***REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE***

*MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE*

*LA RECHERCHE SCIENTIFIQUE*

***UNIVERSITE MOULOUD MAMMERI TIZI-OUZOU***

*FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE*

***DEPARTEMENT D'INFORMATIQUE***

# *Memoire*

**DE FIN D'ETUDES**

**D'un MASTER ACADEMIQUE**

***Domaine : Mathématiques et Informatiques***

***Filière : Informatique***

***Spécialité : Réseaux mobilités et systèmes embarqués***

## *Thème :*

## *Implémentation de l'algorithme BLA pour le Problème d'ordonnancement*

***Proposé et dirigé par :***

*M me OUKFIF Karima.*

***Réalisé par :***

*M elle Ounadi Fadhela.*

*M elle Bennour Kamelia.*

*Membres de jury :*

*M r DAOUI Mehammed*
*M me. BELKADI Malika*
*M me .Aoudjit Rachida*

*Année universitaire 2018/2019*

# Remerciement

# Remerciements

*Nous tenons à remercier Dieu tout puissant qui nous a armés de courage, de volonté et surtout de patience.*

*Nous exprimons notre sincère gratitude à toutes les personnes qui ont rendu ce mémoire possible par leurs aides et leurs contributions.*

*Nos premiers remerciements sont adressés tout d'abord à notre promotrice Mme OUKFIF pour nous avoir proposé ce sujet et de l'avoir dirigé. Ses conseils et ses encouragements nous ont été précieux, nous apprécions aussi son regard critique sur ce travail.*

*Nous tenons également à remercier les membres de jury qui ont eu l'amabilité d'examiner et de juger notre mémoire.*

*Nous gardons une place toute particulière à nos familles, nous leurs exprimons toute notre profonde reconnaissance car ils nous ont constamment aidé, par leur soutien moral et leurs encouragements pour achever ce travail.*

# Dédicaces

*Dédicace*

*Je dédie ce modeste travail*

*À MES CHERS PARENTS*

*Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être. Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours. Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitterai jamais assez. Puisse Dieu, le Très Haut, vous accorder santé, bonheur et longue vie et faire en sorte que jamais je ne vous déçoive.*

*A Mon CHER ET ADORABLE FRERE*

*Amine le généreux, en témoignage de mon affection fraternelle, de ma profonde tendresse et reconnaissance, je te souhaite une vie pleine de bonheur et de succès et que Dieu, le tout puissant, te protège et te garde.*

*Aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés, et qui m'ont accompagnaient durant mon chemin d'études supérieures, mes aimables amis, collègues de travail, et frères de cœur, toi Lynda Djaferi, Ferroudja Djermane, Radia Lounas, Kanza Debiane, Massinissa, Dr Slimani Zohir, Naima Bouksil, Mr Ziane Mustapha et Mr Hadj kaddour Ferhat.*

*A MA BINOME FADILA*

*Je te remercie pour ton sérieux et ta ponctualité. Je te souhaite une vie plaine du bonheur et de succès.*

*UNE SPECIALE DEDICACE A CETTE PERSONNE QUI MA AIDEE POUR LA REALISATION DE CE TRVAIL, ET POUR QUI JE PORTE BEAUCOUP DE RESPECT ET D'ADMIRATION. A TOI SOHAIB*

Bennour Kamelia

# *Dédicaces*

Je dédie ce travail :

A la mémoire de mon père je ne saurais exprimer mon grand chagrin à ton absence

A ma très chère maman

Aucune dédicace me saurait exprimer mon grand respect, et ma reconnaissance pour

Les sacrifices que tu as consentis pour mon éducation. J'implore dieu le tout puissant

De vous accorder la bonne santé et langue vie.

A mon fiancé à Qui je dois une grande part de la réussite de mon projet grâce a son soutien et ses conseils ainsi que sa compréhension et à qui je souhaite tout le bonheur du monde.

Ma vie ne serait pas aussi magique sans ton présence et ton amour.

A mes deux frères, a toute mes sœurs, et toute ma famille, puisse dieu vous donne santé, bonheur, courage et surtout réussite.

A toutes mes amies Dihia, Joujou et T.Dihia à qui je souhaite le sucés et la réussite.

E pour en finir, A ma très chère binôme Kamelia, avec qui je partage ce travail, je te remercie pour ton sérieux, ton soutient dans la réalisation de ce travail de recherche je te souhaite la réussite et tout le bonheur. Ainsi que toute ta famille.

*Ounadi Fadhela*

# Sommaire

| | |
|---|---|
| Introduction Générale | 2 |
| **Chapitre I** | |
| ***Généralités sur le Cloud Computing et le Fog Computing*** | |
| Introduction | 3 |
| I. Le Cloud Computing | 3 |
| I.1 Définition et concept du Cloud Computing | 3 |
| I.2 Historique du Cloud Computing | 5 |
| I.3 Les modèles de déploiement du Cloud Computing | 6 |
| I.3.1 Le Cloud public | 6 |
| I.3.2 Le Cloud privé | 6 |
| I.3.3 Le Cloud Communautaire | 7 |
| I.3.4 Le Cloud hybride | 8 |
| I.4 Les modèles de services de Cloud Computing | 8 |
| I.4.1 IaaS (Infrastructure as a service) | 9 |
| I.4.2 PaaS (Platform as a service) | 10 |
| I.4.3 SaaS (Software as a service) | 11 |
| I.5 Contrainte de l'infrastructure | 12 |
| I.6 La virtualisation dans le Cloud Computing | 13 |
| I.6.1 Définition | 14 |
| I.6.2 Le fonctionnement de la virtualisation | 14 |
| I.7 Avantage du Cloud Computing | 15 |
| I.8 Inconvénients du Cloud Computing | 16 |
| II. Fog Computing | 17 |
| II.1 Définition et concept du Fog Computing | 17 |
| II.2 L'architecture du Fog Computing | 18 |
| II.3 L'évolution dans le Fog Computing | 19 |
| II.4 Les domaines d'utilisation du Fog Computing | 20 |
| II.5 Différence entre le Fog et le Cloud | 21 |
| II.6 Avantage du Fog | 22 |
| III. L'ordonnancement dans le Cloud Computing | 23 |
| III.1 Définition | 23 |
| III.2 Les données d'un problème d'ordonnancement | 24 |
| III.3 Types d'ordonnancement | 25 |
| III.4 La dépendance des tâches d'une application | 26 |
| III.5 Les principaux algorithmes classiques d'ordonnancement | 28 |
| Conclusion | 30 |

# Chapitre II

## *Les métaheuristiques AG et BLA*


| | |
|---|---|
| Introduction | 31 |
| I. Etat de l'art des métaheuristiques d'optimisation | 31 |
| I.1 L'optimisation | 31 |
| I.2 Qu'est-ce qu'un problème d'optimisation ? | 31 |
| I.3 Algorithme d'optimisation | 33 |
| I.3.1 La méthode heuristique | 34 |
| I.3.2 La méthode métaheuristique | 34 |
| 1. Les métaheuristiques à solution unique | 36 |
| 2. Les métaheuristiques à population de solution | 36 |
| 2.1 Algorithme évolutionnaire | 36 |
| II. Concept et généralité sur les algorithmes génétiques (GA) | 39 |
| II.1 Définition des algorithmes génétiques AG | 39 |
| II.2 Terminologie propre aux algorithmes génétiques | 40 |
| II.3 Fonctionnement des algorithmes génétiques | 41 |
| II.3.1 codage des paramètres | 42 |
| II.3.2 généralité de la population initiale | 42 |
| II.3.3 Evaluation des individus | 42 |
| II.3.4 Les opérateurs | 42 |
| II.4 Les paramètres des algorithmes génétiques | 45 |
| II.5 Propriété des algorithmes génétiques | 46 |
| II.6 Convergence (Critère d'arrêt) | 46 |
| III. Algorithme de vie des abeilles (BLA) | 47 |
| II.1 Les abeilles en nature | 48 |
| II.2 Recherche de nourriture chez les abeilles | 50 |
| II.2.1 Exploration des sources de nourriture | 50 |
| II.2.2 Méthode de communication chez les abeilles | 50 |
| Conclusion | 53 |

