

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

**Mémoire de fin d'études**  
**Pour l'obtention du diplôme de Master en Informatique**  
**Option : Réseaux et Systèmes Distribués (RSD)**

*Thème*

Shuffled Frog Leaping Algorithm pour l'ordonnancement  
des tâches dans le Cloud computing

Réalisé par :

❖ Mohammed El hadi

Présenté le 29 Juin 2022 devant le jury composé de :

|                          |              |                       |
|--------------------------|--------------|-----------------------|
| ❖ Mr Mohamed Lehsaini    | Président    | Université de Tlemcen |
| ❖ Mr Badr Benmammar      | Encadrant    | Université de Tlemcen |
| ❖ Mr Arslan Nedhir Malti | Co-encadrant | Université de Tlemcen |
| ❖ Mr Benmouna Youcef     | Examineur    | Université de Tlemcen |

Année universitaire : 2021-2022

## **Remerciement**

Je tiens à exprimer toute ma reconnaissance à mon encadrant de ce projet de fin d'études, Mr. Badr Benmammar et mon co-encadrant Mr. Malti Arslan. Je les remercie de m'avoir encadré, orienté, aidé et conseillé.

Je tiens à remercier les membres du jury : Mr Mohamed Lehsaini et Mr Benmouna Youcef, qui m'ont fait l'honneur de bien vouloir étudier et examiner mon travail.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je remercie mes très chers parents, qui ont toujours été là pour moi. Je remercie mes sœurs, et mes frères, pour leurs encouragements.

Enfin, je remercie mes amis qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

## Dédicace

*Je dédie ce modeste travail*

*À mes chers parents, source de bonheur et de réussite dans ma vie. Qu'Allah les bénisse.*

*À mes frères et sœurs pour leur patience et leur aide à la réalisation de ce mémoire.*

*À ma famille élargie et aux nombreux amis, qui m'ont tant soutenu et encouragé la réalisation de ce travail.*

*À tous mes professeurs pour leur soutien inconditionnel et leurs encouragements à poursuivre mes études universitaires.*

*À tous ceux qui croient en moi et prient pour mon succès.*

*Mohammed El hadi*



## Résumé

Le Cloud computing est la disponibilité à la demande de différentes ressources matérielles et logicielles du système informatique à travers l'internet. Dans ce contexte, l'ordonnancement des tâches est un élément très important pour le bon fonctionnement du système. Les fournisseurs et les utilisateurs des services Cloud ont des objectifs souvent contradictoires et l'ordonnanceur doit tenir compte de ce type de contrainte. Par conséquent, l'ordonnancement des tâches dans le Cloud computing devient un problème d'optimisation multi objectif. Dans ce travail, nous avons appliqué l'algorithme SFLA pour l'ordonnancement des tâches multi-objectif dans le Cloud computing. L'implémentation de l'algorithme a été réalisée par l'outil de simulation CloudSim et les résultats obtenus sont très satisfaisants.

**Mots Clés :** Cloud computing, ordonnancement des tâches, SFLA, CloudSim.

## ملخص

الحوسبة السحابية هي التوافر عند الطلب لموارد أجهزة وبرامج أنظمة الحاسوب المختلفة عبر الإنترنت. في هذا السياق، تعد جدولة المهام عنصرًا مهمًا للغاية من أجل حسن سير النظام. غالبًا ما يكون لمقدمي ومستخدمي الخدمات السحابية أهداف متناقضة ويجب أن يأخذ المجدول هذا النوع من القيد في الاعتبار. لذلك، تصبح جدولة المهام في الحوسبة السحابية مشكلة تحسين متعددة الأغراض. في هذا العمل، طبقنا خوارزمية SFLA لجدولة المهام متعددة الأهداف في الحوسبة السحابية. تم تنفيذ الخوارزمية بواسطة أداة المحاكاة CloudSim وكانت النتائج التي تم الحصول عليها مرضية للغاية.

**الكلمات المفتاحية :** الحوسبة السحابية، جدولة المهام، SFLA، CloudSim.

## Abstract

Cloud computing is the on-demand availability of various hardware and software resources of the computer system through the Internet. In this context, task scheduling is a very important element for the proper functioning of the system. Providers and users of Cloud services often have conflicting goals and the scheduler has to take this kind of constraint into account. Therefore, task scheduling in Cloud computing becomes a multi-objective optimization problem. In this work, we applied the SFLA algorithm for multi-objective task scheduling in Cloud computing. The implementation of the algorithm was done by the CloudSim simulation tool and the results obtained are very satisfactory.

**Key words:** Cloud computing, task scheduling, SFLA, CloudSim.

## Liste des matières

|   |            |
|---|------------|
| <b>Remerciement</b>   | <b>I</b>   |
| <b>Dédicace</b>   | <b>II</b>  |
| <b>Résumé</b>   | <b>III</b> |
| <b>Table des matières</b>                                     | <b>IV</b>  |
| <b>Liste des tables</b>                                       | <b>VI</b>  |
| <b>Liste des figures</b>                                      | <b>VII</b> |
| <b>Liste des abréviations</b>                                 | <b>IX</b>  |
| <b>Introduction générale</b>                                  | <b>1</b>   |
| <b>Chapitre 1 : Cloud computing</b>                           | <b>2</b>   |
| 1.1 Introduction  | 3          |
| 1.2 Définition  | 3          |
| 1.3 Modèles de déploiement                                    | 3          |
| 1.4 Types de service  | 4          |
| 1.5 Outils de développement                                   | 4          |
| 1.6 Sécurité dans le Cloud                                    | 5          |
| 1.7 Virtualisation  | 7          |
| 1.8 L'internet des objets                                     | 8          |
| 1.9 Hébergement des jeux en ligne multi-joueurs dans le Cloud | 9          |
| 1.10 Ordonnancement des tâches dans le Cloud computing        | 11         |
| 1.11 Conclusion   | 11         |
| <b>Chapitre 2 : Les métaheuristiques</b>                      | <b>12</b>  |
| 2.1 Introduction  | 13         |
| 2.2 Méthodes de résolution                                    | 13         |
| 2.3 Classification des métaheuristiques                       | 14         |
| 2.4 Recherche aléatoire                                       | 16         |
| 2.5 Recherche locale  | 17         |
| 2.6 Shuffled Frog Leaping Algorithm                           | 21         |
| 2.7 Conclusion  | 23         |

|  |           |
|--|-----------|
| <b>Chapitre 3 : Implémentation de l'algorithme et évaluation des résultats obtenus</b> | <b>24</b> |
| 3.1 Introduction   | 25        |
| 3.2 Outils de simulations  | 25        |
| 3.3 Critères d'évaluation  | 27        |
| 3.4 IHM développée   | 27        |
| 3.5 Résultats obtenus et étude comparative   | 31        |
| 3.6 Conclusion   | 38        |
| <b>Conclusion générale</b>   | <b>39</b> |
| <b>Références</b>  | <b>40</b> |

## Liste des Tables

|   |    |
|---|----|
| <b>Tableau 1.1:</b> Les différents types de service Cloud.....        | 4  |
| <b>Tableau 2.1:</b> Exemple de voisinages de la recherche locale..... | 17 |
| <b>Tableau 3.1 :</b> Les paramètres de simulation de CloudSim. ....   | 31 |
| <b>Tableau 3.2 :</b> Spécification des machines utilisées.....        | 31 |

## Liste des Figures

|  |    |
|--|----|
| <b>Figure 1.1:</b> Algorithmes existants dans la sécurité du Cloud. ....                       | 7  |
| <b>Figure 1.2:</b> Les types de virtualisation. ....   | 8  |
| <b>Figure 1.3a:</b> IoT M2M Hardware Solutions .....   | 9  |
| <b>Figure 1.3b:</b> IT Cloud vs CoT .....  | 9  |
| <b>Figure 1.4:</b> Model d'ordonnancement des tâches dans le Cloud computing.....              | 11 |
| <b>Figure 2.1:</b> Les méthodes d'optimisation classique.....                                  | 13 |
| <b>Figure 2.2:</b> Classification de métaheuristiques. ....                                    | 14 |
| <b>Figure 2.3 :</b> Recherche à base de solution unique vs recherche à base de population..... | 16 |
| <b>Figure 2.4 :</b> Le graphe de la recherche aléatoire.....                                   | 17 |
| <b>Figure 2.5 :</b> Stratégie de sélection de l'algorithme « Best Improvement » .....          | 18 |
| <b>Figure 2.6 :</b> Exemple de l'algorithme « Best Improvement » .....                         | 19 |
| <b>Figure 2.7 :</b> Stratégie de sélection de l'algorithme « First Improvement » .....         | 20 |
| <b>Figure 2.8 :</b> Exemple de l'algorithme « First Improvement » .....                        | 20 |
| <b>Figure 2.9 :</b> Stratégie de sélection pour améliorer les voisins .....                    | 20 |
| <b>Figure 2.10 :</b> Règle du saut de grenouille .....   | 21 |
| <b>Figure 2.11 :</b> Méthodologie de l'algorithme SFLA .....                                   | 22 |
| <b>Figure 3.1 :</b> Architecture de CloudSim.....  | 26 |
| <b>Figure 3.2 :</b> Interface principale.....  | 27 |
| <b>Figure 3.3 :</b> Caractéristique de l'algorithme SFLA .....                                 | 28 |
| <b>Figure 3.4 :</b> Création des machines physiques.....                                       | 28 |
| <b>Figure 3.5 :</b> Création des machines virtuelles.....                                      | 29 |
| <b>Figure 3.6 :</b> Création des Cloudlets .....   | 29 |
| <b>Figure 3.7 :</b> Type d'ordonnancement.....   | 30 |
| <b>Figure 3.8 :</b> Configurations des pondérations.....                                       | 30 |
| <b>Figure 3.9 :</b> Comparaison entre SFLA et RR en termes de makespan pour 40 VMS .....       | 32 |
| <b>Figure 3.10 :</b> Comparaison entre SFLA et RR en termes de makespan pour 60 VMS .....      | 32 |



|  |    |
|--|----|
| <b>Figure 3.11</b> : Comparaison entre SFLA et RR en termes de makespan pour 80 VMS .....  | 33 |
| <b>Figure 3.12</b> : Comparaison entre SFLA et RR en termes de makespan pour 100 VMS ..... | 33 |
| <b>Figure 3.13</b> : Comparaison de makespan pour 40 VMS .....                             | 34 |
| <b>Figure 3.14</b> : Comparaison de makespan pour 60 VMS .....                             | 34 |
| <b>Figure 3.15</b> : Comparaison de makespan pour 80 VMS .....                             | 35 |
| <b>Figure 3.16</b> : Comparaison de makespan pour 100 VMS .....                            | 35 |
| <b>Figure 3.17</b> : Comparaison de coût pour 40 VMS .....                                 | 36 |
| <b>Figure 3.18</b> : Comparaison de coût pour 60 VMS .....                                 | 36 |
| <b>Figure 3.19</b> : Comparaison de coût pour 80 VMS .....                                 | 37 |
| <b>Figure 3.20</b> : Comparaison de coût pour 100 VMS .....                                | 37 |

## Liste des abréviations

| Acronyme | Signification                              |
|----------|--|
| AES      | Advanced Encryption Standard               |
| AWS CDK  | Amazon Web Service Cloud Development Kit   |
| CoT      | Cloud of Things                            |
| CRC32    | Cyclic Redundancy Check 32                 |
| DES      | Data Encryption Standard                   |
| DSA      | Digital Signature Algorithm                |
| ECDSA    | Elliptic Curve Digital Signature Algorithm |
| HDD      | Hard Disk Drive                            |
| HTTP     | Hypertext Transfer Protocol                |
| IoT      | Internet of Things                         |
| M2M      | Machine to Machine                         |
| MD5      | Message-digest 5                           |
| NESW     | North East South West                      |
| OS       | Operating System                           |
| PGP      | Pretty Good Privacy                        |
| QoS      | Quality of Service                         |
| RC5      | Ronald Rivest 5                            |
| RR       | Round Robin                                |
| SFLA     | Shuffled Frog Leaping Algorithm            |
| SHA      | Secure Hash Algorithm                      |
| SSD      | Solid State Drive                          |
| SSL      | Secure Sockets Layer                       |
| VM       | Virtuel Machine                            |

## Introduction générale

Depuis la dernière décennie, les applications traditionnelles ont toujours été très compliquées et coûteuses en termes de configuration. En effet, vous auriez besoin systématiquement d'une équipe d'experts pour installer et configurer les différentes opérations informatiques. Après l'apparition du Cloud computing, il est devenu possible d'éliminer ces problèmes en payant ce dont vous avez besoin. Aujourd'hui, de nombreuses entreprises ont déplacé leurs applications vers le Cloud en raison des mises à jour logicielles automatiques, de la capacité illimitée des ressources, de la réduction des coûts ainsi que de son environnement sécurisé. Le Cloud computing est également conçu pour satisfaire les clients en leur offrant les meilleures performances dans leur domaine spécifique. La réalisation de cet objectif contribuera à donner des priorités appropriées aux différentes applications sensibles. Cependant, l'ordonnancement des tâches est un outil qui permet d'exécuter automatiquement des actions prédéfinies chaque fois qu'un certain ensemble de conditions est rencontré.

Dans ce contexte, il existe des algorithmes métaheuristiques que nous pouvons utiliser pour trouver une meilleure solution dans le domaine de l'ordonnancement des tâches. En effet, les métaheuristiques sont conçues pour générer une bonne solution à un problème d'optimisation en un temps de calcul raisonnable.

Notre contribution dans le cadre de ce projet de fin d'études consiste à appliquer une métaheuristique pour l'optimisation de plusieurs objectifs dans le Cloud computing. La problématique traitée concerne l'ordonnancement des tâches multi-objectif dans le Cloud computing par rapport à 2 métriques de qualité de services (QoS), qui sont le makespan et le coût. Le simulateur CloudSim a été utilisé dans le cadre de ce travail et les résultats obtenus sont très satisfaisants.

Le reste de ce manuscrit est structuré comme suit : le premier chapitre est dédié à la présentation du cadre général de notre travail qui est le Cloud computing. Dans le deuxième chapitre, nous avons présenté les métaheuristiques et leurs caractéristiques en se focalisant sur le SFLA. Dans le troisième et dernier chapitre, nous avons présenté notre contribution dans le cadre de ce travail qui consiste à l'adaptation et l'implémentation de l'algorithme SFLA pour l'ordonnancement des tâches multi-objectif dans le Cloud computing. L'outil de simulation CloudSim a été utilisé pour la réalisation des différentes simulations et pour la comparaison mono-objectif avec le Round Robin par rapport au makespan.

# Chapitre 1 : Cloud computing

## 1.1 Introduction

De nos jours, l'avancement de la technologie a rendu notre vie très facile. Les gens ont énormément profité de ça, car cela leur permet de maintenir leur infrastructure informatique et d'avoir un bon fonctionnement des activités dans leur entreprise. Grâce au Cloud computing, il est devenu maintenant un moyen facile d'accéder et de partager les informations avec un niveau de sécurité et de flexibilité supérieur.

L'objectif de ce chapitre est de présenter les concepts fondamentaux du Cloud computing : sa définition, ses caractéristiques, ses modèles de livraison et de déploiement ainsi que de détailler certains des enjeux de la sécurité dans le Cloud.

## 1.2 Définition

Le Cloud computing est un système informatique distribué qui offre la distribution de différents types de services et de ressources aux clients à travers l'internet. En d'autres termes, les fournisseurs de services Cloud proposent des services informatiques suivis par des stratégies de prix où les utilisateurs sont facturés en fonction de leur utilisation, comme le service « pay-as-you-go ». Ces services de réseau ont permis à de nombreuses entreprises de bénéficier d'une connectivité mondiale, car cela permet à leurs employés et clients d'accéder au point de terminaison le plus proche. Cependant, il existe une grande diversité parmi les services de Cloud dans lesquels ils peuvent être classés en quelques modèles et type de service [1].

