%20y#v=onepage&q=Adaptation.

[25] Djebbar Esma INSAF. « optimisation d'ordonnancement et d'allocation de ressources dans les cloud computing ». Thèse de doct. Université d'Oran 1- Ahmed Ben Bella, 2016, p. 23-44. URL : https://theses.univ-oran1.dz/document/15201701t.pdf.

[26] Centre IT INTEL. *« Guide pratique Virtualisation et cloud computing »*. 2013. URL : http://www.asprom.com/dossier/intel1.pd.

[27] Ostermannand IOSUP et al. « A performance analysis of EC2 cloud computing services for scientific computing ». In : *International Conference on Cloud Computing*. 2009, p. 115-131. URL : https://www.researchgate.net/publication/221366062_A_Performance_Analysis_of_EC2_Cloud_Co%20mputing_Services_for_Scientific_Computing.

[28] Abraham Jerrin et al. « Genetic Algorithm For Solving The Uncapacitated Facility Location Problem ». Thèse de doct. 2013, p. 2278-0181. URL : https://www.researchgate.net/publication/230521714_A_Genetic_Algorithm_for_the_Uncapacitated_Facility_Location_Problem.

[29] Khaled M Khalil et al. « CLOUD SIMULATORS–AN EVALUATION STUDY ». In : *International Journal "Information Models and Analyses"* (2017), p. 1-23. URL : http://www.foibg.com/ijima/vol06/ijima06-01-p01.pdf.

[30] Lachlan, e Hutchings et Russel. *« Cloud Infrastructure Security & Compliance with GoGrid: A GoGrid White Paper addressing Cloud Infrastructure Security and Compliance Concerns for Large, Medium and Small Businesses »*. 2010. URL : https://cloud.report/Resources/Whitepapers/4d3c4e73-0b19-47f6-8184-29d41da1d60f_119.pdf/.

[31] Fonseca M, Fleming et al. « Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization. » In : *Icga*. 1993, p. 416-423. URL : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.9077&rep=rep1&type=pdf.

[32] Mahdi. « Optimisation multiobjectif par un nouveau schéma de coopération méta/exacte ». Thèse de doct. 2000, p. 16. URL : https://bu.umc.edu.dz/theses/informatique/MAH4857.pdf.

[33] Malik et al. « Comparison of Task Scheduling Algorithms in Cloud Environment ». In : *International Journal of Advanced Computer Science and Application* (2018), p. 1-7. URL : https://thesai.org/Downloads/Volume9No5/Paper_50-%20Comparison_of_Task_Scheduling_Algorithms.pdf.

[34] Fadila Mehidid. « Algorithme Génétique ». Thèse de doct. Cergy-Pontoise, 2013, p. 13-23. URL : https://hal.archives-ouvertes.fr/tel-01167131/.

[35] Peter Mell, Tim Grance et al. « The NIST definition of cloud computing ». In : (2011), p. 1-2. URL : https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[36] Meslem et Debbas. « Ordonnancement et réplication de données dans le Cloud Computing ». Thèse de doct. 2017. URL : https://docplayer.fr/112124283-Ordonnancement-et-replication-de-donnees-dans-le-cloud-computing.html.

[37] Noman et Aqeelur. « A comparative study of major service providers for cloud computing ». In : *proceedings of 1st International Conference on Information and Communication Technology Trends, At Karachi, Pakistan*. 2014, p. 5. URL : https://www.researchgate.net/publication/258489864_A_Comparative_Study_of_Major_Service_Providers_for_Cloud_Computing.

[38] Pandaba, Prafulla et Ray. « Modified Round Robin Algorithm for Resource Allocation in Cloud Computing ». Thèse de doct. 2016, p. 870-898. URL : https://www.sciencedirect.com/science/article/pii/S1877050916306287.

[39] Benabid Rabah. « Optimisation Multiobjectif de la Synthèse des factsFACTS par les Particules en Essaim pour le Contrôle de la Stabilité de Tension des Réseaux Electriques ». Thèse de doct. Université de Laghouat-Amar Telidji, 2007. URL : http://www.univ-tebessa.dz/fichiers/laghouat/memoireb.pdf.

[40] Rackspace. *« Public Cloud Hosting, Computing Storage and Networking by Rackspace »*. URL : https://www.rackspace.com/cloud//.

[41] Redhat. *« Un fournisseur de cloud, qu'est-ce que c'est ? »* URL : https://www.redhat.com/fr/topics/cloud-computing/what-are-cloud-providers/.

[42] Schaffer. « Multiple objective optimization with vector evaluated genetic algorithms ». In : *Proceedings of the first international conference on genetic algorithms and their applications, 1985*. Lawrence Erlbaum Associates. Inc., Publishers. 1985, p. 93-100. URL : https://www.researchgate.net/publication/216301392_Multiple_Objective_Optimization_with_Vector_Evaluated_Genetic_Algorithms.

[43] Sellami et al. « IMMUNE GENETIC ALGORITHM FOR SCHEDULING SERVICE WORKFLOWS WITH QOS CONSTRAINTS IN CLOUD COMPUTING ». Thèse de doct. 2013, p. 1-15. URL : https://www.researchgate.net/publication/260839376_Immune_genetic_algorithm_for_scheduling_service_workflows_with_QoS_constraints_in_cloud_computing.

[44] Srinivas et Deb. « Muiltiobjective optimization using nondominated sorting in genetic algorithms ». In : *Evolutionary computation* 2.3 (1994), p. 221-248. URL : https://web.njit.edu/~horacio/Math451H/download/SrinivasDeb_GA.pdf.

[45] Vallee, Vary-Sinamale et Belkacem. « DATA CENTERS et CLOUD COMPUTING ». Thèse de doct. UTEC Marne La Vallée, 2016, p. 6. URL : http://docplayer.fr/57913951-M2-m2i-gr-a-datacenters-cloud-computing-memoire-de-recherches-utec-marne-la-vallee-arthur-vary-sinamale-idir-tiret-matthias-ali-belkacem.html.

[46] Vmware. *VMware vCloud Suite*. URL : http://wmware.com.

[47] Linlin Wu, Saurabh Kumar Garg et Rajkumar Buyya. « SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments ». In : *Journal of Computer and System Sciences* (2012), p. 1280-1299. URL : 99.%20http://www.cloudbus.org/papers/AdmissionControlInClouds-JCSS.pdf.

[48] Xi-Xian. « An Anatomy of Joyent Smart Cloud Computing Technology ». In : *Journal of Xi'an Eurasia University* (2012), p. 1-19. URL : http://hoffmanmarcom.com/pdf-doc/Joyent-Smart-Architecture-for-Cloud-Computing.pdf.

[49] Yassa. « Allocation optimale multicontraintes des workflows aux ressources d'un environnement Cloud Computing ». Thèse de doct. Cergy-Pontoise, 2014. URL : https://hal.archives-ouvertes.fr/tel-01167131/.

[50] Zaigham. « Cloud computing: Characteristics and deployment approaches ». In : *2011 IEEE 11th International Conference on Computer and Information Technology*. 2011, p. 121-126. URL : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.462.9149&rep=rep1&type=pdf.

[51] Zitzler, Kalyanmoy et Lothar. « Comparison of multiobjective evolutionary algorithms: Empirical results ». In : *Evolutionary computation* (2000), p. 173-195. URL : https://sop.tik.ee.ethz.ch/publicationListFiles/zdt2000a.pdf.

[52] Zitzler et Thiele. « An evolutionary algorithm for multiobjective optimization: The strength pareto approach ». In : *TIK-report* (1998). URL : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.9077&rep=rep1&type=pdf.