# Chapitre III

## *Application de BLA pour le problème d'ordonnancement*


| | |
|---|---|
| Introduction | 54 |
| I. Formulation du Problème | 54 |
| I.1 La fonction objective d'évaluation | 55 |
| I.2 minimisation du makespane | 56 |
| II. Formulation de la solution | 56 |
| II.1 Initialisation de l'algorithme BLA | 57 |
| II.2 Critère d'arrêt de l'algorithme BLA | 58 |
| II.3 Opérateur d'optimisation de BLA | 58 |
| II.3.1 Croisement | 59 |
| II.3.2 Mutation | 60 |
| III. Approche de recherche locale gourmande | 61 |
| III.1 Description de l'algorithme ABC | 61 |
| Conclusion | 66 |


# Chapitre IV

## *Réalisation et Tests*


| | |
|---|---|
| Introduction | 67 |
| I. Définition de base | 67 |
| I.1 Data center (DC) | 67 |
| I.2 La machine virtuelle (VM) | 68 |
| I.3 Cloudlet (Task) | 69 |
| I.4 Host | 69 |
| I.5 Broker | 69 |
| II. Exemple de simulation de BLA | 70 |
| III. Objectif à optimiser | 77 |
| III.1 La fonction fitness utilisé | 77 |
| III.2 Evaluation des performances | 79 |
| III.2.1 Etude comparative avec d'autres métaheuristiques | 81 |

| | |
|---|---|
| Conclusion | 85 |
| Conclusion générale | 86 |

# *Liste des figures*

# LISTE DES FIGURES


| | |
|---|---|
| Figure I.1 : Cloud Computing | 4 |
| Figure I.2 : L'évolution dans le Cloud Computing | 5 |
| Figure I.3 : Modèle de déploiement d'un Cloud public | 6 |
| Figure I.4 : Modèle de déploiement d'un Cloud privé | 7 |
| Figure I.5 : Modèle de déploiement d'un Cloud communautaire | 7 |
| Figure I.6 : Modèle de déploiement d'un Cloud hybride | 8 |
| Figure I.7 : les différentes couches des services du Cloud Computing | 9 |
| Figure I.8 : IaaS du Cloud Computing | 10 |
| Figure I.9 : PaaS du Cloud Computing | 11 |
| Figure I.10: SaaS du Cloud Computing | 12 |
| Figure I.11 : Les contraintes de l'infrastructure du Cloud | 12 |
| Figure I.12 : La virtualisation | 14 |
| Figure I.13 : L'architecture du Fog Computing | 19 |
| Figure I.14 : Nombre d'appareils connectés dans le monde entier de 2015 à 2025 (en milliards) | 21 |
| Figure I.15 : Structure d'un DAG | 27 |
| Figure II.1 : Classification des métaheuristiques | 35 |
| Figure II.2 : Différentes classes des métaheuristiques | 37 |
| Figure II.3 : Notions de base de l'algorithme évolutionnaire | 38 |
| Figure II.4 : Structure de l'algorithme génétique | *41* |
| Figure II.5 : croisement à 1-point | *44* |
| Figure II.6 : Croisement 2-points | *44* |
| Figure II.7 : Mutation d'un gène | *44* |
| Figure II.8 : pseudo-code pour un algorithme d'abeille simple | 48 |
| Figure II.9 : la danse des ouvrières | 51 |
| Figure II.10 : l'indice de la direction | 52 |
| Figure III.1 : la représentation des individus de BLA | 57 |
| Figure III.2 : La sélection de 3 individus | 58 |
| Figure III.3 : Le croisement de 2 parents | 60 |
| Figure III.4 : La mutation d'une Progéniture P | 61 |
| Figure III.5 : Pseudo-code de la partie recherche de nourriture | 64 |
| Figure III.6 : Pseudo-code de la partie recherche de nourriture (suit) | 65 |
| Figure IV.1 : Data center de Facebook | 68 |
| Figure IV.2 : Exemple du classement de 3 individus | 71 |
| Figure IV.3 : Exemple du classement de 3 ouvrières | 72 |

| | |
|---|---|
| Figure IV.4 : les points de croisement des deux parents | 73 |
| Figure IV.5 : les progénitures résultant après le croisement | 74 |
| Figure IV.6 : Exemple de calcul du voisinage | 75 |
| Figure IV.7 : la nouvelle source de voisinage | 76 |
| Figure IV.8 : Le résultat de la simulation | 76 |
| Figure IV.9 : Courbe représentative des valeurs de la fonction fitness avec différentes valeurs de pondération données par BLA | 79 |
| Figure IV.10 : Courbe représentative de Makespan donné par BLA en fonction de nombre de tâches avec deux choix de paramètres de BLA | 80 |
| Figure IV.11 : Courbe représentative du flow time donnée par BLA en fonction de nombre de tâches avec deux choix de paramètres de BLA. | 81 |
| Figure IV.12 Histogramme représentatif des résultats la fitness avec les facteurs de pondérations W1 = 0 et W2 = 1 selon les algorithmes GA et BLA | 83 |
| Figure IV.13 Histogramme représentatif des résultats de la fitness avec les facteurs de pondérations W1 = 1 et W2 = 0 selon les algorithmes GA et BLA. | 84 |

# *Liste des tableaux*

| | |
|---|---|
| Tab I.1 : La différence entre le Fog et le Cloud | 22 |
| TabIV.1 étude comparative de la fonction fitness avec différentes valeurs de pondération | 78 |
| TabIV.2 les résultats de la fitness avec les facteurs de pondérations<br>W1 = 0,5 et W2 = 0,5. | 82 |
| TabIV.3 les résultats de la fitness avec les facteurs de pondérations W1 = 0 et W2 = 1. | 82 |
| TabIV.4 les résultats de la fitness avec les facteurs de pondérations W1 = 1 et 0. | 83 |

# *Introduction Générale*

# *Introduction générale :*

L'informatique a connu ces dernières années une importante évolution et les technologies sont de plus en plus matures et sophistiquées. Les serveurs de calculs et de stockage voient leur rapport qualité, prix en constante augmentation. La même observation peut être appliquée aux technologies réseaux et à la bande passante offerte qui tend à devenir illimitée.

La demande croissante pour de nouveaux services informatique plus économiques a permis l'émergence d'une nouvelle architecture est connue sous le nom de « Cloud Computing ».
L'informatique dans les nuages est un paradigme distribué à grande échelle permettant d'exploiter pleinement les ressources informatiques disponibles dans lequel ces ressources sont abstraites, virtualisées, dynamiques, évolutives, hautement disponibles et configurable. Les services sont offerts sur demande à des clients externes via une connexion internet haute débit. cependant l'exploitation de telles infrastructures (Cloud) n'est pas toujours facile vu les caractéristique des ressources qui les constituent. On cite le problème de gestion de données et le problème d'allocation de ressources et l'ordonnancement des tâches. Nous nous basons principalement dans notre mémoire à ce dernier concept.