## 1.3 Modèles de déploiement

Le modèle de déploiement identifie l'objectif et la nature de l'environnement du Cloud. Par conséquent, il est important que les clients spécifient le type d'architecture en fonction de la quantité de données qu'ils souhaitent stocker à des personnes ayant accès à l'infrastructure. Il existe principalement quatre types de modèles de déploiement de Cloud listé comme suit [2] :

- **Cloud public** : cette forme de Cloud est moins sécurisée car elle permet à chacun d'accéder facilement aux systèmes et aux services d'infrastructure. Cependant, c'est une excellente option pour l'hébergement Cloud où ces services sont fournis gratuitement.
- **Cloud privé** : il est aussi appelé Cloud interne à cause de son environnement sécurisé et protégé par des pare-feu qui appartiennent à la gouvernance du département informatique. De plus, un système existant peut être configuré pour accorder uniquement aux utilisateurs autorisés un accès direct à leurs données.
- **Cloud hybride** : il s'agit d'une utilisation mixte du Cloud public et du Cloud privé. Il peut être utilisé pour tirer avantage des services de Cloud public lorsque la capacité du Cloud privé est insuffisante.
- **Cloud communautaire** : il est hébergé par une communauté particulière pour partager des ressources et des services afin de répondre aux besoins des communautés, des industries ou des entreprises.

## 1.4 Types de service

Le Cloud computing se compose de trois principaux modèles de services qui sont : Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a service (SaaS). Le tableau ci-dessous montre la différence entre eux et ce qu'ils peuvent offrir à une entreprise en termes d'utilisation [3] :




| <i>Classe de service</i>   | <i>Contenu des services</i>   | <i>Caractéristiques</i>  | <i>Exemple</i>   |
|--|---|--|--|
| <br><b>IaaS</b>   | <ul style="list-style-type: none"> <li>✓ Réseaux sociaux</li> <li>✓ Office Suits</li> <li>✓ Traitement des fichiers</li> </ul>                        | <ul style="list-style-type: none"> <li>✓ Technologie de virtualisation de plateforme.</li> <li>✓ Très flexible et hautement évolutif.</li> <li>✓ Rentable.</li> <li>✓ Accès basé sur l'interface graphique et l'API.</li> <li>✓ Connectivité Internet.</li> </ul>  | <ul style="list-style-type: none"> <li>✓ AWS EC2</li> <li>✓ Rackspace</li> <li>✓ Digital Ocean</li> <li>✓ Linode</li> </ul>                    |
| <br><b>PaaS</b>   | <ul style="list-style-type: none"> <li>✓ Langages de programmation</li> <li>✓ Frameworks</li> <li>✓ Données structurées</li> </ul>                    | <ul style="list-style-type: none"> <li>✓ Construit sur la technologie de virtualisation.</li> <li>✓ Facile à exécuter sans connaissances approfondies en administration système.</li> <li>✓ Possibilité de choisir parmi différents niveaux de ressources en fonction de la taille de votre entreprise.</li> </ul> | <ul style="list-style-type: none"> <li>✓ Windows Azure</li> <li>✓ OpenShift</li> <li>✓ AWS Elastic Beanstalk</li> </ul>                        |
| <br><b>SaaS</b> | <ul style="list-style-type: none"> <li>✓ Serveurs</li> <li>✓ Backups</li> <li>✓ Data Storage</li> <li>✓ Load Balancer</li> <li>✓ Firewalls</li> </ul> | <ul style="list-style-type: none"> <li>✓ Hébergé sur un serveur distant par un tiers.</li> <li>✓ Scalable, avec différents niveaux pour les petites, moyennes et grandes entreprises.</li> <li>✓ Inclusif, offrant sécurité, conformité et maintenance dans le cadre du coût</li> </ul>                            | <ul style="list-style-type: none"> <li>✓ Google Apps</li> <li>✓ Dropbox</li> <li>✓ Zendesk</li> <li>✓ HubSpot</li> <li>✓ Salesforce</li> </ul> |

Tableau 1.1: Les différents types de service Cloud

## 1.5 Outils de développement

Le développement des applications dans le Cloud est un concept principal pour les développeurs afin de créer leurs applications sans avoir télécharger quoi que ce soit sur leur ordinateur. Dans ce cas, un certain nombre d'étapes peuvent être configurées pour créer, éditer, maintenir et déboguer des programmes sans écrire de nouveau code en production. Ainsi, il est important de déterminer le meilleur outil de développement basé sur le Cloud qui correspond à leurs besoins [4].

### **1.5.1 Windows Azure SDK**

Les Azure SDK sont des collections de bibliothèques de langages de programmation tels que JAVA, Python, .NET, iOS, etc. Ils vous aident à créer des applications qui interagissent avec les services Azure pour offrir de nombreux avantages tels que la flexibilité, la conformité, la sécurité et la prise en charge des analyses. Les Azure SDK peuvent également être gérés par des outils comme Azure CLI, Azure PowerShell, AzCopy et Azure Cloud Shell [4].

### **1.5.2 AWS Cloud Development KIT**

AWS Cloud Development Kit (AWS CDK) est une infrastructure de développement logiciel open source permettant de définir des ressources d'application Cloud à l'aide de langages de programmation familiers. Il offre des outils et des fonctionnalités existants pour créer des applications Cloud [4].

### **1.5.3 Eucalyptus**

Eucalyptus est un logiciel informatique open source payant permettant de fournir un ensemble combiné d'API pour créer une compatibilité dans un Cloud privé et hybride. Il est utilisé pour relier des programmes à des systèmes utiles. De plus, Eucalyptus permet aux utilisateurs de provisionner des ressources de calcul et de stockage à la demande dans les produits IaaS [4].

### **1.5.4 Apache CloudStack**

CloudStack est un logiciel de Le Cloud computing open source permettant de gérer un grand réseau de machines virtuelles afin de créer une plate-forme IaaS hautement disponible et évolutive. Il prend en charge les hyperviseurs les plus populaires comme VMware vSphere, KVM, XenServer, XenProject et Hyper-V ainsi que les conteneurs OVM et LXC. Par ailleurs, CloudStack peut être géré par des outils tels que l'interface Web, les outils de ligne de commande et des API basée sur des requêtes [4].

## **1.6 Sécurité dans le Cloud**

La sécurité du Cloud est un ensemble de protocoles conçus pour protéger les différents types de systèmes. Cela inclut les données, les applications et l'infrastructure. Les fournisseurs de Cloud sont responsables de l'environnement du client car ils doivent être conscients de toute action nuisible et de toute menace causée par des criminels éventuels. En général, ils ont une connaissance approfondie des opérations de sécurité réseau, ce qui leur permet de maintenir un excellent niveau de sécurité, à partir du système d'exploitation jusqu'à la couche matérielle. En outre, ils mettent en œuvre des procédures comme les politiques de l'entreprise pour empêcher leurs propres employés de consulter les données des clients. Cependant, les clients doivent également rester sous leur propre responsabilité contre les menaces en ligne [5].

La sécurité est un domaine qui est en plein expansion dans le Cloud computing. Par conséquent, la plupart des entreprises et des agences de sécurité embauchent des experts pour

améliorer leurs techniques de sécurité. Cela inclut l'évaluation de nouveaux algorithmes après avoir analysé les performances et les vulnérabilités des anciennes versions [5].

### 1.6.1 Confidentialité des données et risque de sécurité

Les menaces à la sécurité des données et les vulnérabilités du système sont deux facteurs importants dont chaque entreprise doit être consciente. Les pirates et les cybercriminels peuvent utiliser plusieurs méthodes pour contourner la sécurité et obtenir un accès non autorisé afin de voler des informations privées et de perturber les systèmes informatiques. De plus, ils peuvent subir d'énormes pertes et nuire à la réputation des entreprises qui n'ont pas réussi à protéger leurs données.

Selon des statistiques élaborées dans le domaine du Cloud computing, presque toutes les entreprises s'appuient entièrement sur le Cloud puisque 81 % d'entre eux ont déclaré avoir une stratégie multi-Cloud déjà définie ou en cours d'élaboration, tandis qu'à la fin de 2021, 67 % de toutes les infrastructures d'entreprise étaient basées sur le Cloud. Par conséquent, il est toujours nécessaire de prendre en compte les catastrophes qui peuvent survenir au fil du temps, car cela pourrait se transformer en pertes financières, en réclamations de clients et en faillite d'entreprise si ces services devenaient indisponibles. Pour illustrer, certaines entreprises célèbres n'ont pas pris des mesures préventives et ont été exposées à des cyberattaques comme le site d'achat chinois Taobao d'Alibaba en novembre 2019, la violation de données sur Facebook en 2019 et LinkedIn en 2021, ainsi que l'incident bien connu qui s'est produit chez la plateforme SaaS de Google dans lequel leurs systèmes ont été attaqués par la Chine en janvier 2010 [6].

### 1.6.2 Chiffrement dans le Cloud

Le chiffrement utilise des algorithmes avancés pour coder les données du client afin qu'elles deviennent illisibles. Un exemple de ceci est de stocker un texte chiffré dans le Cloud après l'avoir transformé à partir de son texte en clair d'origine. Il vise à empêcher les autres locataires du Cloud de tout type de vol ou de corruption de données. Pourtant, il existe deux formes de chiffrement utilisées dans le Cloud décrites ci-dessous [7] :

- **Chiffrement des données en transit** : il chiffre les données à travers les protocoles HTTP et SSL. Le SSL encode tout le trafic au sein du canal pour empêcher les utilisateurs non autorisés de lire et de modifier les informations transférées au cours de la session. De plus, les utilisateurs génèrent des clés numériques pour verrouiller et déverrouiller les données cryptées.
- **Chiffrement des données en repos** : ce type de chiffrement utilise également des clés pour le codage ou le décodage comme le chiffrement des données en transit. Et même si les données sont perdues, volées ou partagées par erreur, le contenu est pratiquement inutile sans la clé de cryptage.

### 1.6.3 Algorithmes cryptographiques

Il existe plusieurs algorithmes cryptographiques utilisés pour chiffrer les données dans le Cloud et ils sont classés en trois types listés comme suit [8] (voir la figure 1.1) :



- **Cryptographie à clé symétrique** : c'est un système où l'expéditeur et le destinataire ont la même clé pour chiffrer et déchiffrer les messages.
- **Cryptographie à clé asymétrique** : il s'agit d'un système dans lequel deux clés différentes sont utilisées pour chiffrer et déchiffrer les données, par exemple une clé publique est utilisée pour chiffrer les données pour tout le monde, et une clé privée est utilisée pour déchiffrer les données pour le destinataire seul.
- **Hushing key** : il utilise des valeurs d'entrée aléatoires et produit une valeur de sortie fixe qui peut être utilisée pour identifier l'utilisateur pour stocker ces données privées.

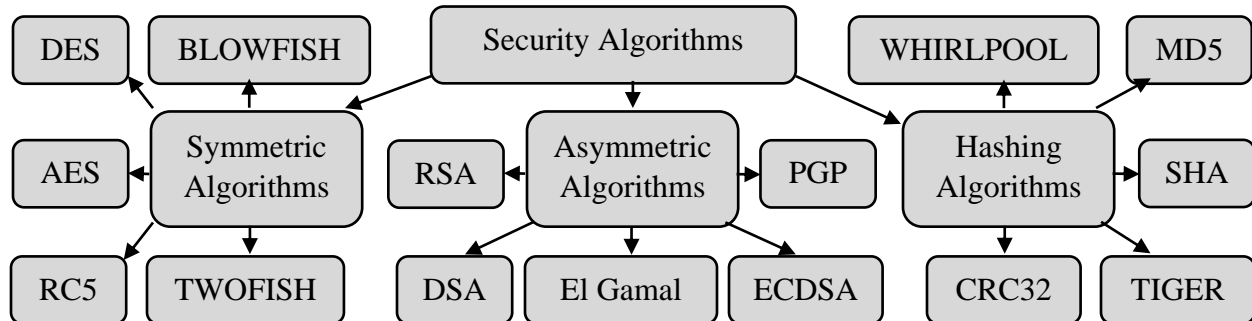


Figure 1.1: Algorithmes existants dans la sécurité du Cloud

## 1.7 Virtualisation

La virtualisation est la création d'un environnement informatique virtuel à partir d'un environnement physique. Cela permet la partition d'un seul ordinateur physique en plusieurs machines virtuelles. Chaque machine virtuelle peut alors exécuter plusieurs systèmes d'exploitation ou applications simultanément sur la même machine. Par conséquent, cela permettra d'économiser du coût de diverses ressources informatiques ainsi que par une augmentation de la productivité, des performances et de la réactivité [9].

### 1.7.1 Types de virtualisation

Il existe 6 types de virtualisation dans le Cloud (voir la figure 1.2) :

- **Virtualisation des applications** : il s'agit d'un processus dans lequel les applications sont virtualisées et transmises d'un serveur à l'appareil de l'utilisateur final, tels que les ordinateurs portables, les smartphones et les tablettes. De plus, il est particulièrement répandu pour les entreprises qui exigent l'utilisation de leurs applications lors de leurs déplacements à condition qu'une connexion Internet soit disponible [10].
- **Virtualisation de stockage** : la technologie repose sur un logiciel permettant d'identifier la capacité de stockage disponible à partir de périphériques physiques, puis d'agréger cette capacité en un pool de stockage pouvant être utilisé dans un environnement virtuel par des machines virtuelles (VM) [10].
- **Virtualisation des postes de travail** : un poste de travail virtualisé ou bureau virtuel peut être hébergé soit directement sur l'ordinateur du client soit sur un serveur dans le centre de données [11].

- **Virtualisation des systèmes d'exploitation :** la virtualisation des systèmes d'exploitation (OS) est une technologie qui consiste à adapter un système d'exploitation standard afin qu'il puisse exécuter différentes applications gérées par plusieurs utilisateurs sur un seul ordinateur [11].
- **Virtualisation des serveurs :** elle consiste à partitionner un serveur physique en un certain nombre de petits serveurs virtuels à l'aide d'un logiciel de virtualisation (hyperviseur). Les différentes partitions sont isolées les unes des autres [11].
- **Virtualisation de réseau :** il s'agit d'une reproduction logicielle d'un réseau physique. Elle consiste à combiner les ressources disponibles dans un réseau en divisant la bande passante disponible en différents canaux, chacun étant séparé et distingué [11].

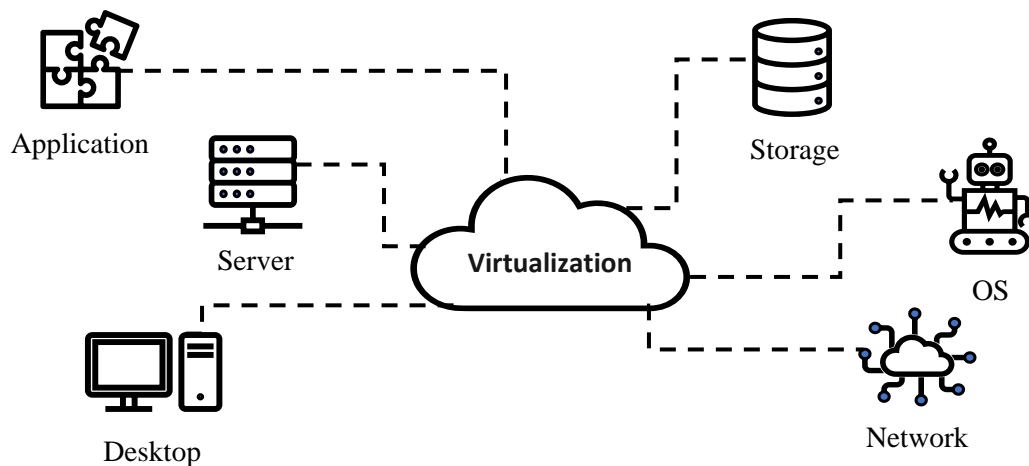


Figure 1.2: Les types de virtualisation.

## 1.8 L'internet des objets

L'internet des objets est un système d'appareils informatiques utilisé pour connecter des personnes, des données ou des objets à travers l'internet. C'est une combinaison de système embarqué, de capteurs et d'applications de contrôle pour améliorer la qualité de notre vie et résoudre les nombreux problèmes qu'une personne peut rencontrer (voir la figure 1.3a). De plus, les solutions d'IoT peuvent offrir une large gamme de services et de produits aux entreprises afin d'atteindre une activité potentielle dans leurs domaines respectifs. Par exemple, l'entreprise Samsara a développé un appareil qui peut améliorer les anciens véhicules juste en la plaçant sur le tableau de bord. Cependant, une fois l'installation terminée, les clients peuvent obtenir des analyses sur les données collectées à partir des plates-formes de Cloud, comme les alertes, les besoins de maintenance, l'emplacement et la consommation d'essence, etc., tout en les combinant dans une application pour avoir une vue complète, et plus détaillée [12].

Cloud of Things (CoT) fait référence à l'intégration de l'Internet des objets avec le Cloud computing. Il s'agit d'un nouveau paradigme technologique créé pour gérer le nombre élevé d'appareils qui manquent de service et de capacité de stockage. Par conséquent, CoT a également la même structure que le Cloud computing dans la location de ressources et d'espace de stockage avec la considération de confidentialité et de sécurité (voir la figure 1.3b).

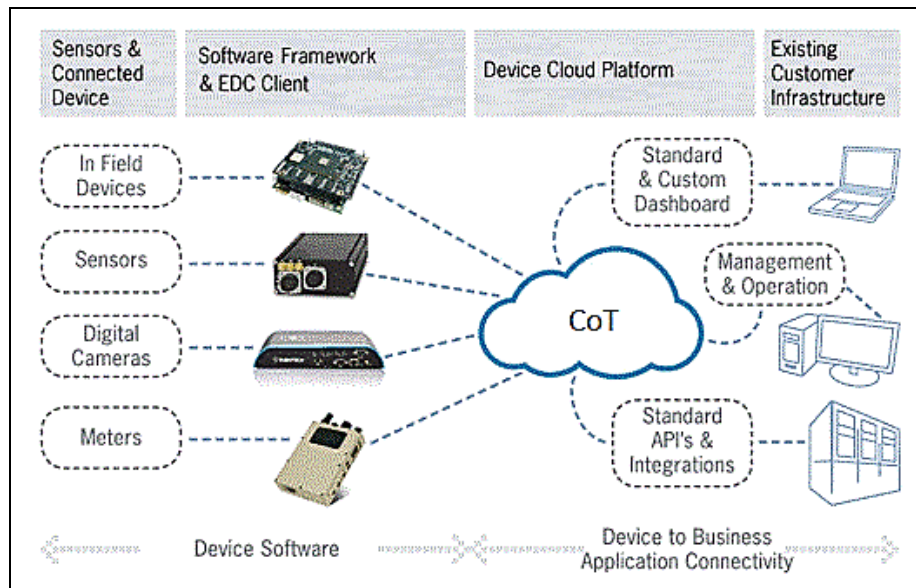


Figure 1.3a: IoT M2M Hardware Solutions [13]

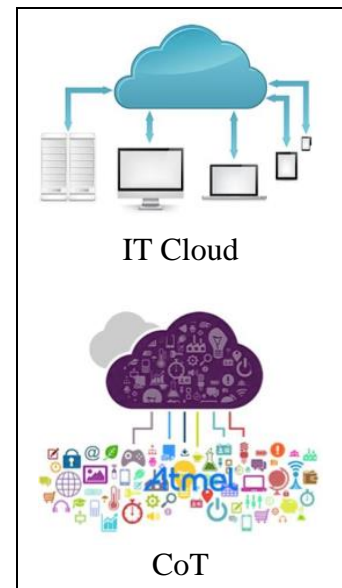


Figure 1.3b: IT Cloud vs CoT

## 1.9 Hébergement des jeux en ligne multi-joueurs dans le Cloud

Les jeux en ligne multi-joueurs existent depuis des décennies comme la plus grande application distribuée en raison de ses nombreux joueurs à travers le monde. Ces jeux en ligne sont hébergés sur des serveurs pour stocker les données collectées et calculer les différentes opérations soumises par les actions des joueurs. En plus, le temps de réponse doit toujours être pris en compte pour une expérience de jeu fluide, surtout si vous êtes dans un jeu interactif. Cela inclut la mise en place de dizaines de milliers de ressources de calcul afin de fournir une meilleure qualité de service (QoS) aux joueurs [1]. Par exemple, le jeu World of Warcraft (WoW) est hébergé dans 10 centres de données répartis dans le monde, avec une infrastructure de 13 250 lames de serveur, 75 000 cœurs de processeur et 112,5 téraoctets de RAM de lame, y compris des installations au Texas, en Californie, en France et en Allemagne., la Suède, la Corée du Sud et la Chine, etc...

Les jeux en ligne multi-joueurs sont implémentés avec divers langages de programmation comme C++, Java, Lua, Python, etc., ainsi que de nombreuses dépendances de bibliothèques. Cependant, selon le type de jeu, les développeurs de jeux professionnels peuvent parfois utiliser plus d'un langage à la fois pour maintenir des fonctionnalités avec des niveaux d'abstraction plus élevés qui nécessitent des efforts à long terme et beaucoup moins de temps.

### 1.9.1 Opérateur de jeux

La plate-forme d'hébergement se compose de centres de données et de fournisseurs de Cloud dispersés dans le monde entier. Un service de prédiction de charge supervise la projection de la distribution future des entités dans le monde du jeu qui s'avèrent avoir le plus grand impact sur la charge du serveur. Sur la base de la charge projetée, un service d'allocation de ressources alloue des serveurs locaux à la session de jeu. Le modèle d'hébergement tient également compte

de la taille et de la durée minimale d'allocation des ressources. Le volume de ressources est défini comme le nombre minimum de ressources pouvant être allouées pour une requête [1].

L'opérateur de jeu soumet des demandes de ressources aux hébergeurs en spécifiant le type, le nombre et la durée pour lesquels les ressources sont souhaitées. Une fois les ressources disponibles sélectionnées, elles sont allouées aux opérateurs du jeu. Les ressources allouées sont réservées à l'exécution des serveurs pendant toute la durée de la demande de l'opérateur du jeu [1].

### 1.9.2 Contexte de la virtualisation

La virtualisation est la solution pour porter les services logiciels des jeux en lignes sur différentes plateformes. Il fournit une plate-forme informatique uniforme pour configurer l'installation et le déploiement du jeu. Cependant, les jeux en ligne multi-joueurs peuvent être exécutés sur une machine virtuelle lourde ou sur une machine virtuelle légère. Une machine virtuelle lourde exécute le système d'exploitation complet et la partie middleware, par opposition à une machine virtuelle légère qui n'installe qu'une couche de contrôle et de communication minimale. Le temps total nécessaire pour lancer une machine virtuelle lourde est exprimé par la formule :

$$T = t_c + t_x + t_s + t_r \quad (1.1)$$

D'où  $t_c$  est le temps pour la préparation de la machine virtuelle,  $t_x$  est le temps pour le transfert de la machine virtuelle,  $t_s$  est le temps pour le démarrage de la machine virtuelle et  $t_r$  est le temps pour la suppression de la machine virtuelle [1].

L'évaluation de la qualité de l'expérience de jeu peut être configurée par une mesure de sous-allocation de ressources qui caractérise le pourcentage de ressources qui n'ont pas été allouées par rapport à la quantité nécessaire à l'exécution transparente du jeu en ligne multi-joueurs [1].

L'évaluation de l'efficacité de l'allocation des ressources peut être configurée par une mesure de surallocation des ressources qui caractérise le pourcentage d'une ressource (CPU, mémoire, réseau) allouée à partir de la quantité utilisée pour l'exécution transparente d'une session multi-joueurs [1].

Les deux mesures d'une simulation sont exprimées par la formule :

$$U(t) = O(t) = \frac{(A * L) * t}{T * L_{max}} * 100 \quad (1.2)$$

D'où  $t$  est la durée de l'événement de sous-allocation,  $A$  est la quantité de ressources allouées,  $L$  est la quantité de ressources nécessaires (mesurée à partir des traces), et  $L_{max}$  est la charge maximale déterminée par la conception du jeu [1].

La charge relative dans une certaine configuration de machine peut être calculée avec le pourcentage de la charge RuneScape théorique maximale qui pourrait être hébergée dans la

configuration donnée. Par exemple, la charge relative d'une session de jeu composée de  $N$  zones hébergées sur  $M$  machines, où une machine peut héberger exactement une zone, est :  $(N/M)*100$  [1].

### 1.10 Ordonnancement des tâches dans le Cloud computing

Le concept d'ordonnancement est défini par l'allocation de ressources appropriées aux différentes tâches. L'ordonnancement des tâches dans le Cloud computing prend en compte certaines contraintes telles que le budget, la qualité de service (QoS) et le débit du système pour réduire le coût et le temps de traitement ainsi que d'économiser l'énergie et d'augmenter l'utilisation des ressources [14].

Les utilisateurs du Cloud ont des applications avec une charge fluctuante et, par conséquent, un bon ordonnancement des ressources est indispensable pour répondre aux exigences du client et satisfaire une QoS efficace. Par conséquent, les utilisateurs louent et paient les ressources afin d'exécuter leurs applications, et une fois qu'ils ont terminé, les ressources sont restituées au fournisseur de services (voir la figure 1.4).

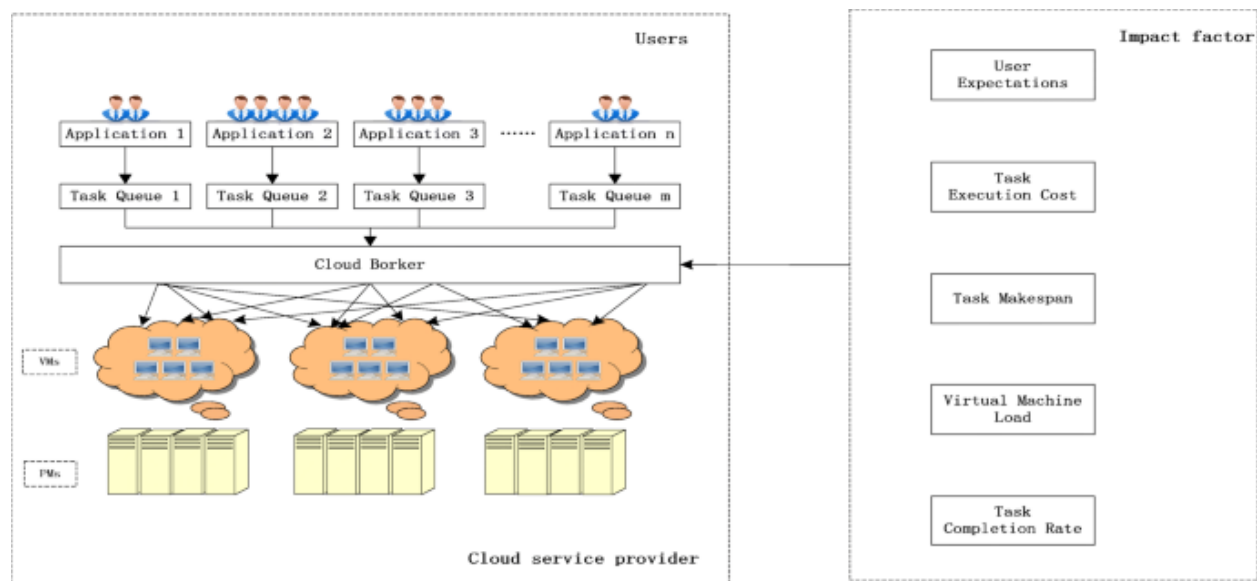


Figure 1.4 : Modèle d'ordonnancement des tâches dans le Cloud computing [15]

### 1.11 Conclusion

Dans ce chapitre, nous avons abordé le Cloud computing, en ce qui concerne les opportunités, les défis et les activités dans ses différents domaines. Nous avons également donné un aperçu de la relation entre les objets d'internet et le Cloud computing, ainsi que détaillé certains problèmes de sécurité et les précautions à prendre pour un environnement sûr.

Dans le chapitre suivant, nous consacrerons à la présentation des métaheuristiques.

## Chapitre 2 : Les métaheuristiques

## 2.1 Introduction

L'optimisation de forme devient un outil essentiel pour presque tous les domaines. Les développements récents dans l'optimisation géométrique et topologique des formes ont eu un impact considérable dans la science, l'ingénierie, l'économie et les industries, notamment dans l'automobile et l'aéronautique. Cependant, il existe un problème objectif d'optimisation de forme et de fiabilité des structures. Ces problèmes peuvent être résolus avec l'utilisation des métaheuristiques évolutionnaires, puisqu'elles sont efficaces et fiables [16].

L'objectif de ce chapitre est de présenter les concepts de métaheuristique : sa définition, son classement par rapport aux méthodes d'optimisation, sa classification, ainsi que détaillé certains d'algorithmes utilisés de métaheuristique.

## 2.2 Méthodes de résolution

Les problèmes peuvent être résolus soit par des méthodes exactes ou par des méthodes approchées tout dépend de la complexité (voir la figure 2.1).

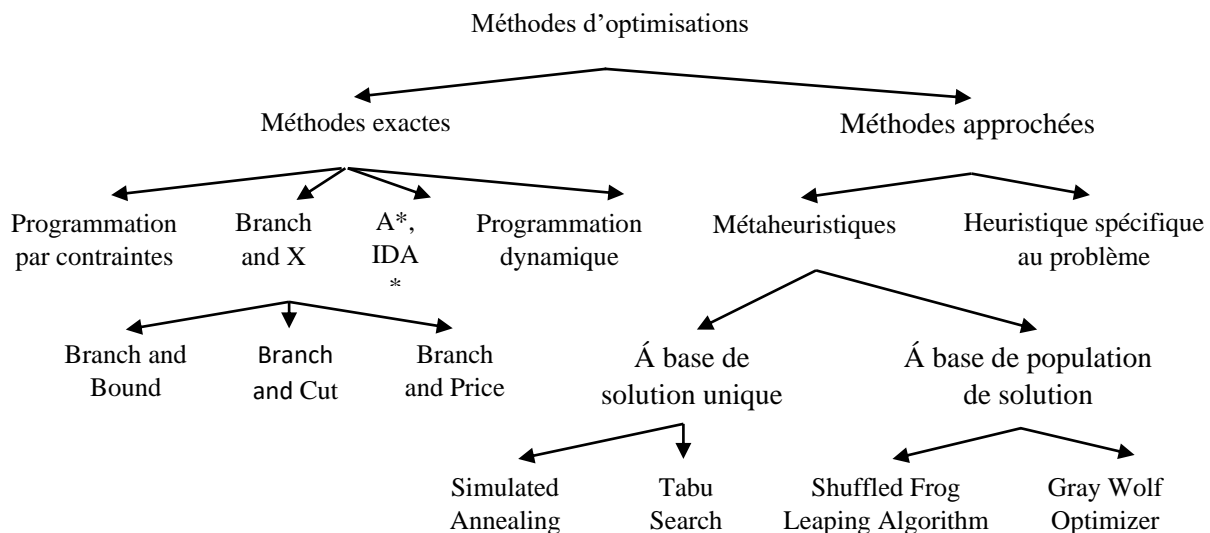


Figure 2.1 : Les méthodes d'optimisation classique [17][18]

### 2.2.1 Méthodes exactes

Les méthodes exactes fournissent la solution optimale à coup sûr dans lequel nous garantissons que si vous utilisez cette méthode, vous obtiendrez la meilleure solution possible. Par ailleurs, on ne peut pas les utiliser très souvent car il faudrait parfois trop de temps pour trouver la solution optimale. Donc fondamentalement, il ne sert à rien d'appliquer des méthodes conversationnelles à ces problèmes difficiles car cela prendrait trop de temps [17].