L'ordonnancement est un aspect fondamental dans le cadre du Cloud Computing, son objectif est d'assigner les tâches aux ressources disponibles de manière à optimiser certains critères comme par exemple minimiser le temps de calcul de l'ensemble des tâches, car l'intérêt de l'utilisateur est d'avoir un programme qui se termine au plus tôt possible.

Dans ce travail, nous proposons une étude sur l'ordonnancement de tâches dans le Cloud computing. Notre étude se focalise sur l'utilisation de la métaheuristique bio-inspiré de Bees Life Algorithme BLA (algorithme inspiré des abeilles), visant à répondre au problème d'ordonnancement de tâches.

Pour cela notre mémoire est subdivisé en quatre chapitres qui sont comme suit :

- Chapitre I intitulé : « ***Généralités sur le Cloud et le Fog Computing »*** présente les concepts du Cloud et du Fog Computing, et les notions fondamentales qui les constituent, ensuite nous aborderons la notion de L'ordonnancement et ses différents algorithmes classiques.

- Chapitre II intitulé : « ***Les métaheuristique AG et BLA »*** celui-ci comprend le principe de la métaheuristique d'optimisation et de l'algorithme génétique ainsi que l'algorithme BLA.
- Chapitre III intitulé : ***« Application de BLA pour le problème d'ordonnancement »*** dans ce chapitre nous avons détaillé la métaheuristique BLA ses caractéristique et son fonctionnement.
- Chapitre IV intitulé : ***« Réalisation et Tests »*** qui portera sur l'implémentation et la réalisation de l'application. Nous ferons une étude comparative avec l'algorithme génétique AG ainsi l'ensemble des tests et résultats de la méthode implémentée.

Pour en conclure nous avons une dernière partie Annexe qui détaille et explique l'ensemble des outils et moyens que nous avons mets au profit pour la réalisation de ce travail de recherche.

# *Chapitre I :*

# *Généralités sur le Cloud et le FogComputing*

## *Introduction*

Les technologies de l'information et de la communication évoluent et révolutionnent nos modes de vie et de travail. Le Cloud Computing est un paradigme qui regroupe un ensemble de techniques et de pratiques consistant à accéder, en libre-service, à des ressources généralement positionnées dans des Datacenters, à travers une infrastructure réseau (Internet).

*L'ordonnancement*, est un aspect important pour le fonctionnement du Cloud, son objectif est d'assigner les tâches aux ressources disponibles de manière à minimiser le temps de leur calcul.

Dans ce chapitre, nous allons aborder, le CLOUD COMPUTING, ses types, ses avantages et ses inconvénients, ensuite on passera au FOG COMPUTING pour étudier son architecture et déterminer les éléments qui caractérisent les deux concepts (le Cloud et le Fog Computing) pour en finir avec les problèmes d'ordonnancement des tâches dans le Cloud Computing.

# *I. Le Cloud Computing*

## *I.1 Définitions et concepts du Cloud Computing*

Le Cloud Computing est l'un des tout derniers paradigmes de la technologie dans le monde des IT, c'est un réseau de ressources informatique accessibles publiquement ou en privé sur Internet ou Intranet. ***[I.1]***

- ***D'après NIST[1](National Institute of Standards and Technology)***

« Le Cloud Computing est un modèle qui permet un accès réseau à la demande et pratique à un pool partagé de ressources informatiques configurables (telles que réseaux, serveurs, stockage, applications et services) qui peuvent être provisionnées rapidement et distribuées avec un minimum de gestion ou d'interaction avec le fournisseur de services. »

[1]Le NIST : est un organisme fédéral non réglementé relevant du Département des États-Unis Fondé en 1901. La mission du NIST est de promouvoir l'innovation et la compétitivité industrielle au États-Unis en faisant progresser la science de la mesure, les normes et technologie de manière à renforcer la sécurité économique et à améliorer notre qualité de vie.

- **D'après Numergy[2]**

« Le Cloud Computing 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 culture-informatique***

« Le Cloud Computing est un modèle qui permet un accès omniprésent, pratique et à la demande à un réseau partagé et à un ensemble de ressources informatiques configurables (comme par exemple : des réseaux, des serveurs, du stockage, des applications et des services) qui peuvent être provisionnées et libérées avec un minimum d'administration. »

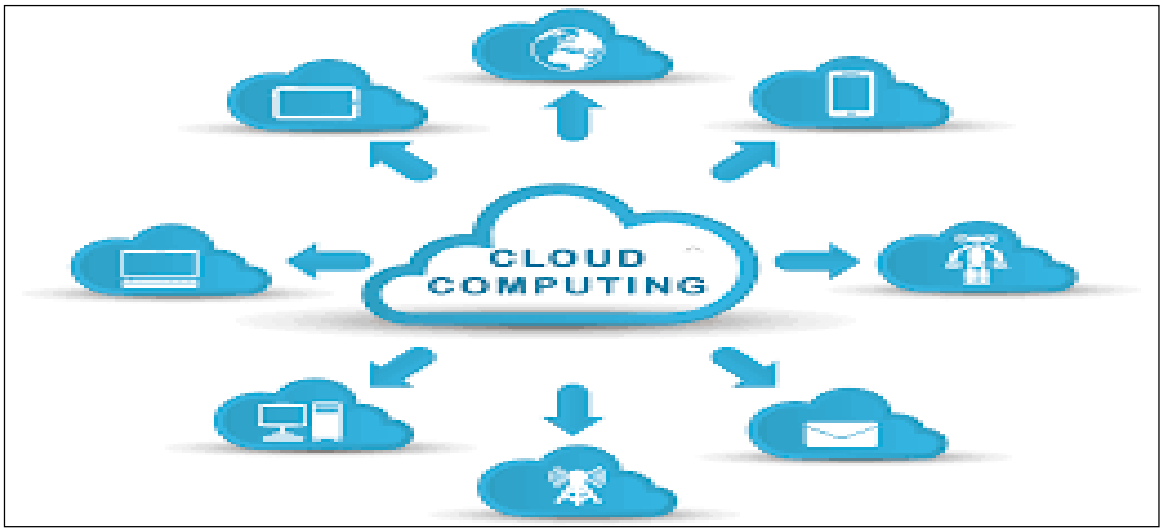



**Figure I.1 : Cloud Computing *[1.1]***

[2]Numergy était une société française de Cloud Computing fondée en 2012 par SFR et Bull dans le cadre de projet Andromède de Cloud souverain française. Placés sous procédure de sauvegarde en octobre 2015 elle est intégralement rachetée par SFR Group en janvier 2016

## *I.2 Historique du Cloud Computing*

Il est communément admis que le concept de Cloud Computing a été initié par le géant Amazon en 2002. Le cybermarchand avait alors investi dans un parc informatique afin de pallier les surcharges des serveurs dédiés au commerce en ligne constatées durant les fêtes de fin d'année. A ce moment-là, Internet comptait moins de 600 millions d'utilisateurs mais la fréquentation de la toile et les achats en ligne étaient en pleine augmentation. En dépit de cette augmentation, les ressources informatiques d'Amazon restaient peu utilisées une fois que les fêtes de fin d'année étaient passées. Ce dernier a alors eu l'idée de louer ses capacités informatiques le reste de l'année à des clients pour qu'ils stockent les données et qu'ils utilisent les serveurs. Ces services étaient accessibles via Internet et avec une adaptation en temps réel de la capacité de traitement, le tout facturé à la consommation.

Cependant, ce n'est qu'en 2006 qu'Amazon comprit qu'un nouveau mode de consommation de l'informatique et d'internet faisait son apparition.