### 2.2.2 Méthodes approchées

Les méthodes approchées garantissent des solutions de haute qualité dans un délai raisonnable, mais ils ne garantissent pas l'optimalité des solutions obtenues. Par ailleurs, ces méthodes peuvent être divisées en deux classes décrites comme suit :

- **Les heuristiques** : sont des règles empiriques simples basées sur l'expérience (résultats déjà obtenus) et sur l'analogie. Ils génèrent des solutions de haute qualité dans un délai raisonnable pour une utilisation pratique, mais il n'y a aucune garantie de trouver une solution optimale [19].
- **Les métaheuristiques** : sont généralement des algorithmes apparus dans les années 80 dans le but d'apporter une bonne solution à des problèmes d'optimisation difficiles (typiquement les NP-problèmes). Ils peuvent être représentés comme un ensemble de solutions pour des problèmes qui sont trop grands pour être résolu. Cependant, les métaheuristiques ne garantissent pas une solution optimale globale car elles peuvent souvent trouver de bonnes solutions avec moins d'effort de calcul que d'autres approches telles que les algorithmes d'optimisation ou les heuristiques simples [16].

### 2.3 Classification des métaheuristiques

Il y a de nombreux critères de classification qui peuvent être utilisés pour les algorithmes de métaheuristiques. Ces algorithmes peuvent être classés comme suit (voir la figure 2.2) :

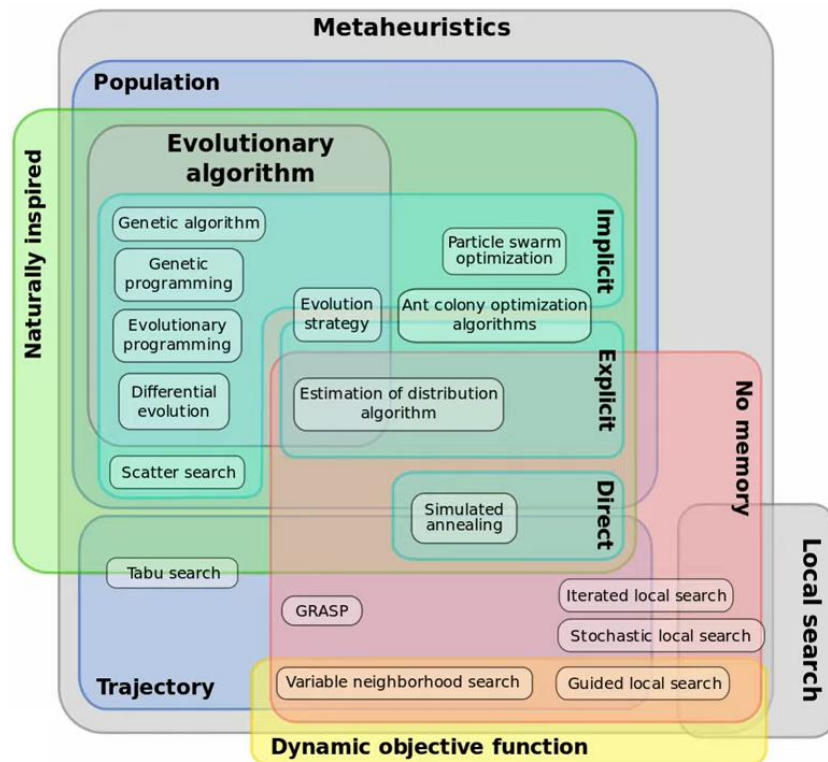


Figure 2.2 : Classification de métaheuristiques [20]



### 2.3.1 Déterministe vs stochastique

De nombreux algorithmes sélectionnent aléatoirement une condition initiale, et si on applique cette sélection, on trouve que presque toutes les métaheuristiques sont de type stochastique.

Cependant, il y a des métaheuristiques qui donneront toujours la même solution si on fixe une condition initiale, et elles seront classées de type déterministe.

Il y en a d'autres, que même lorsqu'on définit une condition initiale, on obtient des résultats potentiellement différents car ils contiennent des composants aléatoires.

Donc, ces métaheuristiques stochastiques plutôt que de donner une solution unique, elles donneront toute une distribution de solutions et c'est une chose que nous devons prendre en compte lors de l'évaluation des performances de ces algorithmes [17].

### 2.3.2 Inspiré de la nature vs non inspiré de la nature

Ce type de métaheuristique s'inspire de la nature et des espèces animales capables de résoudre des problèmes vraiment difficiles grâce à cette évolution. Certains d'entre eux sont des algorithmes évolutifs et des systèmes immunitaires artificiels qui s'inspirent principalement de la biologie. Il y a aussi tout un domaine de l'optimisation appelé optimisation des essaims de particules (particle swarm optimization) et ils sont basés sur des colonies de fourmis ou d'abeilles, et des recuits simulés inspirés avec des processus physiques [17].

### 2.3.3 Utilisation de la mémoire vs méthode sans mémoire

On peut aussi classer les métaheuristiques selon qu'elles utilisent ou non des informations du passé. Cependant, une utilisation de mémoire doit être nécessaire lors de la recherche comme la recherche locale, GRASP, et le recuit simulé. De plus, il existe d'autres algorithmes qui utilisent une mémoire qui contient certaines informations extraites en ligne lors de la recherche comme les mémoires à court terme et à long terme dans la recherche tabou [17].

### 2.3.4 Itératif vs gourmand

Les algorithmes itératifs sont ceux qui commencent par une solution complète ou plusieurs solutions complètes, puis ils transforment et changent ces solutions au fil du temps. Bien que, les algorithmes gourmands partent d'une solution vide puis ils construisent cette solution étape par étape jusqu'à ce qu'ils arrivent à la fin de l'algorithme pour atteindre une solution complète [17].

### 2.3.5 Recherche basée sur la population vs recherche basée sur une solution unique

Les algorithmes basés sur une solution unique (également appelés algorithmes basés sur la trajectoire) sont des algorithmes qui fonctionnent avec une seule solution. Alors que les algorithmes basés sur la population fonctionnent avec une distribution de solutions en même temps. Par exemple, comme on le voit sur la figure 2.3, un randonneur marche dans les montagnes afin de trouver le sommet, tandis que la recherche basée sur la population montre

qu'il y a tout un groupe de randonneurs essayant de trouver le sommet et, par conséquent, ils peuvent partager leurs résultats et échanger les informations entre eux, afin que chacun puisse trouver la région la plus haute et la plus parfaite [17].



Figure 2.3 : Recherche à base de solution unique vs recherche à base de population

## 2.4 Recherche aléatoire

La recherche aléatoire est un algorithme qui est plus basique et simple que nous pouvons utiliser pour trouver la solution optimale dans un certain espace de solution. Cette recherche peut être appliquée dans un cas de flux d'exploration qui consiste à échantillonner un certain nombre de solutions et à sélectionner la meilleure dans cet échantillon.

**Algorithm 1:** Random search [21]

```

1: Best ← some initial random candidate solution
2: repeat
3:   S ← a random candidate solution
4:   if Quality(S) > Quality(Best) then
5:     Best ← S
6: until Best is the ideal solution, or we have run out of time
7: return Best

```

**Quelle est la probabilité de trouver la solution optimale (parmi m solutions possibles) en n itérations ?**

- La probabilité d'atteindre la cible dans une itération donnée est  $1/m$ .
- La probabilité d'atteindre la cible dans n itérations est égale à :  
 = la probabilité d'atteindre la cible dans au moins 1 d'entre eux  
 =  $1 - \text{probabilité de ne pas atteindre la cible dans aucune des n itérations}$   
 =  $1 - (\text{probabilité de ne pas atteindre la cible en une itération})^n$   
 =  $1 - (1-1/m)^n = P(\text{succès en n itérations})$ .

On peut représenter la fonction de cette probabilité par un graphe (voir la figure 2.4), d'où l'axe vertical représente la probabilité de succès « m » et l'axe horizontal représente le nombre d'itérations « n ». Cependant, la probabilité de succès ne serait que d'environ 63%  $(1-1/e)$  pour n'importe quel nombre d'itérations car il y a des solutions qui se répètent à chaque fois [22].

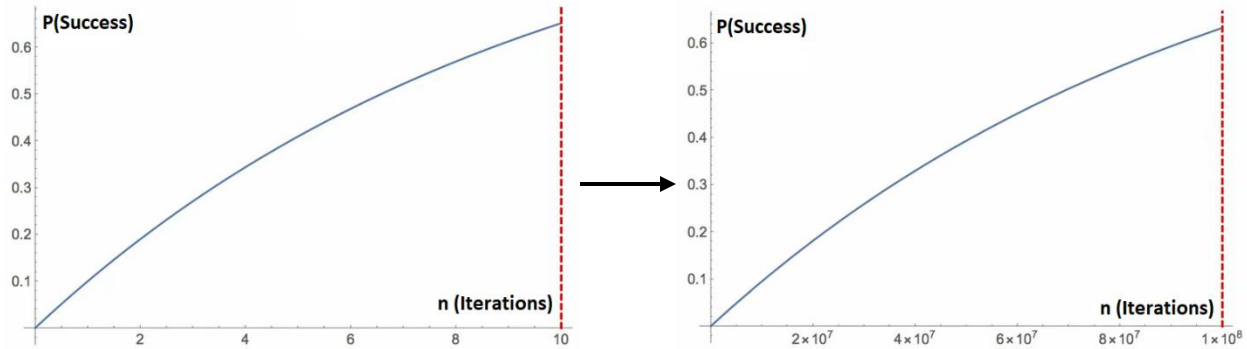


Figure 2.4 : Le graphe de la recherche aléatoire [22]

## 2.5 Recherche locale

La recherche locale est une métaheuristique basée sur une solution unique qui est liée à l'intensification et à l'exploration.

**Algorithm 2:** High-level model of S-metaheuristics [17]

- 1: **Input:** Initial solution  $s_0$ .
- 2:  $t = 0$ ;
- 3: **Repeat**
- 4:     Generate( $C(s_t)$ ); //Generate candidate solutions (Partial or complete neighborhood) from  $s_t$
- 5:      $s_{t+1} = \text{Select}(C(s_t))$ ; //Select solution from  $C(s)$  to replace the current solution  $s_t$
- 6:      $t = t + 1$ ;
- 7: **Until** Stopping criteria satisfied
- 8: **Output:** Best solution found

On peut créer un voisinage local pour effectuer une recherche uniquement dans celui-ci, puis on va choisir l'un des voisins selon certains critères, après on va se déplacer vers ce voisin qu'on a choisi et on recherche le voisinage de ce voisin, à la fin on répète ce processus itérativement jusqu'on trouve la meilleure solution. De plus, comme il est montré dans le tableau ci-dessous, les voisins doivent avoir la même qualité de voisinage pour effectuer une bonne recherche.

|  |   |  |
|--|---|--|
|  |   |  |
| Le cercle représente le voisinage de $s$ dans un problème continu à deux dimensions. | Les voisins d'une solution (ex. (0.1.0)) sont le nœud adjacent dans le graphe | Les nœuds de la forme (ex. (2.3.1)) représentent la solution du problème |

Tableau 2.1 : Exemple de voisinages de la recherche locale [17]

La propriété principale qui doit caractériser un voisinage est la localité. Par conséquent, on peut dire que la cartographie des voisinages a une localité forte si les voisinages ont une qualité similaire. Dans ce cas, la recherche locale effectuera une recherche significative dans la résolution du problème.

### 2.5.1 Famille d'algorithmes de la recherche locale

Il y a toute une famille dans la recherche locale qui peut être divisée en 3 parties qui sont décrites comme suit :

#### 2.5.1.1 Meilleure amélioration

Dans cette stratégie, le meilleur voisin est sélectionné. L'exploration du voisinage est exhaustive, c'est-à-dire que tous les mouvements possibles sont tentés pour une solution afin de sélectionner la meilleure solution voisine. Cet algorithme est donc déterministe. Cependant, ce temps d'exploration peut être chronophage pour les grands voisinages [17].

**Algorithm 3:** Best improvement [23]

- 1: Create initial solution  $s$ ;
- 2: **while** local optimum not reached **do**
- 3:     Determine complete neighborhood  $\mathcal{N}$  of current solution  $s$ ;
- 4:     **if**  $\mathcal{N}$  contains at least one improving solution **then**
- 5:         Choose best solution  $s'$  from  $\mathcal{N}$ ;
- 6:         Switch over to solution  $s'$  (current solution  $s$  is replaced by  $s'$ )
- 7:     **end**
- 8: **end**

L'idée est de partir d'une certaine solution et de chercher tout le voisinage de cette solution, puis on sélectionne le meilleur voisin parmi tous les voisins, et on répète ce processus jusqu'à la fin. Cet algorithme est donc déterministe puisque la condition de la solution initiale est fixée dès le départ (voir la figure 2.5).

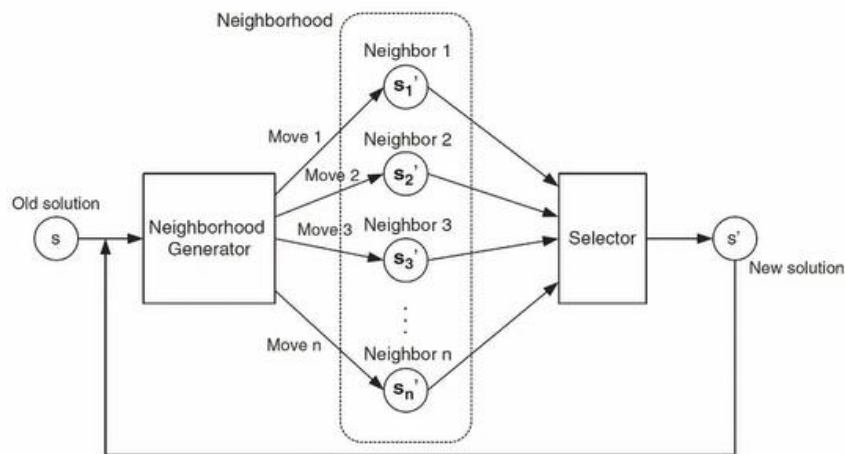


Figure 2.5: Stratégie de sélection de l'algorithme « Best improvement » [23]

La figure 2.6 montre un exemple de cet algorithme. On commence par choisir une solution optimale qui est la valeur « 3 » et notre but est de minimiser cette solution. Donc, ce que nous allons vraiment faire, c'est rechercher tous les voisinages qui sont liés à cette solution. donc parmi les quatre voisins on sélectionne celui qui a la valeur minimale qui est la valeur « 1 ». on répète cette étape même si la nouvelle valeur est égale à la valeur précédente, mais dans ce cas on finira ici car il n'y a pas de voisin qui soit mieux que la solution actuelle.

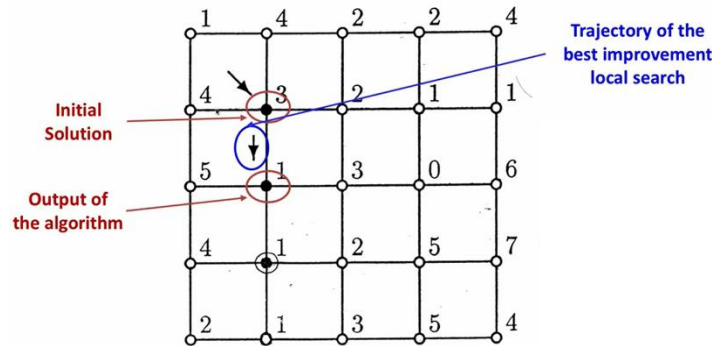


Figure 2.6: Exemple de l'algorithme (Best improvement) [23]

### 2.5.1.2 Première amélioration

Cette stratégie consiste à choisir le premier voisin améliorant qui est meilleur que la solution courante. Ensuite, un voisin en amélioration est immédiatement sélectionné pour remplacer la solution actuelle. Cette stratégie implique une évaluation partielle du voisinage. Dans une exploration cyclique, le voisinage est évalué de manière déterministe suivant un ordre donné de génération des voisins. Dans le pire des cas (c'est-à-dire lorsqu'aucune amélioration n'est trouvée), une évaluation complète du voisinage est effectuée [17].

**Algorithm 4:** First improvement [23]

- 1: Create initial solution  $s$ ;
- 2: **while** local optimum not reached **do**
- 3:     **repeat**
- 4:         Create new neighbor  $s'$  by applying a move
- 5:     **until**  $s'$  is better than  $s$  or no more moves available;
- 6:     **if**  $s'$  is better than  $s$  **then**
- 7:         Switch over to solution  $s'$  (current solution  $s$  is replaced by  $s'$ )
- 8:     **end**
- 9: **end**

Cet algorithme est également déterministe. Il consiste à chercher le voisinage de la solution initiale, et dès qu'on trouve un voisin qui est meilleur, on s'y dirige vers lui en respectant le même ordre.

Le pire des cas de cet algorithme est d'évaluer toutes ces solutions mais en général on n'évalue pas tous les voisins (voir l'exemple de la figure 2.7).

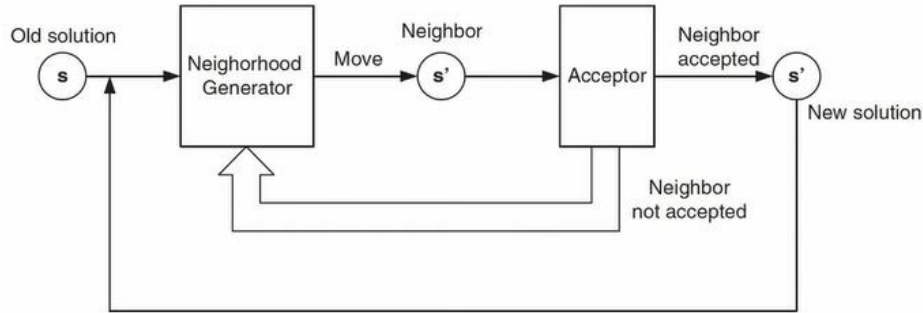


Figure 2.7: Stratégie de sélection de l'algorithme (First improvement) [23]

Pour illustrer, on prend le même exemple que précédemment et on suppose qu'on va respecter le même ordre de mouvement des voisins. De plus, l'ordre est défini au début par la condition "NESW" qu'il est essentiellement dans le sens des aiguilles d'une montre.

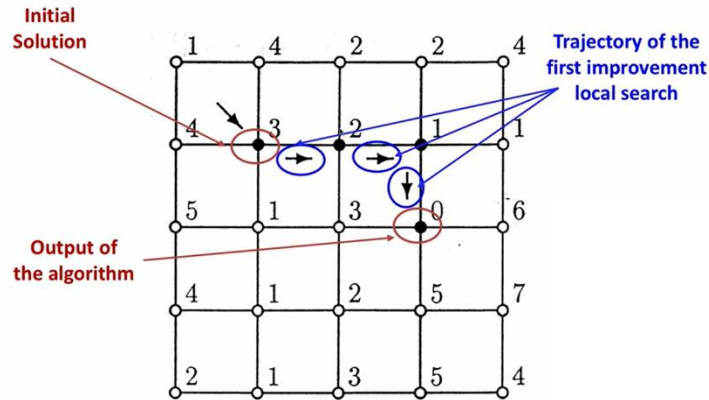


Figure 2.8: Exemple de l'algorithme (First improvement) [23]

### 2.5.1.3 Sélection aléatoire

Dans cette stratégie, une sélection aléatoire est appliquée aux voisins qui s'améliorent <sup>[12]</sup>.

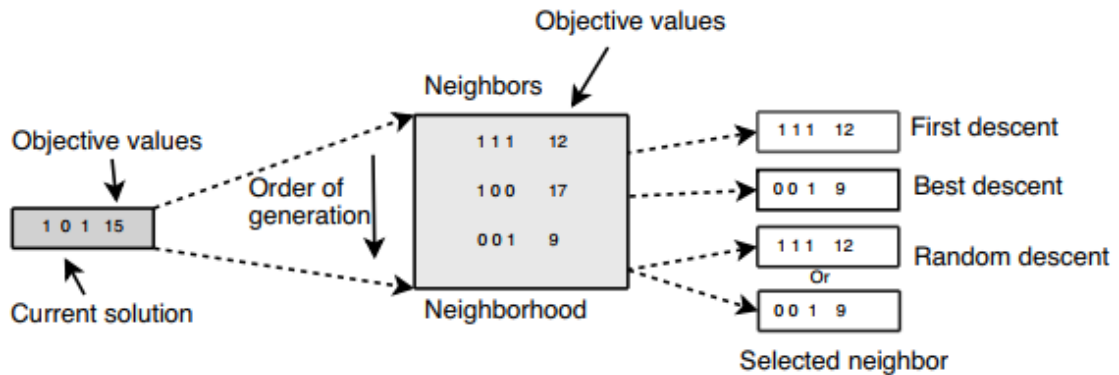


Figure 2.9: Stratégies de sélection pour améliorer les voisin [17]

## 2.6 Shuffled Frog Leaping Algorithm

L'algorithme de sauts de grenouilles mélangé ou bien Shuffled Frog Leaping Algorithm (SFLA) en anglais est un algorithme de recherche qui a été proposé par MM Eusuff et KE Lansey en 2003 pour résoudre les différents problèmes d'optimisation combinatoire [24]. Cet algorithme se compose d'un ensemble de population virtuelle interactive de grenouilles partitionnées en différents memplexes. Les grenouilles virtuelles agissent comme des hôtes où un hôte est une unité d'évolution culturelle. L'algorithme effectue simultanément une recherche locale indépendante dans chaque memplex. La recherche locale est complétée à l'aide d'une méthode de type optimisation par essaim de particules adaptée aux problèmes discrets. Pour assurer l'exploration globale, les grenouilles virtuelles sont périodiquement mélangées et réorganisées en nouveaux memplexes dans une technique similaire à celle utilisée dans l'algorithme d'évolution complexe mélangée (voir la figure 2.11). Les informations entre différents memplexes circulent à travers un processus de saut [25] (voir la figure 2.10). Dans chaque memplex, les grenouilles fournissent la meilleure solution  $X_b$  et la pire  $X_w$ . La grenouille donnant la meilleure solution dans toute la population est notée  $X_g$ . Lors de l'évolution d'un memplex, c'est-à-dire lors de l'exploration locale, la plus mauvaise grenouille saute vers la meilleure solution selon la règle suivante [26] :

$$D = r \times (|X_b - X_w|) \quad (0 < r < 1) \quad (2.1)$$

$$X_{w'} = X_w + D. \quad (2.2)$$

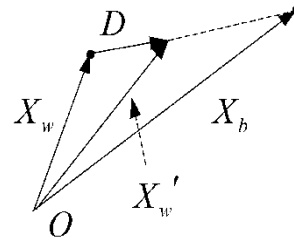


Figure 2.10 : Règle du saut de grenouille [27]

### Algorithm 5: SFLA [26]

**Step 1:** Set the size  $F$  of population, the number  $M$  of memplexes and the number  $N$  of iterations.

**Step 2:** Generate a random population of  $F$  solutions and evaluate each solution.

**Step 3:** Sort the population and determine the best solution  $X_g$ .

**Step 4:** Partition the population into  $M$  memplexes.

**Step 5:** Local search: for each memplex, repeat for  $N$  iterations:

- Determine the best solution  $X_b$  and the worst solution  $X_w$ .
- Calculate  $X_{w'}$  from  $X_b$  (apply equations 1 and 2)
- **if** ( $X_{w'}$  is better than  $X_w$ ) **then** replace  $X_w$  by  $X_{w'}$
- **else** calculate  $X_{w'}$  from  $X_g$  (apply equations 1 and 2 with replacing  $X_b$  by  $X_g$ )
  - **if** ( $X_{w'}$  is better than  $X_w$ ) **then** replace  $X_w$  by  $X_{w'}$
  - **else** Randomly generate  $X_{w'}$  and replace  $X_w$  by  $X_{w'}$
  - **end if**
- **end if**

**Step 6:** Bring together the  $M$  memplexes to build again the population.

**Step 7:** Go to step 3 if the stop criterion is not reached.

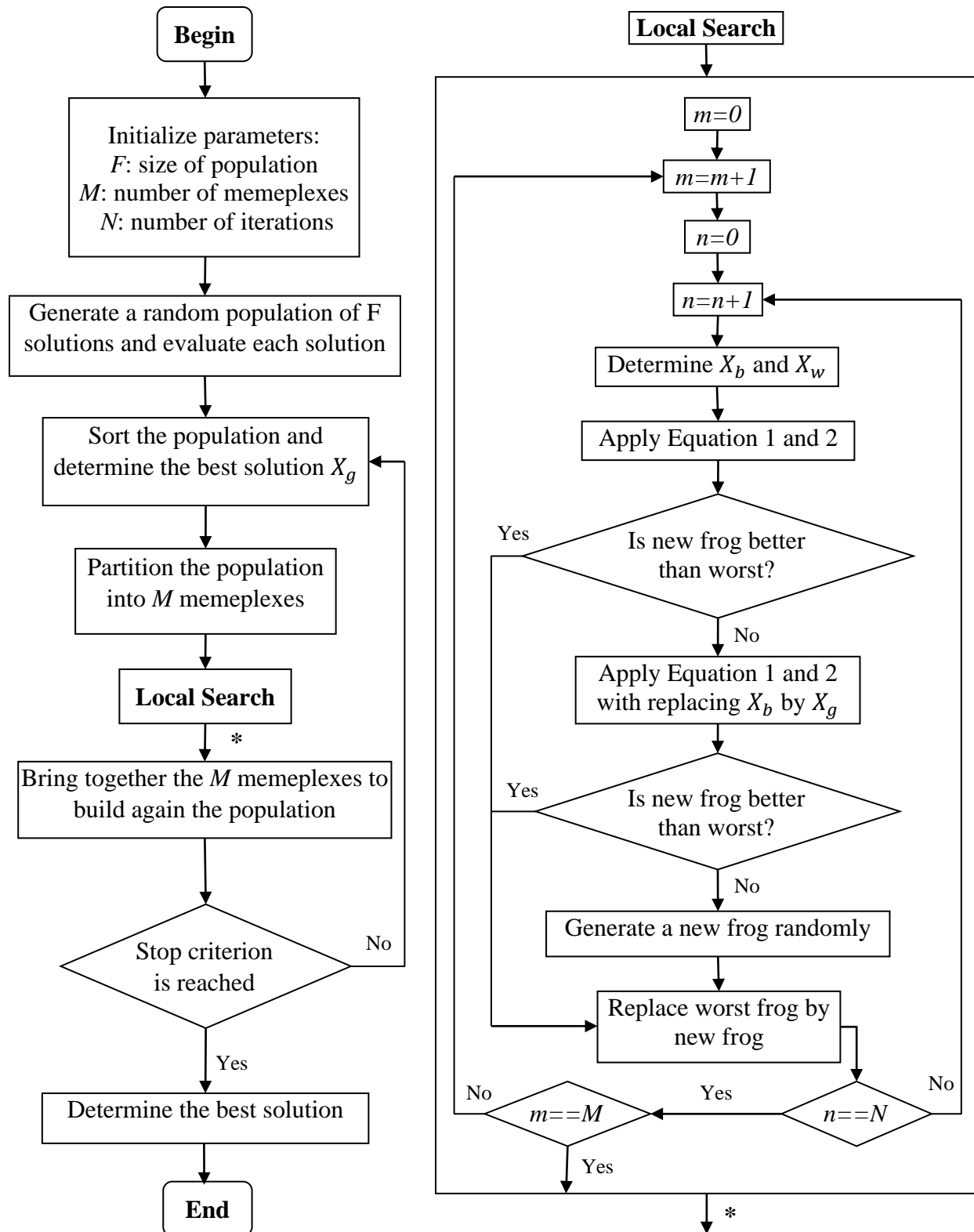


Figure 2.11: Méthodologie de l'algorithme SFLA [26]



## **2.7 Conclusion**

Dans ce chapitre, nous avons présenté et classifié les métaheuristiques et nous avons vu qu'elles appartiennent aux méthodes approximatives, ce qui signifie qu'elles ne garantissent pas de donner la solution optimale, mais de donner des solutions satisfaisantes qui sont assez bonnes en très peu de temps.

Après cela, nous avons vu trois algorithmes de la métaheuristique, l'un est la recherche aléatoire qui consiste sur l'exploration pure et l'autre est la recherche locale qui consiste sur l'exploitation et l'intensification. Le troisième algorithme SFLA est un algorithme de recherche métaheuristique basé sur la population inspirée des systèmes naturels.

Dans le chapitre suivant, On va appliquer l'algorithme SFLA pour l'ordonnancement des tâches dans le Cloud computing.

## Chapitre 3 : Implémentation de l'algorithme et évaluation des résultats obtenus

### **3.1 Introduction**

Le problème de l'ordonnancement des tâches est un problème qui peut être classé parmi les problèmes NP-complets en raison de sa complexité. Par conséquent, ce genre de problèmes peut être résolu par l'utilisation des métaheuristiques afin de trouver une solution optimale en un temps raisonnable.

Après avoir exposé dans le chapitre précédent certaines métaheuristiques, nous sommes intéressés dans ce chapitre à l'algorithme SFLA afin de l'adapter et l'appliquer pour l'ordonnancement des tâches dans le Cloud computing par rapport à 2 critères de QoS qui sont : le makespan et le coût. Dans ce cas, un individu de la population est un ordonnancement. Nous présentons également une évaluation mono-objectif afin de comparer le SFLA avec un autre algorithme prédéfini dans CloudSim qui est le Round Robin.

### **3.2 Outils de simulations**

Le développement de l'algorithme SFLA a été réalisé à l'aide des outils de simulation suivants :

#### **3.2.1 Java**

Java est un langage de programmation et une plateforme informatique créé par Sun Microsystems en 1995. Il est utilisé dans le développement web et le backend de plusieurs sites notamment Google, Amazon et YouTube etc. Grâce à son utilisation générale et avec sa structure orientée objet, il est devenu l'un des langages de programmation les plus utilisés aujourd'hui avec plus de 3 milliards d'appareils construites avec Java [28].

#### **3.2.2 NetBeans**

Apache NetBeans est un environnement de développement open source, une plate-forme d'outils et un Framework d'application qui a été développé par Sun Microsystems en juin 2000. Il permet la prise en charge native pour divers langages de programmation tels que Java, C, C++, JavaScript, etc. Il supporte également les outils de collaboration pour créer différentes applications telles que les applications Web, les applications mobiles et les systèmes embarqués [29].

En septembre 2016, Oracle, qui a racheté Sun, propose de céder le projet à la fondation Apache Software Foundation, ce qui est accepté en octobre 2016 [29].

#### **3.2.3 JFreeChart**

JFreeChart est un Framework open source pour le langage de programmation Java, qui permet aux développeurs d'afficher facilement des graphiques d'aspect professionnel dans leurs applications. Il a été développé par David Gilbert en février 2000, et il est disponible sous les termes de la licence LGPL.

### 3.2.4 CloudSim

Actuellement, la plupart des chercheurs dans le monde travaillent avec des fonds de recherche limités. Par conséquent, il n'est pas toujours visible pour eux de louer l'infrastructure nécessaire pour exécuter leurs tests répétitifs. De plus, la flexibilité de changer l'environnement d'exécution principal est également limitée dans l'environnement de Cloud. De même, les changements de charge en temps réel du fournisseur de services Cloud affecteront la qualité de ses résultats de test. Par conséquent, il ya un besoin d'un environnement stimulant pour obtenir une bonne qualité des résultats de ces tests. A partir de là, CloudSim est l'un des outils qui peuvent répondre à ce besoin dans une certaine mesure.

CloudSim est un Framework de simulation de scénarios Cloud développé dans le laboratoire du département d'informatique et de génie logiciel de l'université de Melbourne. Il est composé d'un ensemble de bibliothèques de classes écrites en Java pour fournir des services informatiques fiables, sécurisés, tolérants aux pannes et évolutifs. De plus, il décrit les centres de données, des ressources informatiques, des machines virtuelles pour évaluer les performances de nouveaux algorithmes et applications avant le développement réel de produits Cloud [30].