Bien avant la naissance du terme de Cloud Computing, utilisé par les informaticiens pour qualifier l'immense nébuleuse du net, des services de Cloud étaient déjà utilisés comme le webmail2, le stockage de données en ligne (photos, vidéos,…) ou encore le partage d'informations sur les réseaux sociaux. ***[I.2]***

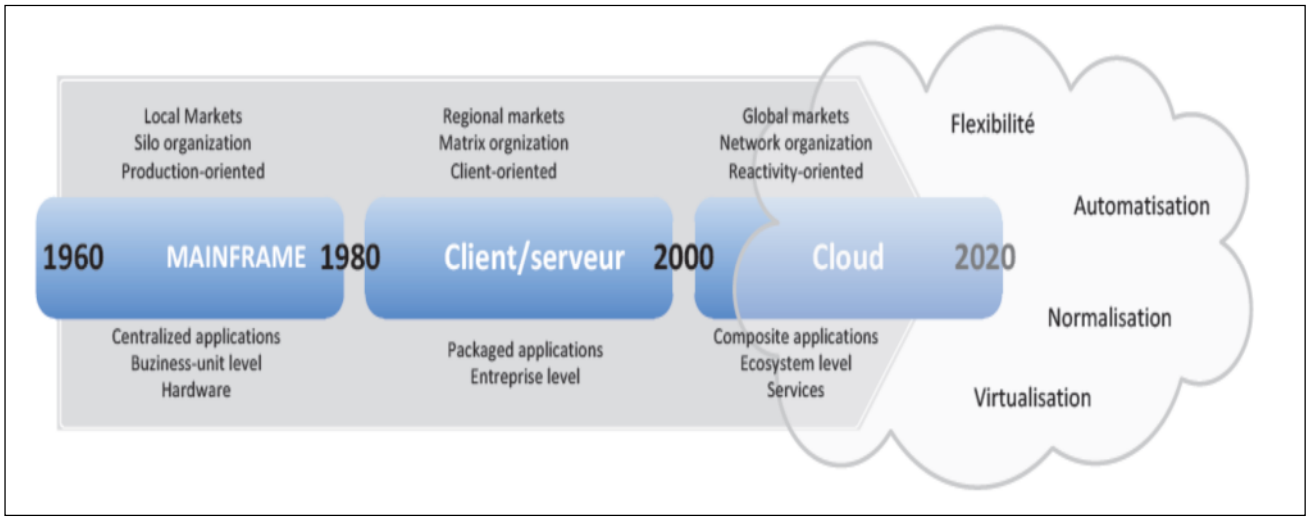



**Figure I.2 : L'évolution dans le Cloud Computing *[1.2]***

## I.3 Les modèles de déploiement du Cloud Computing

Les entreprises, ainsi que les organisations scientifiques, ont des besoins différents en ce qui concerne le type, la disponibilité ou la sécurité des services qu'ils emploient. Pour y remédier, plusieurs modèles de déploiements du Cloud ont été proposés, chacun d'eux présentent des points spécifiques. ***[I.3]***

### I.3.1 Le Cloud public

Le Cloud public offre ses services à n'importe quel client qui accède à l'Internet. En règle générale, les services sont gérés par des fournisseurs de l'extérieur du site, qui contrôlent également le réseau et les paramètres de sécurité, tandis que l'utilisateur peut obtenir à la demande des ressources gratuits ou facturés en se basant sur l'utilisation par payement.

Le Cloud public permet aux systèmes et aux services d'être facilement accessibles par tous les utilisateurs, il est peut-être moins sécurisé en raison de son ouverture, exemple : le courrier électronique. ***[I.3]***

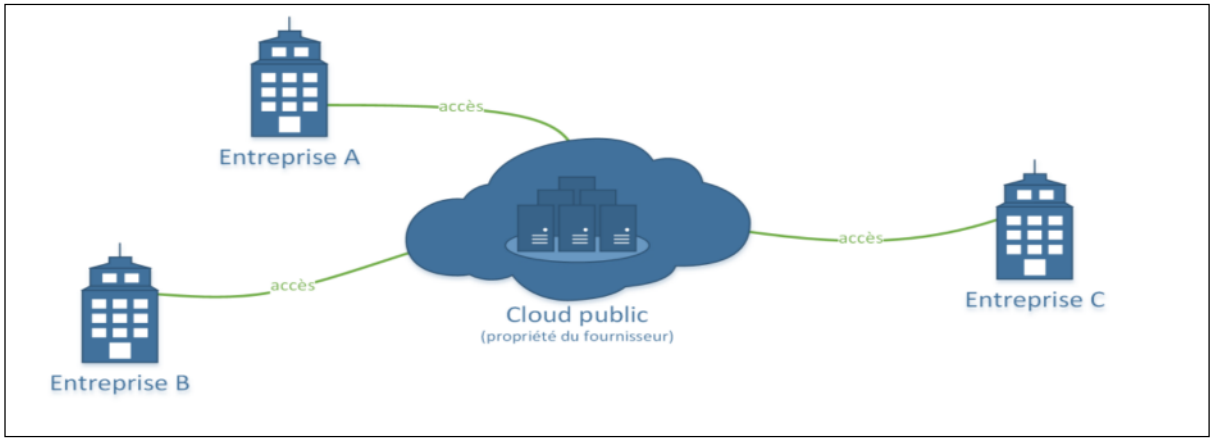



**Figure I.3 : Modèle de déploiement d'un Cloud public *[1.3]***

### I.3.2 Le Cloud privé

Les réseaux, serveurs, et infrastructures de stockage associés aux Cloud privés sont dédiés à une seule entreprise et ne sont pas partagés avec d'autres. Le Cloud est entièrement contrôlé par l'entreprise elle-même, L'avantage du Cloud privé est qu'il offre le plus haut degré de sécurité et de fiabilité. Ce haut degré de contrôle et de transparence permet au propriétaire d'un Cloud privé de se conformer plus facilement à des normes, politiques de

sécurités ou conformités réglementaires qui peuvent être requises dans certains domaines. Avoir un accès à un Cloud privé implique d'avoir un accès complet à l'infrastructure physique et aussi à la construction et à la gestion dans le Cloud. ***[I.3]***

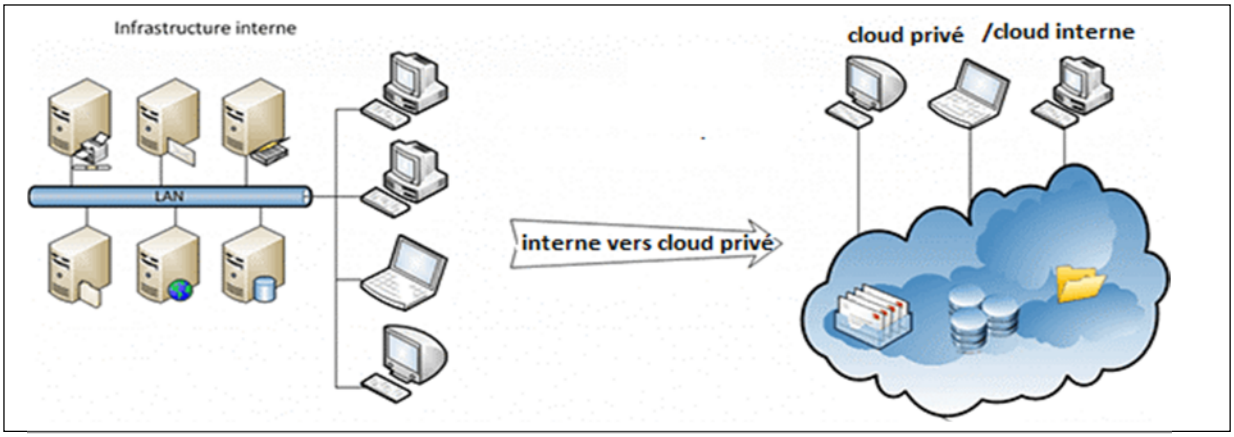



**Figure I.4 : Modèle de déploiement d'un Cloud privé *[1.4]***

### *I.3.3 Le Cloud Communautaire*

L'architecture est dédiée à une communauté professionnelle spécifique, pour permettre de travailler de manière collaborative sur un même projet.

L'infrastructure commune est spécifiquement conçue pour répondre aux exigences de la communauté, à titre d'exemple, des organismes gouvernementaux, des hôpitaux ou des entreprises de télécommunication qui auraient des contraintes de réseau, de sécurité, de stockage, de calcul ou d'automatisations similaires pourraient trouver des intérêts communs à déployer collectivement un Cloud communautaire. ***[I.3]***

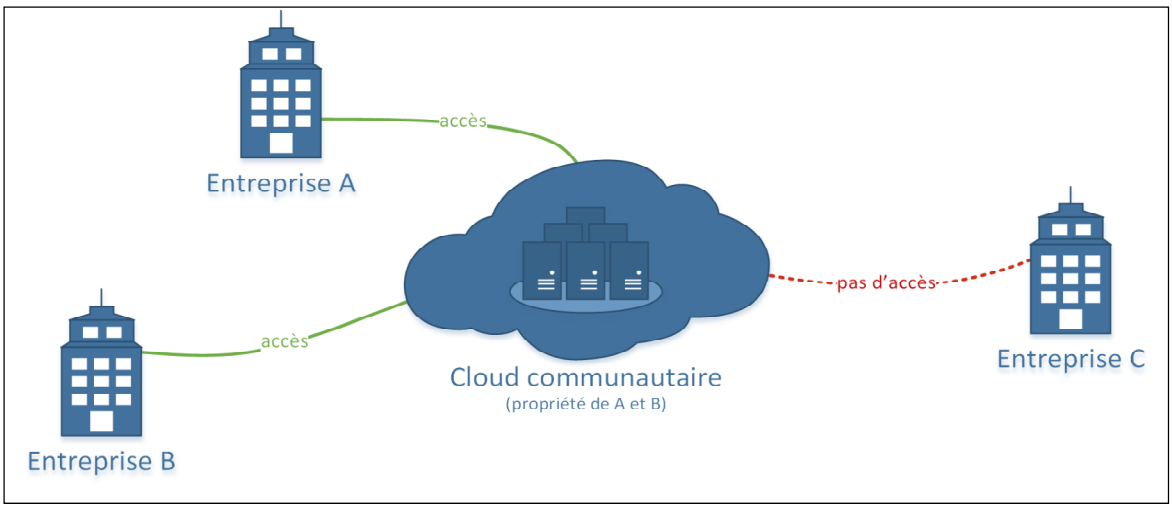



**Figure I.5 : Modèle de déploiement d'un Cloud communautaire *[1.3]***

### *I.3.4 Le Cloud hybride*

Comme nous venons de le voir avec le Cloud public, certains problèmes peuvent se poser, liés à la sécurité de l'information. Il est alors possible de «mélanger» les deux approches du Cloud, privé et public, en débouchant sur une plate-forme hybride. Ce terme n'évoque pas vraiment un Cloud en tant que tel, puisqu'il s'agit de faire cohabiter et communiquer un Cloud privé et un Cloud public. Dans le public, on déportera les éléments non sensibles et dans le privé, on gardera les données, applications sensibles liées au business de l'entreprise. ***[I.3]***

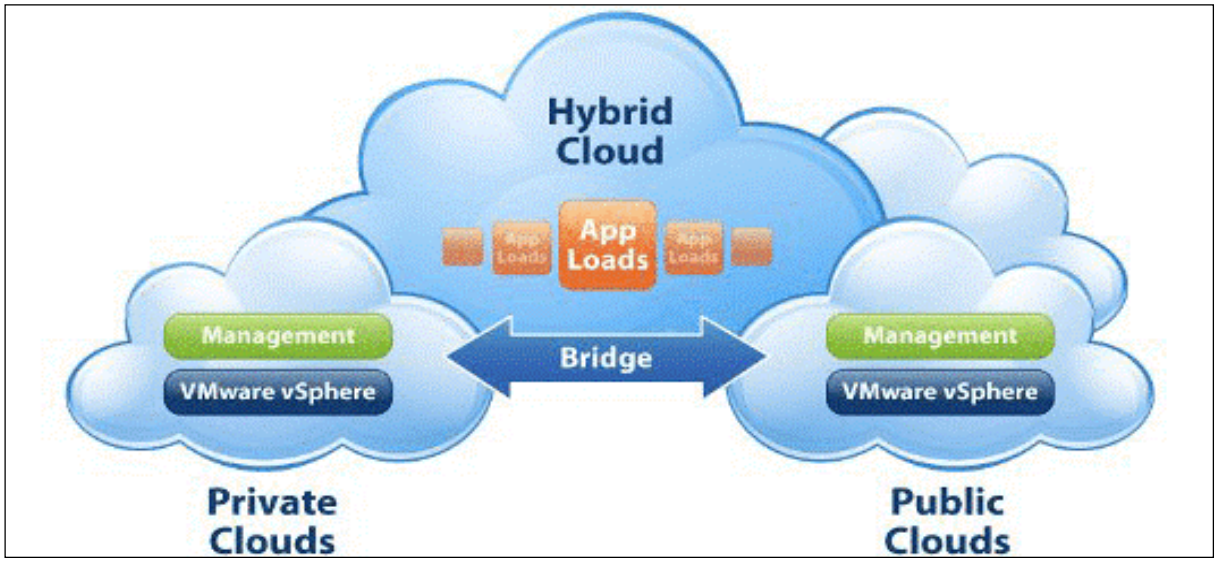



**Figure I.6 : Modèle de déploiement d'un Cloud hybride.** ***[1.4]***

## *I.4 Les modèles de services de Cloud Computing*

Le Cloud Computing offre un modèle d'accès à la demande à des ressources informatiques partagées et configurables (des réseaux, des serveurs, du stockage, des applications et des services) qui peuvent être rapidement provisionnées et mises à disposition avec un effort minimum de gestion et d'interactions avec le fournisseur. La référence aux Cloud évoque une dématérialisation de l'infrastructure informatique. L'idée est de proposer une capacité de traitement informatique sans se soucier des ressources physiques nécessaires à son maintien, ni à sa localisation, ni à sa puissance, …etc.