La figure 3.1 représente la structure de programmation CloudSim et ses éléments de base. Le modèle CloudSim peut être représenté comme un ensemble de classes où chaque classe contient ses propres composants. Pour illustrer, un serveur physique a la capacité d'attribuer des traitements préconfigurés comme la mémoire, le stockage et même la politique de provisionnement pour allouer des cœurs de traitement aux VMs [4].

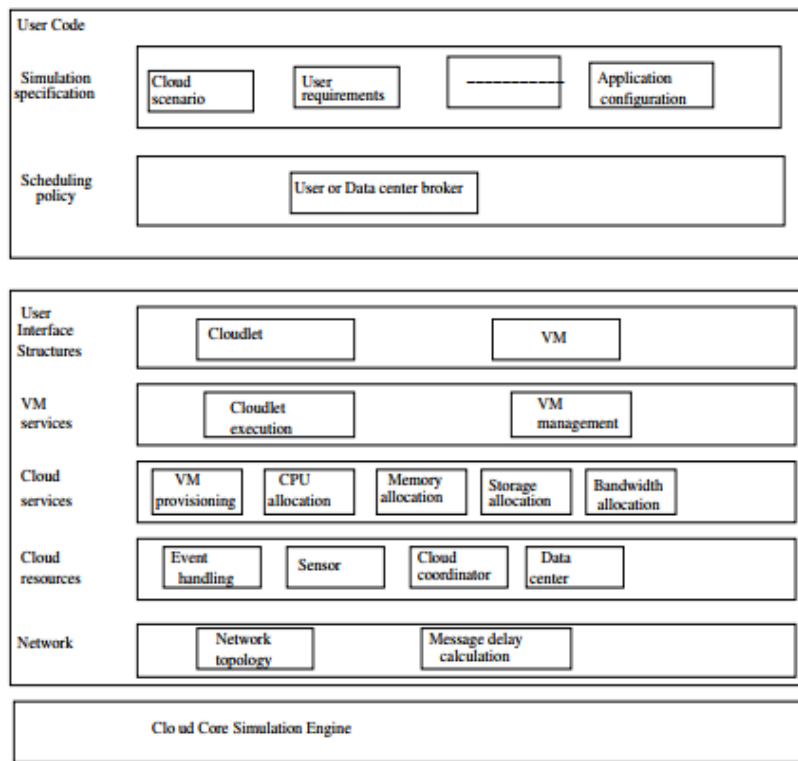


Figure 3.1: Architecture de CloudSim [4]

### 3.3 Critères d'évaluation

Dans le cadre de ce travail, nous avons utilisé les deux métriques de QoS suivantes :

- **Makespan** : est le temps total pris par les ressources pour terminer l'exécution de toutes les tâches. L'utilisation de la machine virtuelle est définie comme la qualité de l'utilisation des ressources dans le cloud [31].
- **Coût** : varie selon le besoin mais il est recommandé de payer des frais supplémentaires pour des performances optimales des VMs.

### 3.4 IHM développée

CloudSim s'exécute uniquement en mode console et n'a pas d'interface graphique, par conséquent nous avons créé une IHM pour faciliter la manipulation de la simulation.

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

**Interface principale** : dès le lancement de notre application, la fenêtre d'accueil illustrée dans la figure 3.2 apparaît, elle contient les informations de notre PFE.



Figure 3.2: Interface principale

**Configuration des paramètres de l'algorithme SFLA** : pour configurer le Cloud, nous commençons par saisir les paramètres nécessaires de l'algorithme SFLA comme la taille de population, la taille du memeplex, le nombre d'itérations locale et globale (voir la figure 3.3).

Figure 3.3 : Caractéristiques de l'algorithme SFLA

**Configuration des machines physiques :** une fois nous avons configuré les caractéristiques de l'algorithme. Nous devons configurer les machines physiques qui sont caractérisées par le nombre des Datacenters, le nombre des hôtes qui se trouvent dans chaque Datacenter, le nombre de processeur (PE) dans chaque hôte, la vitesse de chaque processeur (MIPS), la RAM, la bande passante et aussi la capacité de stockage (voir la figure 3.4).

Figure 3.4 : Création des machines physiques

**Configuration des machines virtuelles :** nous devons maintenant configurer les machines virtuelles VMs, de telle sorte qu'un hôte peut être alloué à un ensemble de machines virtuelles. Une machine virtuelle est caractérisée par sa vitesse de traitement (MIPS), son taux d'échec, sa capacité de RAM, sa bande passante ainsi que son stockage (voir la figure 3.5).

Accueil Param de SFLA Machine Physique Machine Virtuelle Cloudlets Simulation Pondération

**Caractéristiques des machines virtuelles**

Nombre VMs

RAM min  RAM max

BW min  BW max

Storage

MIPS min  MIPS min

Retour Suivant

Figure 3.5 : Création des machines virtuelles

**Configuration des Cloudlets :** cette IHM permet de définir les caractéristiques des cloudlets telle que leurs nombres et leurs longueurs.

Accueil Param de SFLA Machine Physique Machine Virtuelle Cloudlets Simulation Pondération

**Caractéristiques des Cloudlets**

Nombre Cloudlets

Length min  Length max

Retour Suivant

Figure 3.6 : Création des Cloudlets

**Simulation :** dans cette IHM, un nombre de simulation doit être attribué pour obtenir des résultats plus corrects (plus le nombre de simulation est élevé, plus les résultats sont corrects). De même, vous devez spécifier le type d'ordonnancement selon l'étude que vous souhaitez avoir.

Accueil Param d'SFLA Machine Physique Machine Virtuelle Cloudlets Simulation Pondération

Type d'ordonnancement

Nombre de simulation

☒ Ordonnancement mono-objectif

☐ Ordonnancement multi-objectif

Retour Suivant

Figure 3.7 : Type d'ordonnancement

**Configuration des pondérations :** si l'ordonnancement multi-objectif est choisi, il faudra donc attribuer une pondération à chaque critère (makespan et coût) afin de le favoriser ou le défavoriser dans chaque individu de la population. A la fin, on clique sur le bouton « Démarrer » pour afficher les résultats obtenus.

Accueil Param d'SFLA Machine Physique Machine Virtuelle Cloudlets Simulation Pondération

Configuration des pondérations

Pondération 1

Makespan Coût

Pondération 2

Makespan Coût

Retour Démarrer

Figure 3.8 : Configuration des Pondérations



### 3.5 Résultats obtenus et étude comparative

Cette section explique la configuration des différentes simulations ainsi que les résultats obtenus après avoir exécuté l'algorithme SFLA pour l'ordonnancement des tâches.

Les simulations ont été réalisées dans un environnement hétérogène comme indiquer dans le tableau 3.1 définissant la valeur associée à chaque paramètre.

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

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

Les simulations ont été réalisées sous différentes machines sous Windows 10 de 64 bits, et le temps d'exécution diffère selon le type de machine utilisé (voir le tableau 3.2) :

| CPU        | Nombre de cœurs | Vitesse d'horloge | RAM         | Hard Disk | Moyenne de temps d'exécutions |
|------------|-----------------|-------------------|-------------|-----------|-------------------------------|
| I7-3537U   | 2               | 2.0 GHz           | 8 GB - DDR3 | SSD       | 05 min 06 sec                 |
| I5-5005U   | 2               | 2.0 GHz           | 4 GB – DDR3 | SSD       | 02 min 25 sec                 |
| I5 10-35G1 | 8               | 1.0 GHz           | 8 GB – DDR3 | HDD       | 39 sec                        |

*Tableau 3.2 : Spécification des machines utilisées.*

### 3.5.1 Evaluation de l'ordonnancement mono-objectif

Pour la validation de l'ordonnancement mono-objectif, une comparaison a été réalisée entre le Round Robin (RR) et le SFLA en termes de makespan (mesuré en secondes). Les 4 figures suivantes représentent les résultats de comparaison pour un nombre de VM égal à 40, 60, 80 et 100 respectivement.

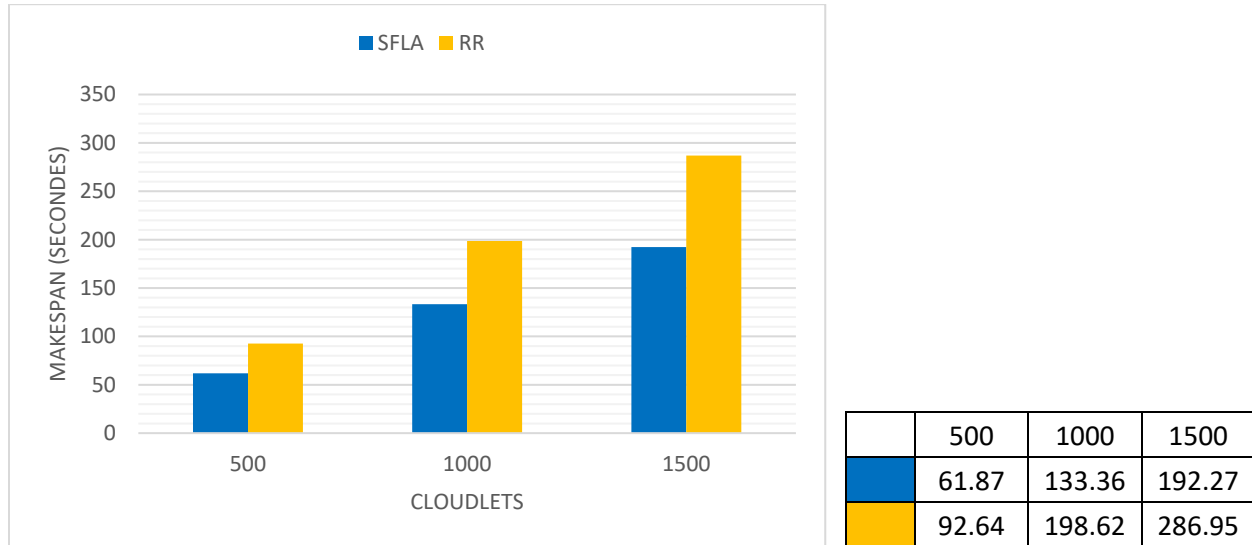


Figure 3.9 : Comparaison entre SFLA et RR en termes de makespan pour 40 VMs

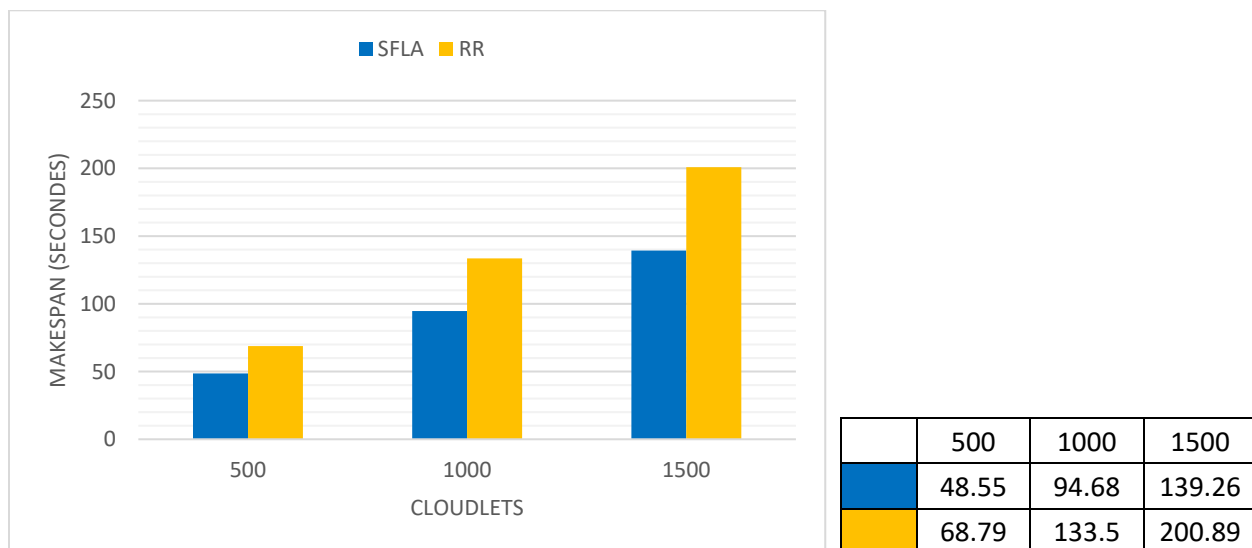


Figure 3.10 : Comparaison entre SFLA et RR en termes de makespan pour 60 VMs

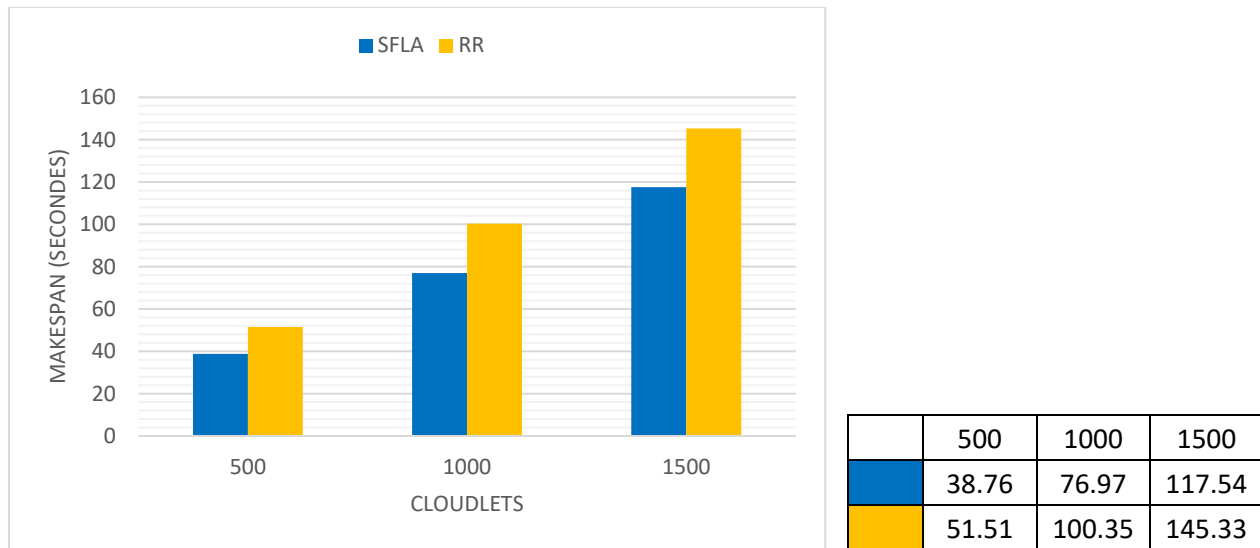


Figure 3.11 : Comparaison entre SFLA et RR en termes de makespan pour 80 VMs

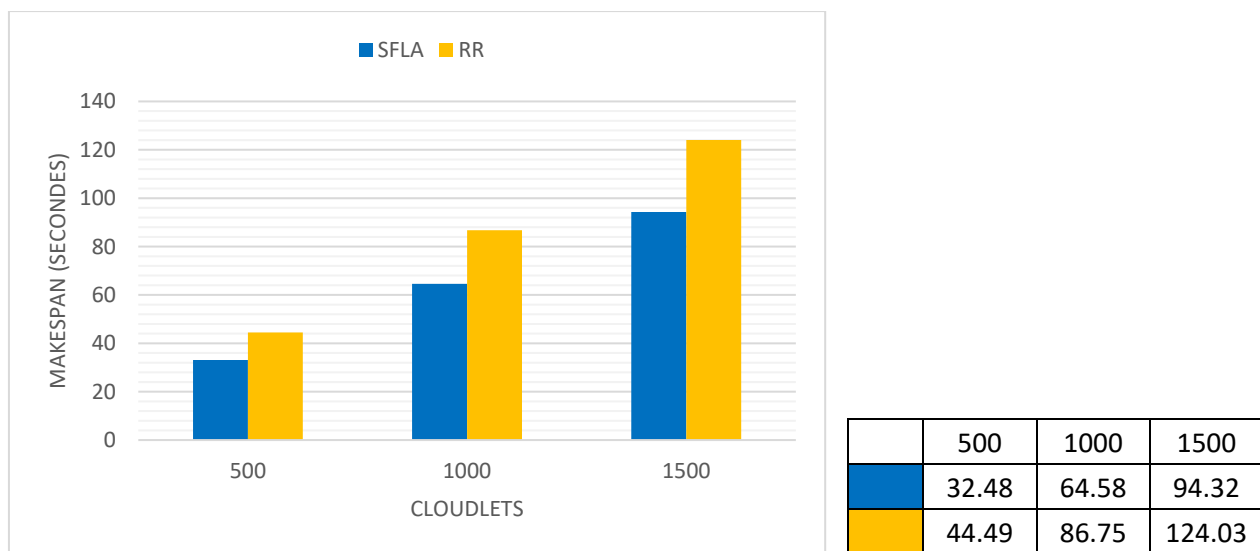


Figure 3.12 : Comparaison entre SFLA et RR en termes de makespan pour 100 VMs

### Résultats :

D'après les graphes obtenus, l'algorithme SFLA a montré son efficacité en termes de makespan par rapport à Round Robin (RR) pour toutes les instances de simulation appliquées. On remarque également que la valeur makespan diminue selon le nombre de VM utilisées.