L'offre de Cloud Computing se décline en trois modèles distincts comme il le montre la figure ci-dessus : ***[I.3]***

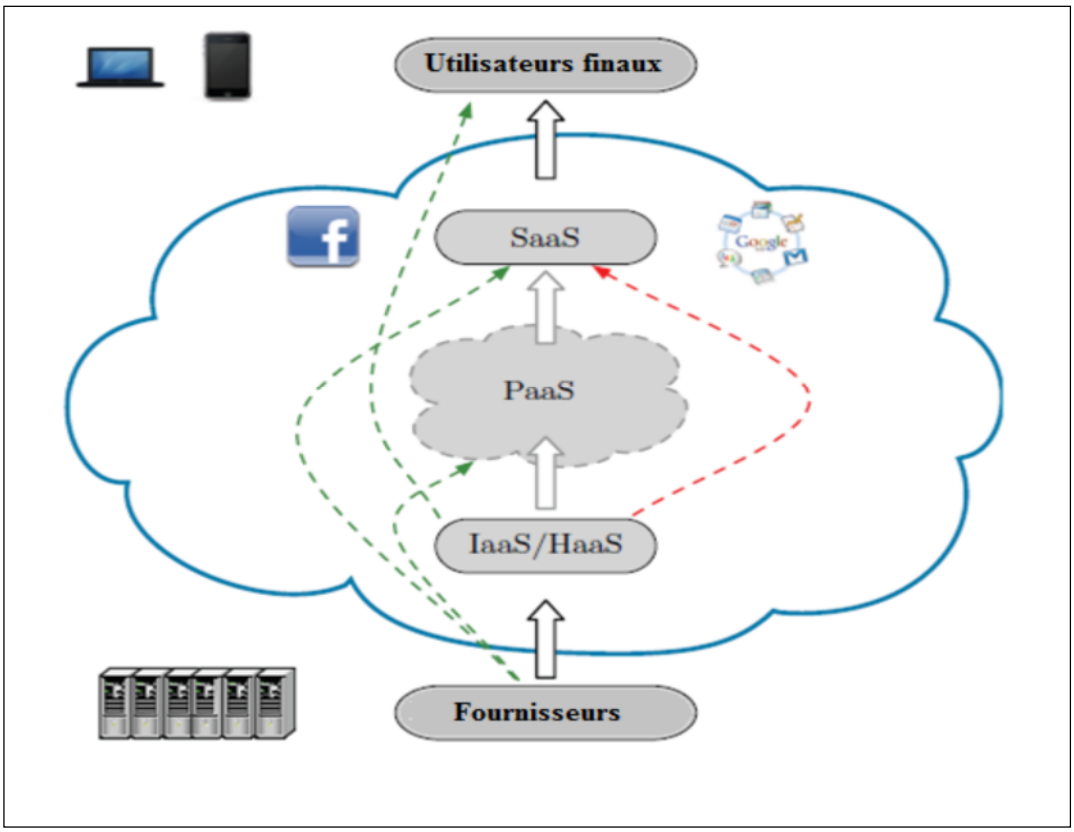



**Figure I.7 : les différentes couches des services du Cloud Computing.** ***[I.3]***

### I.4.1 *L'IaaS (Infrastructure as a Service)*

C'est le niveau le plus bas de la pile de Cloud Computing, est principalement utilisé par les entreprises voulant créer des solutions informatiques sur mesure à des coûts avantageux et facilement extensibles.

Il fournisse aux utilisateurs et aux entreprises un accès à la demande de ressources virtualisés, ainsi l'entreprise pourra évoluer et se développer à son rythme sans avoir besoin d'acheter, d'installer et d'intégrer elles-mêmes le matériel.

L'IaaS est une solution adaptée et adaptable, elle permet aux entreprises d'évoluer à moindre coût car le client ne paie que pour les ressources qu'il a utilisées, c'est l'acte à la demande, il n'y a donc pas de gaspillage, de ressource inutile, ni de perte d'argent.

En outre, en adoptant cette solution technique, les entreprises peuvent faire des économies substantielles sur l'achat, l'installation, la maintenance et les coûts d'utilisation (serveurs, réseau). ***[I.3]***

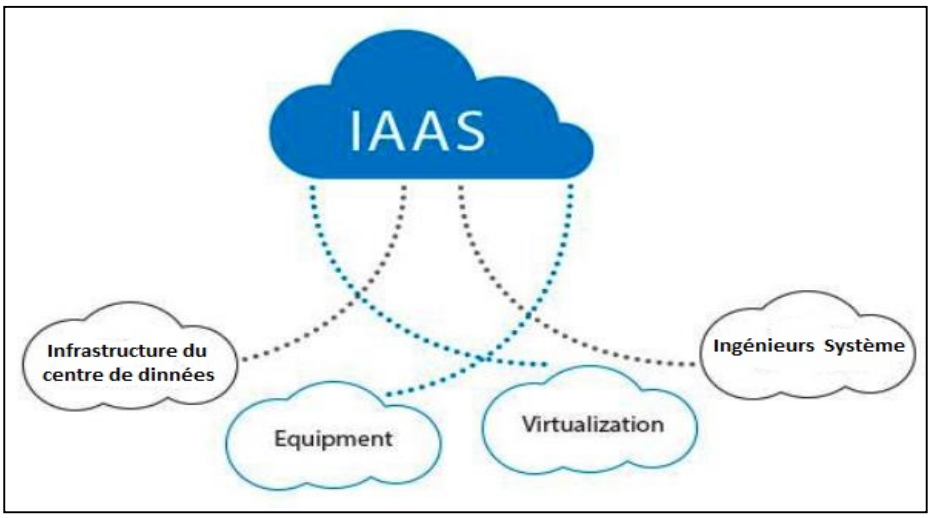



**Figure I.8 : IaaS du Cloud Computing *[1.5]***

### *I.4.2 PaaS (Platform as a Service)*

La Plateforme en tant que Service fournit une plateforme et un environnement informatique qui permet aux développeurs de mettre en place différents services et applications sur Internet. Les services PaaS sont hébergés pour la plupart dans le Cloud et les utilisateurs y accèdent directement, par un navigateur web.

Il fournit un Système d'Exploitation (OS) ainsi que tous les logiciels serveurs associés : serveur d'application, bases de données, d'intégration, les runtimes. Il s'agit donc d'une plateforme logicielle complète, sur laquelle il est possible de développer ou migrer ses propres applications.

L'infrastructure et les applications sont gérées directement par le fournisseur. Le client peut accéder à des services de support s'il le souhaite. Les services proposés aux clients sont constamment mis à jour, avec l'ajout de nouvelles fonctionnalités et l'amélioration des fonctionnalités déjà présentes.

Grâce à un système de services managés, les fournisseurs d'offres PaaS peuvent accompagner et assister les clients développeurs depuis la conception de leur idée originale jusqu'à la création de leurs applications, en passant par les phases de tests et de déploiement.

De même que l'IaaS, les services PaaS sont facturés sur la base de la consommation du client, le client ne paie que ce qu'il utilise cela permet donc des économies d'échelle non négligeable. En effet la mutualisation donc le partage des infrastructures physiques entre différents utilisateurs donne la possibilité d'abaisser sensiblement les coûts de revient du Cloud**.** ***[I.3]***