### 3.5.2 Evaluation de l'ordonnancement multi-objectif

Pour la validation de l'ordonnancement multi-objectif, une autre comparaison a été réalisée en utilisant deux versions pour l'évaluation des solutions dans l'algorithme SFLA selon l'évaluation des critères d'optimisation.

Pour les deux critères, nous avons utilisé trois vecteurs de poids différents « 0.8, 0.2 », « 0.5, 0.5 » et « 0.2, 0.8 » correspondant respectivement aux critères « Makespan, Coût ». Le premier vecteur « 0.8, 0.2 » est utilisé pour favoriser le makespan, le troisième vecteur « 0.2, 0.8 » est utilisé pour favoriser le coût. Le deuxième vecteur « 0.5, 0.5 » donne des poids équitables aux deux métriques.

### 3.5.2.1 Comparaison par rapport au makespan

Les 4 figures suivantes représentent les résultats de comparaison par rapport au makespan pour un nombre de VM égal à 40, 60, 80 et 100 respectivement.

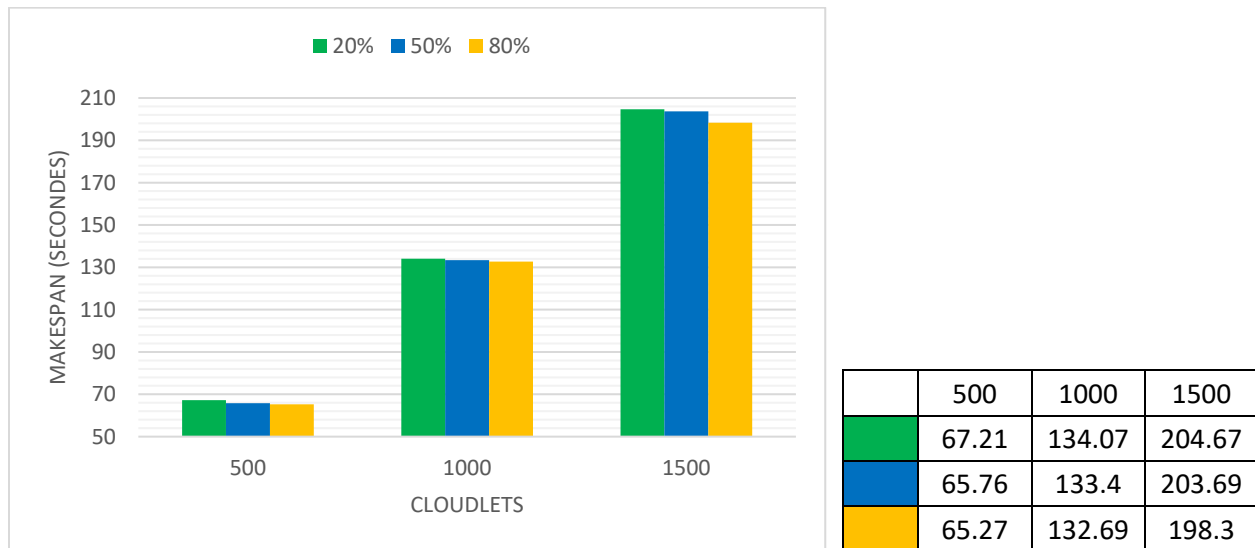


Figure 3.13 : Comparaison en termes de makespan pour 40 VMs

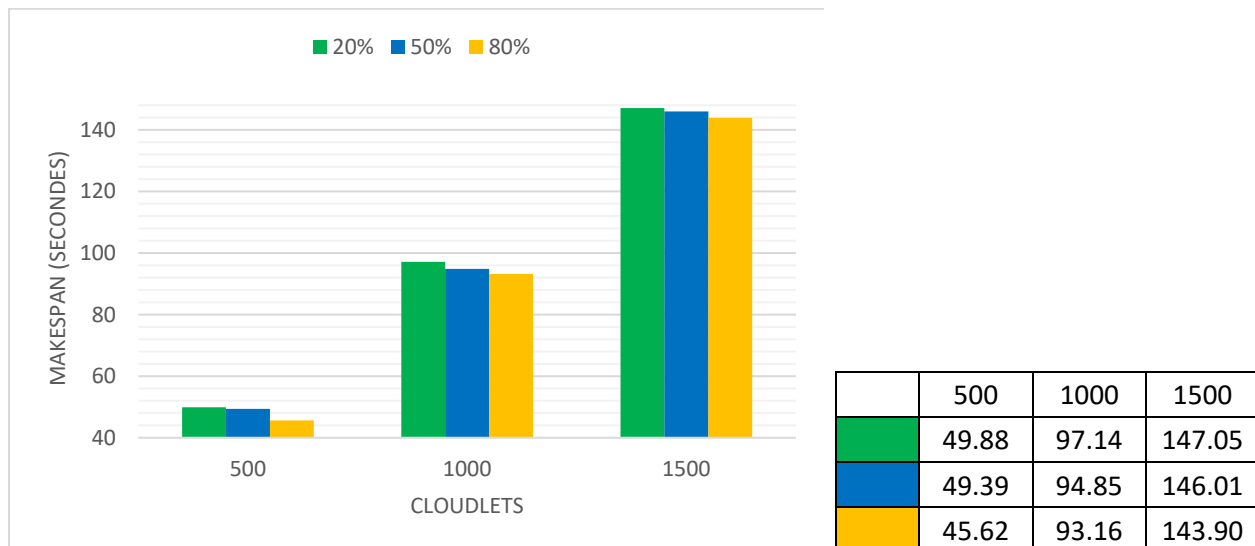


Figure 3.14 : Comparaison en termes de makespan pour 60 VMs

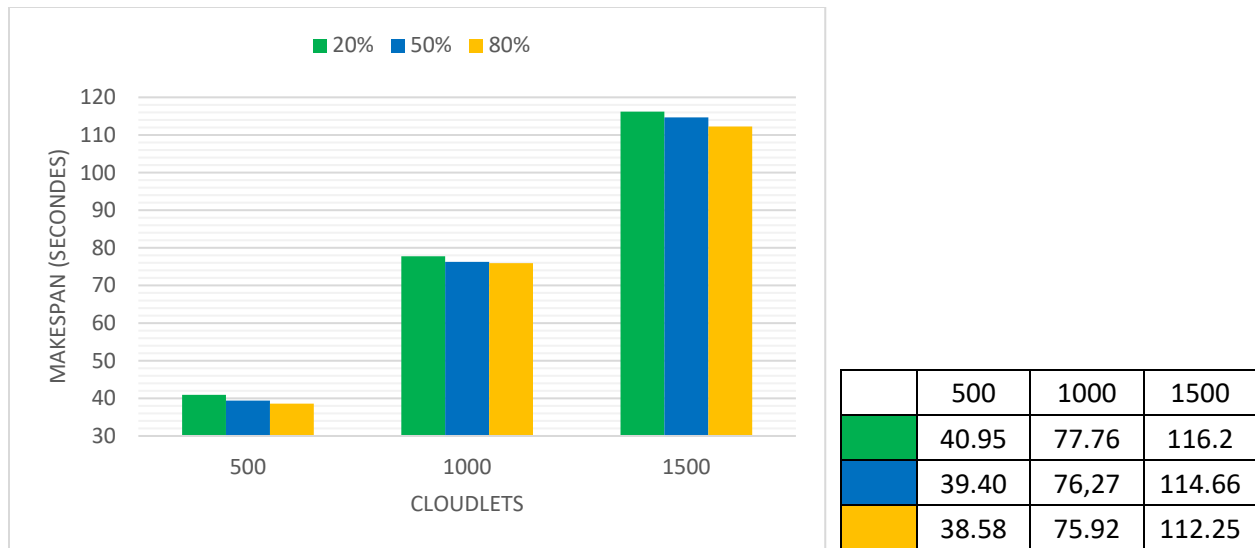


Figure 3.15 : Comparaison en termes de makespan pour 80 VMs

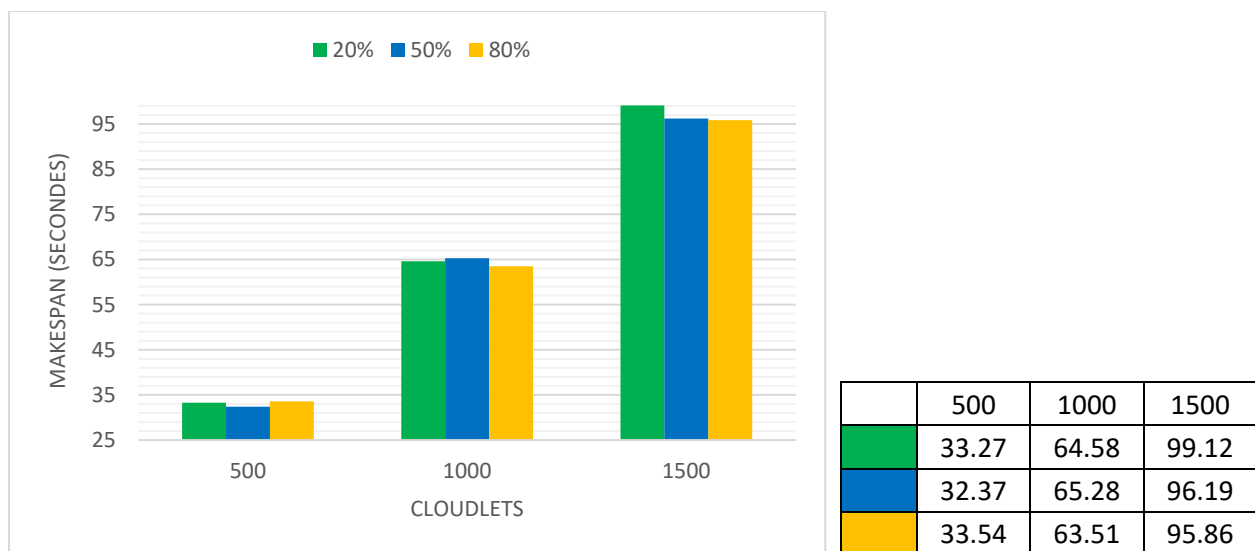


Figure 3.16 : Comparaison en termes de makespan pour 100 VMs

### Résultats :

Peu importe le nombre de VMs et le nombre de tâches, la meilleure façon d'optimiser le makespan est d'utiliser le vecteur de poids « 0.8, 0.2 » car cette dernière pondération est la plus intéressante pour le makespan.

### 3.5.2.2 Comparaison par rapport au coût

Les 4 figures suivantes représentent les résultats de comparaison par rapport au coût pour un nombre de VM égal à 40, 60, 80 et 100 respectivement.