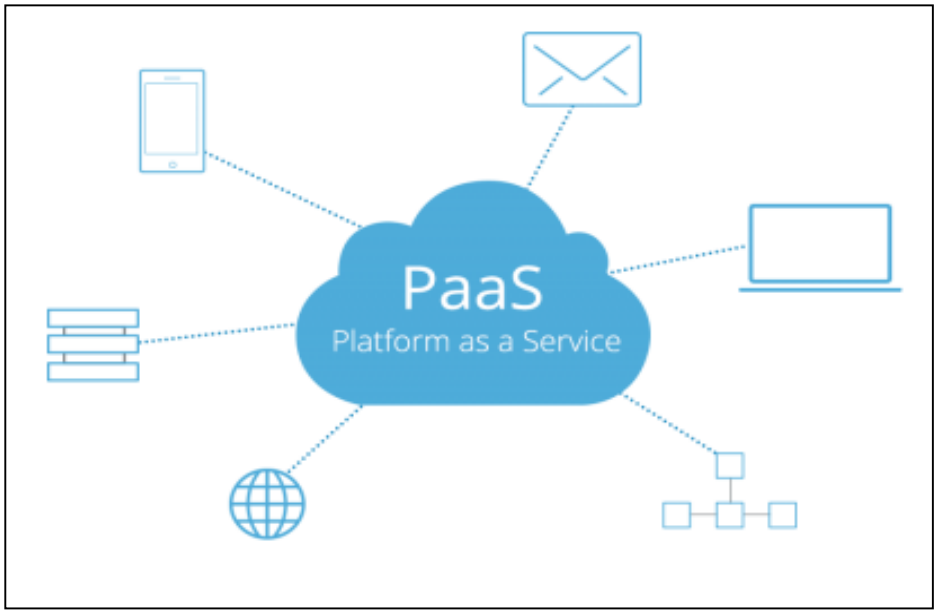



**Figure I.9 : PaaS du Cloud Computing.** ***[1.6]***

### *I.4.3 SaaS (Software as a Service)*

Logiciel en tant que Service, est un modèle de distribution de logiciel à travers le Cloud, il donne un accès aux services via n'importe quel appareil (téléphone, tablette, ordinateur) disposant d'une connexion à Internet. Il n'est pas rare qu'une entreprise utilise des applications pour remplir de nombreuses fonctions, incluant la comptabilité, la facturation, le suivi des ventes, la planification, le monitoring des performances et les communications notamment la messagerie web et la messagerie instantanée.

C'est un modèle de déploiement d'application dans lequel un fournisseur loue une application clé en main à ses clients en tant que service à la demande au lieu de leur facturer des licences. De cette façon, l'utilisateur final n'a plus besoin d'installer tous les logiciels existants sur sa machine de travail. Cela réduit également la maintenance en supprimant le besoin de mettre à jour les applications.

Ce type de modèle transforme les budgets logiciels en dépenses variables et non plus fixes et il n'est plus nécessaire d'acquérir une version du logiciel pour chaque personne au sein de l'entreprise. ***[I.4]***

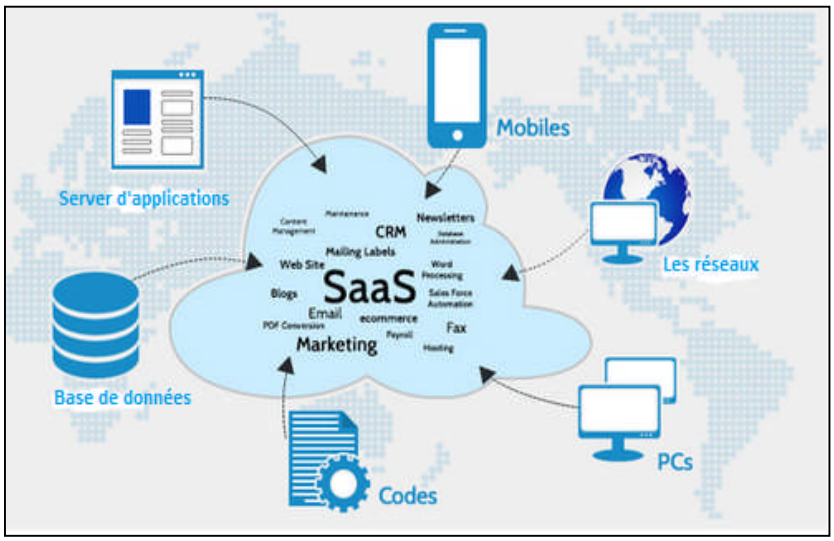



**Figure I.10 : SaaS du Cloud Computing.** ***[I.4]***

## *I.5 Contraintes de l'infrastructure*

Les contraintes fondamentales auxquelles le Cloud devrait faire face sont montrées dans le graphique suivant : ***[I.5]***

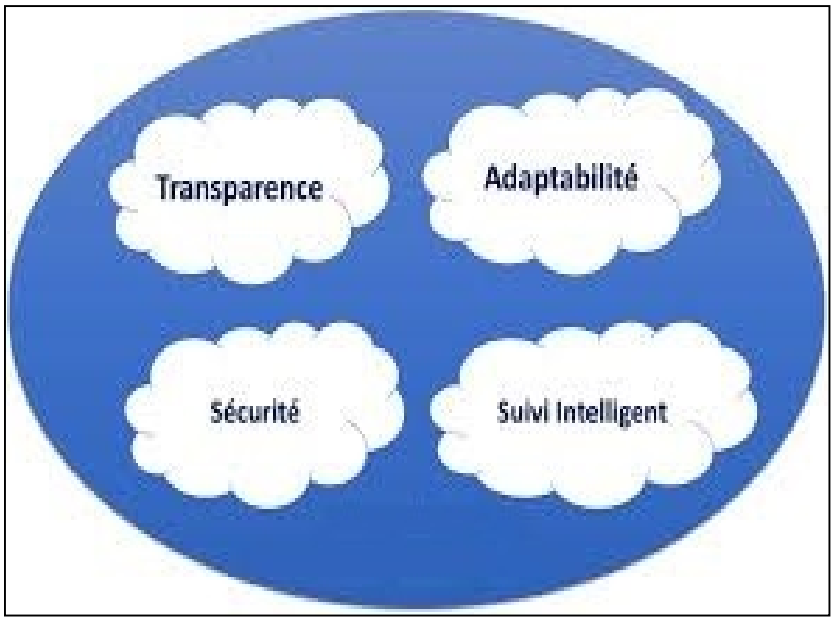



**Figure I.11 : Les contraintes de l'infrastructure du Cloud.** ***[I.5]***

- ***Transparence***

La virtualisation est la clé pour partager les ressources dans l'environnement du Cloud. Mais il n'est pas possible de satisfaire la demande avec une seule ressource ou un seul serveur. Par conséquent, il devrait y avoir une certaine transparence dans les ressources, dans l'équilibre des charges et d'application, de manière à ce que l'on puisse les ajuster à la demande.

- ***Adaptabilité***

Adapter une solution pour distribuer l'application n'est pas aussi simple que d'adapter une application car cela demande une configuration supérieure ou même une rectification du réseau. Ainsi, une solution pour distribuer l'application doit être évolutive, ce qui demande aux infrastructures virtuelles d'être telles que les ressources puisse être approvisionnées et désapprovisionnées facilement.

- ***Suivi intelligent***

Pour assurer la transparence et l'adaptabilité, la distribution de la solution devra bénéficier d'un suivi intelligent.

- ***Sécurité***

Le **centre de données**[3] du Cloud devrait être solidement construit. De même que le point de contrôle, un point d'entrée dans le centre doit aussi être sécurisé.

## *I.6 La virtualisation dans le Cloud Computing*

Une machine virtuelle (VM) est une instance de la machine physique (PM) et elle donne l'illusion aux utilisateurs d'avoir un accès direct à une PM. Les VMs sont utilisées pour permettre le partage de ressources d'un matériel très coûteux. Chaque VM est une copie entièrement protégée et isolée pour réduire des coûts du matériel et d'améliorer la productivité globale en permettant à plusieurs utilisateurs de travailler simultanément sur la même PM. Cela permet à la virtualisation d'augmenter l'utilisation de la machine. ***[I.6]***

[3] **Un centre de données :** est un lieu (et un service) regroupant des équipements constituant du système d'information d'une ou plusieurs entreprises (ordinateur centraux, serveurs, baies de stockage, équipements de réseaux et télécommunications,…etc.).Il fournit des services informatiques en environnement contrôlé (climatisation) et sécurité (système anti-incendie, contre le vol et l'intrusion,…etc.), avec une alimentation d'urgence et redondante.