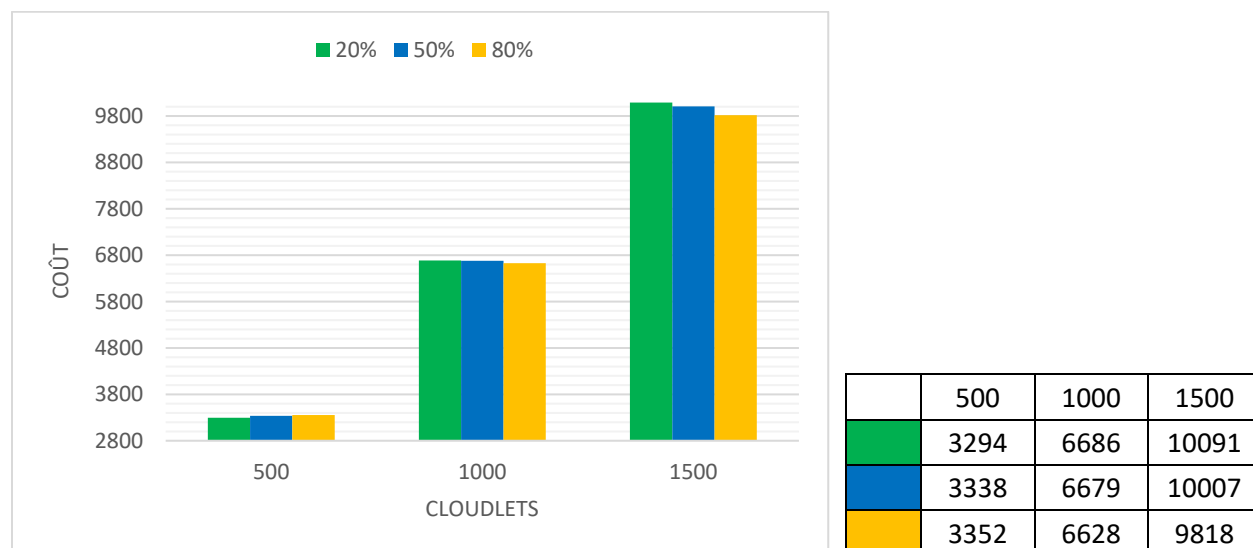


Figure 3.17 : Comparaison en termes de coût pour 40 VMs

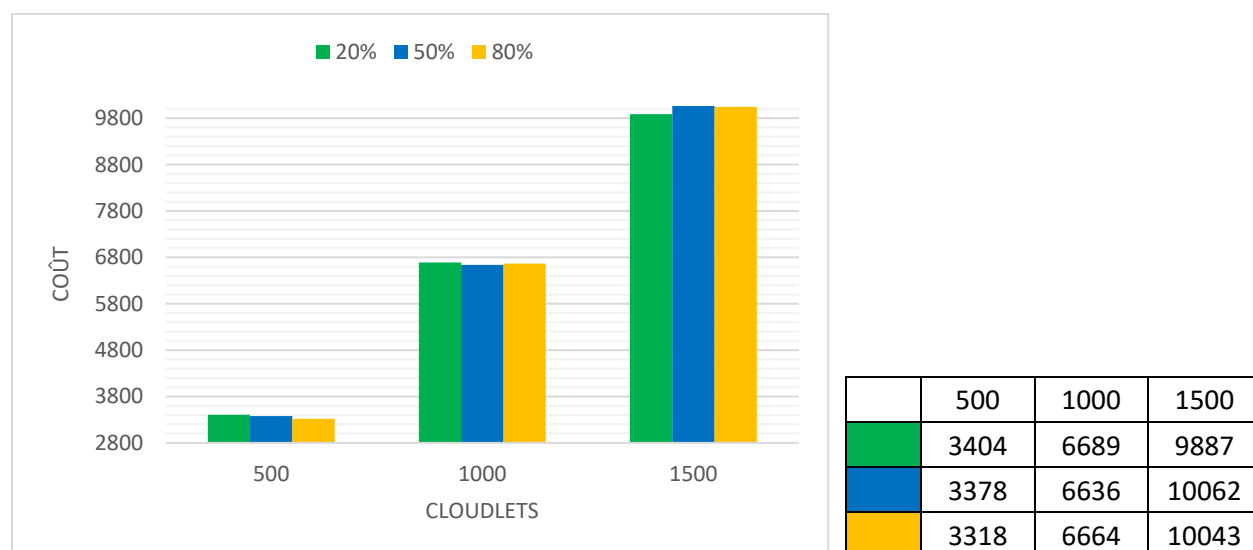


Figure 3.18 : Comparaison en termes de coût pour 60 VMs

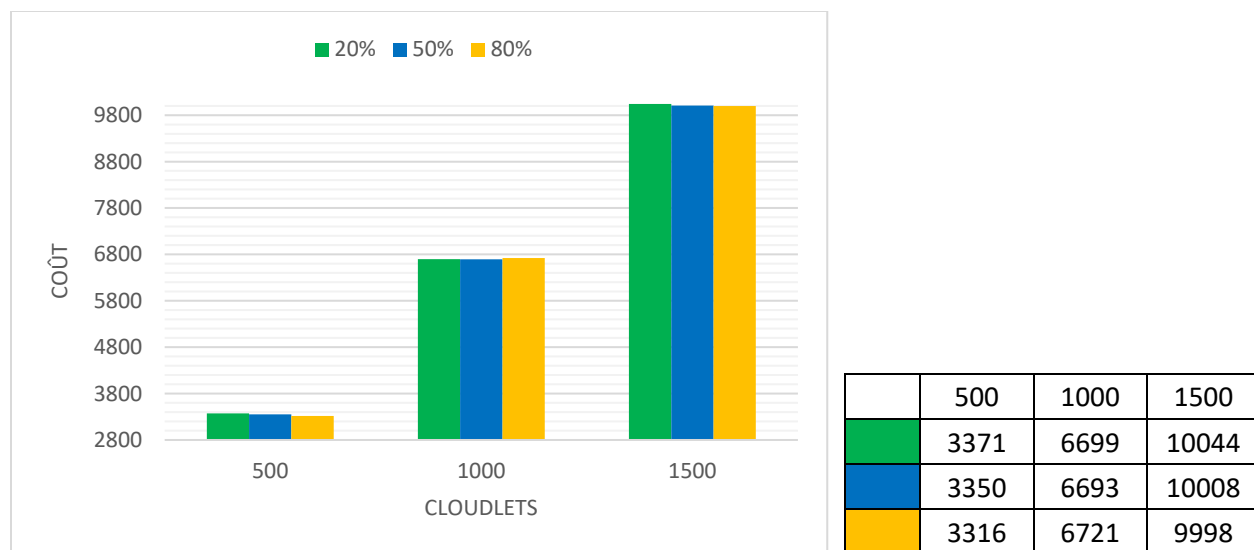


Figure 3.19 : Comparaison en termes de coût pour 80 VMs

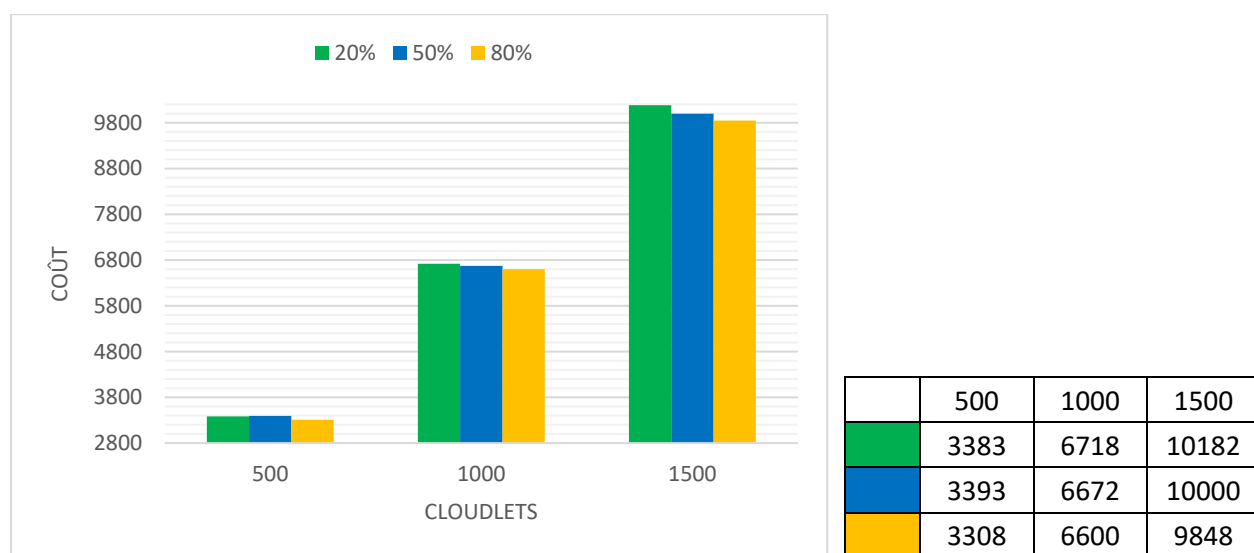


Figure 3.20 : Comparaison en termes de coût pour 100 VMs

### Résultats :

Peu importe le nombre de VMs et le nombre de tâches, la meilleure façon d'optimiser le coût est d'utiliser le vecteur de poids « 0.2, 0.8 » car cette dernière pondération est la plus intéressante pour le coût.

### **3.6 Conclusion**

Dans ce dernier chapitre de ce PFE., nous avons présenté les différents outils de simulation et leurs contributions dans la création d'IHM, le développement de l'algorithme ainsi que l'obtention des résultats. Nous avons remarqué que SFLA est meilleur que RR en termes de makespan dans l'ordonnancement mono-objectif. Par ailleurs, dans l'ordonnancement multi-objectif, les résultats sont un peu différents d'une configuration à une autre mais la meilleure façon d'optimiser le makespan ou le coût est de lui affecter la pondération la plus élevée.



## Conclusion générale

Le Cloud computing est un moyen d'augmenter la capacité de différentes ressources ou d'ajouter des fonctionnalités sans investir dans une nouvelle infrastructure ou obtenir de nouveaux logiciels. Il permet la réalisation de diverses économies dans le déploiement et l'exploitation des solutions informatiques. Cependant, lorsqu'il y a plusieurs critères à considérer pour fournir une QoS acceptable, le problème d'optimisation à résoudre devient un problème d'optimisation multi-objectif avec l'utilisation de poids adaptés pour chaque critère.

Dans le cadre de ce PFE, nous avons simulé une métaheuristique pour résoudre le problème de l'ordonnancement des tâches dans le Cloud computing. Il s'agit de l'algorithme Shuffled Frog Leaping Algorithm qui permet de trouver la solution optimale en un délai acceptable en évaluant deux métriques de QoS qui sont le makespan et le coût.

Pour la réalisation de notre contribution, nous avons étudié deux types d'ordonnancement afin de mener diverses expériences.

Le premier type d'ordonnancement est un ordonnancement mono-objectif qui consiste à traiter ce problème par rapport à un seul critère qui est le makespan. Cette expérimentation a été faite en le SFLA avec un autre algorithme déjà pris en compte dans CloudSim qui est le Round Robin. Les résultats obtenus prouvent que le SFLA est plus performants que le Round Robin.

Le deuxième type d'ordonnancement est un ordonnancement multi-objectif qui consiste à traiter ce problème par rapport à deux critères qui sont le makespan et le coût. Cette expérimentation a été faite par l'utilisation de vecteurs de poids afin de favoriser chaque critère selon le choix de l'utilisateur. D'après les résultats obtenus, nous pouvons conclure que la meilleure façon d'optimiser une métrique est de lui affecter la pondération la plus pertinente.

Au regard de notre travail accompli dans le cadre de ce PFE, il serait intéressant d'élargir le problème d'optimisation à d'autres algorithmes et d'autres critères tels que l'énergie, la fiabilité, etc.

## Références

- [1] Buyya, Rajkumar, James Broberg, and Andrzej M. Goscinski, eds. Cloud computing: Principles and paradigms. John Wiley & Sons, 2010.
- [2] Cloud Deployment Models. Disponible sur : <https://www.geeksforgeeks.org/Cloud-deployment-models/>. Consulté le 15 juin 2022.
- [3] IaaS vs PaaS vs SaaS Enter the Ecommerce Vernacular: What You Need to Know, Examples & More. Disponible sur : <https://www.bigcommerce.com/blog/saas-vs-paas-vs-iaas/#the-three-types-of-Cloud-computing-service-models-explained/>. Consulté le 15 juin 2022.
- [4] Manvi, Sunilkumar, and Gopal K. Shyam. Cloud Computing: Concepts and Technologies. CRC Press, 2021.
- [5] Bisong, Anthony, and M. Rahman. "An overview of the security concerns in enterprise Cloud computing." arXiv preprint arXiv:1101.5613 (2011).
- [6] 7 Most Infamous Cloud Security Breaches. Disponible sur : <https://blog.storagecraft.com/7-infamous-Cloud-security-breaches/>. Consulté le 15 juin 2022.
- [7] How Does Cloud Encryption Work? <https://www.mcafee.com/enterprise/en-us/security-awareness/Cloud/how-does-Cloud-encryption-work.html>. Consulté le 15 juin 2022.
- [8] Characteristics, Types and Applications of Cryptography. Disponible sur : <https://www.analyticssteps.com/blogs/characteristics-types-and-applications-cryptography>. Consulté le 15 juin 2022.
- [9] Namrata Bisht. Virtualization In Cloud Computing and Types. Disponible sur : <https://www.geeksforgeeks.org/virtualization-Cloud-computing-types/>. Consulté le 15 juin 2022.
- [10] Quels sont les différents types de virtualisation informatique ? Disponible sur : <https://www.axido.fr/quels-sont-les-differents-types-de-virtualisation-informatique/>. Consulté le 15 juin 2022.
- [11] Les différents types de virtualisations. Disponible sur : <https://techaxxom.fr/my/gfssio2user02/Blog/SiteAssets/SitePages/Veille%20Technologique/La%20virtualisation.pdf>. Consulté le 15 juin 2022.
- [12] Parwekar, Pritee. "From internet of things towards Cloud of things." 2011 2nd international conference on computer and communication technology (ICCCT-2011). IEEE, 2011.
- [13] Eurotech easing 'Internet of Things' with release of M2M Everywhere Cloud 2.0. Disponible sur : <https://jaxenter.com/eurotech-easing-internet-of-things-with-release-of-m2m-everyware-cloud-2-0-104570.html>. Consulté le 15 juin 2022.
- [14] Ibrahim, Ibrahim Mahmood. "Task scheduling algorithms in cloud computing: A review." Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12.4 (2021): 1041-1053.
- [15] Xu, Jialei, et al. "A many-objective optimized task allocation scheduling model in cloud computing." Applied Intelligence 51.6 (2021): 3293-3310.

- [16] Hachimi, Hanaa. Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications. Diss. INSA de Rouen; École Mohammadia d'ingénieurs (Rabat, Maroc), 2013.
- [17] Talbi, El-Ghazali. Metaheuristics: from design to implementation. Vol. 74. John Wiley & Sons, 2009.
- [18] Talbi, El-Ghazali. "Single solution based metaheuristics." Metaheuristics Des. Implement 74 (2009): 87-189.
- [19] Heuristique - Définition et Explications. Disponible sur : <https://www.techno-science.net/definition/6418.html>. Consulté le 15 juin 2022.
- [20] Metaheuristics classification. Disponible sur : [https://commons.wikimedia.org/wiki/File:Metaheuristics\\_classification.svg](https://commons.wikimedia.org/wiki/File:Metaheuristics_classification.svg). Consulté le 15 juin 2022.
- [21] Lones, Michael. "Sean Luke: essentials of metaheuristics." (2011): 333-334.
- [22] Soft Computing and Intelligent Information Systems. Disponible sur : <https://sci2s.ugr.es/node/124>. Consulté le 15 juin 2022.
- [23] Zäpfel, Günther, Roland Braune, and Michael Bögl. "Metaheuristic search concepts: A tutorial with applications to production and logistics." (2010).
- [24] A Survey on Shuffled Frog-Leaping Algorithm. Disponible sur : <https://towardsdatascience.com/a-survey-on-shuffled-frog-leaping-algorithm-d309d0cf7503>. Consulté le 15 juin 2022.
- [25] Eusuff, Muzaffar, Kevin Lansey, and Fayzul Pasha. "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization." Engineering optimization 38.2 (2006): 129-154.
- [26] Eusuff MM, Lansey KE. Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of Water Resources planning and management. 2003 May;129(3):210-225.
- [27] Wang, Jie-Sheng, Jiang-Di Song, and Jie Gao. "Rough set-probabilistic neural networks fault diagnosis method of polymerization kettle equipment based on shuffled frog leaping algorithm." Information 6.1 (2015): 49-68.
- [28] Java Tutorial. Disponible sur : <https://www.javatpoint.com/java-tutorial>. Consulté le 15 juin 2022.
- [29] Documentation netbeans. Disponible sur : <https://fr.wikipedia.org/wiki/NetBeans>. Consulté le 15 juin 2022.
- [30] The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. Disponible sur: <http://www.cloudbus.org/>. Consulté le 15 juin 2022.
- [31] Sasikaladevi, N. "Minimum makespan task scheduling algorithm in cloud computing." *International Journal of Grid and Distributed Computing* 9.11 (2016): 61-70.

## Résumé

Le Cloud computing est la disponibilité à la demande de différentes ressources matérielles et logicielles du système informatique à travers l'internet. Dans ce contexte, l'ordonnancement des tâches est un élément très important pour le bon fonctionnement du système. Les fournisseurs et les utilisateurs des services Cloud ont des objectifs souvent contradictoires et l'ordonnanceur doit tenir compte de ce type de contrainte. Par conséquent, l'ordonnancement des tâches dans le Cloud computing devient un problème d'optimisation multi objectif. Dans ce travail, nous avons appliqué l'algorithme SFLA pour l'ordonnancement des tâches multi-objectif dans le Cloud computing. L'implémentation de l'algorithme a été réalisée par l'outil de simulation CloudSim et les résultats obtenus sont très satisfaisants.

**Mots Clés :** Cloud computing, ordonnancement des tâches, SFLA, CloudSim.

## ملخص

الحوسبة السحابية هي التوافر عند الطلب لموارد أجهزة وبرامج أنظمة الحاسوب المختلفة عبر الإنترنت. في هذا السياق، تعد جدولة المهام عنصرًا مهمًا للغاية من أجل حسن سير النظام. غالبًا ما يكون لمقدمي ومستخدمي الخدمات السحابية أهداف متناقضة ويجب أن يأخذ المجدول هذا النوع من القيد في الاعتبار. لذلك، تصبح جدولة المهام في الحوسبة السحابية مشكلة تحسين متعددة الأغراض. في هذا العمل، طبقنا خوارزمية SFLA لجدولة المهام متعددة الأهداف في الحوسبة السحابية. تم تنفيذ الخوارزمية بواسطة أداة المحاكاة CloudSim وكانت النتائج التي تم الحصول عليها مرضية للغاية.

**الكلمات المفتاحية :** الحوسبة السحابية، جدولة المهام، SFLA، CloudSim.

## Abstract

Cloud computing is the on-demand availability of various hardware and software resources of the computer system through the Internet. In this context, task scheduling is a very important element for the proper functioning of the system. Providers and users of Cloud services often have conflicting goals and the scheduler has to take this kind of constraint into account. Therefore, task scheduling in Cloud computing becomes a multi-objective optimization problem. In this work, we applied the SFLA algorithm for multi-objective task scheduling in Cloud computing. The implementation of the algorithm was done by the CloudSim simulation tool and the results obtained are very satisfactory.

**Key words:** Cloud computing, task scheduling, SFLA, CloudSim.