### *I.6.1 Définition*

On définit la virtualisation comme un ensemble de techniques matérielles et/ou logiciels visant à faire fonctionner sur une seule machine physique plusieurs systèmes d'exploitation. Elle permet d'exploiter toute la capacité d'une machine physique en la répartissant entre de nombreux utilisateurs ou environnements différents. ***[I.7][I.8]***

### *I.6.2 Le fonctionnement de la virtualisation*

Un système d'exploitation principal (appelé « système hôte ») est installé sur un serveur physique unique. Ce système sert à accueillir d'autres systèmes d'exploitation. Un logiciel de virtualisation appelé « hyperviseur » ou bien « VMM (Virtual Machine Monitor) se situant entre les deux couches matériel et le système d'exploitation (**Figure [1.7]**) est installé sur le système d'exploitation principal. Il permet la création d'environnement clos et indépendants sur lesquels seront installés d'autres systèmes d'exploitations « systèmes invités ». Ces environnements sont des « machines virtuelles ». Un système invité est installé dans une VM qui fonctionne indépendamment des autres systèmes invités dans d'autres machines virtuelles. Chaque machine virtuelle dispose d'un accès aux ressources du serveur physique (CPU, mémoire, espace disque,…). Le VMM prend le contrôle de multiplexage des ressources, il est utilisé aussi pour la gestion d'énergie du système. ***[I.8]***

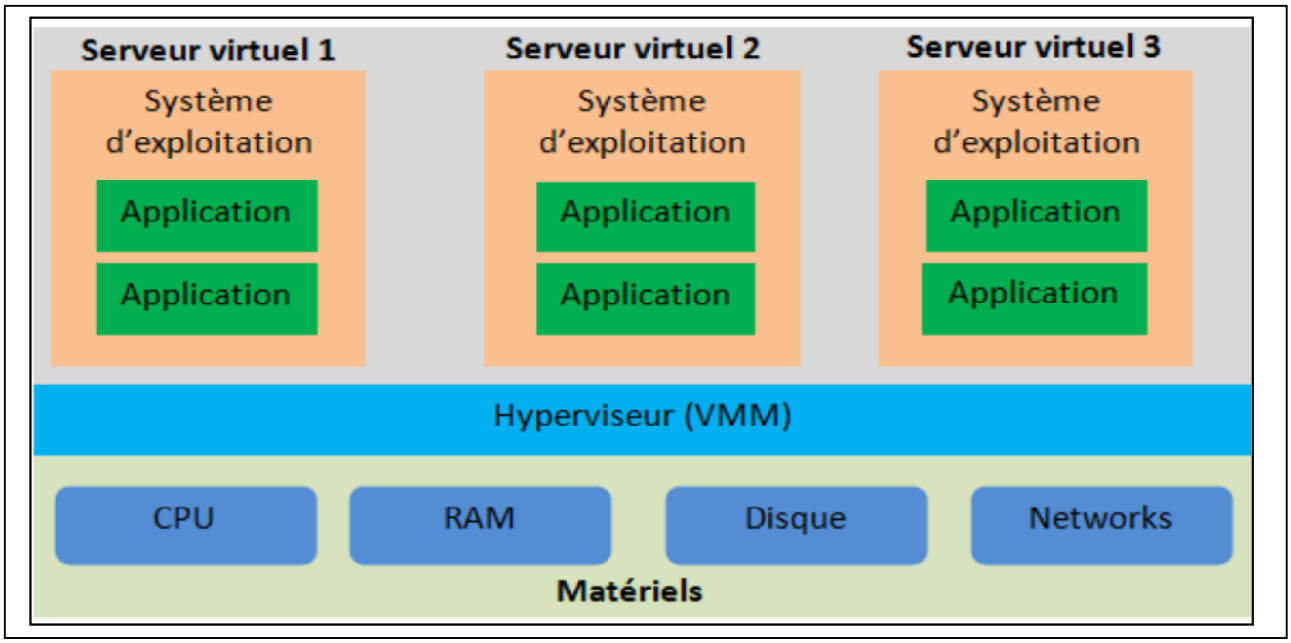



**Figure I.12 : La virtualisation *[I.8]***

## *I.7 Avantages du Cloud Computing*

Le Cloud offre de multiples avantages aux entreprises et aux utilisateurs finaux. Parmi les plus importants : ***[I.9] [I.10]***

- ***La réduction des coûts***

La mutualisation des ressources informatiques et la facturation à l'usage rend le Cloud économiquement attrayant.

- ***L'accessibilité***

Les services du Cloud sont accessible à tout moment, sur tous les supports, via une connexion internet.

- ***Élasticité***

Il est possible, grâce au Cloud, de disposer de plus de ressources instantanément pour soutenir une forte demande. Inversement, par exemple en cas de diminution du nombre d'utilisateurs d'une plate-forme, il est possible de libérer des ressources. Autrement dit, il est possible avec le Cloud d'acquérir de nouvelles ressources en fonction de l'évolution des besoins, et le client peut soit augmenter ou diminuer par exemple le nombre de machines et/ou leurs puissances. Généralement, cette tâche est automatisée et c'est l'opérateur de service qui ajuste les ressources en fonction des besoins. Du point de vue du client (utilisateur), les ressources du Cloud doivent paraître illimitées, accessibles en n'importe quelle quantité et de n'importe quel endroit. [***I.10]***

- ***La disponibilité du service***

Le Cloud permet de garantir les accès et la disponibilité des services. Le fournisseur de services du Cloud s'engage contractuellement sur une interruption minimum des serveurs à travers des SLA[4] (service Level Agreements).

[4]SLA : (service Level Agreements) est un document qui quantifie le niveau de service minimal pour une prestation qu'un fournisseur s'engage à délivrer à son client, Il peut être soit une partie d'un contrat informatique, soit un annexe à un contrat informatique, soit un annexe à des conditions générales. Ou plus simplement « convention de service » est donc un contrat (ou la partie du contrat de service) dans lequel on formalise la qualité du service en question.

- ***Une grande extensibilité***

Les ressources du Cloud sont disponibles à la demande depuis les vastes pools de ressources du Cloud public, afin que les applications qu'elles supportent puissent répondre de manière flexible aux fluctuations d'activistes.

- ***Fiabilité***

La grande quantité de serveurs et de réseaux, dit configurations redondantes signifient qu'en cas de panne de l'un des composants, le service Cloud peut continuer à fonctionner sur les autres composants. Dans les cas où les Clouds tirent leurs ressources de nombreux data centres, un data centre entier peut tomber en panne et les services Cloud n'en seront pas affectés. En d'autres mots, il n'existe aucun point unique de défaillance susceptible de rendre un service de Cloud public vulnérable.

- ***Indépendance de l'emplacement***

La disponibilité des services de Cloud public par connexion Internet assure que les services sont disponibles ou que se trouve le client. Cela représente un nombre immense d'opportunités pour les entreprises, notamment l'accès distant a l'infrastructure IT (en cas d'urgence etc.) ou la collaboration avec des documents en ligne depuis de multiples emplacements.

## *I.8 Inconvénients du Cloud Computing*

Le Cloud offre de nombreux avantages, mais aussi des inconvénients, ils se déclinent selon les points suivants : ***[I.11]***

- ***La connexion internet***

Le Cloud utilisant de manière intensive le transfert de données, il faut avoir une connexion très performante. Si l'utilisateur ou l'entreprise n'a pas de connexion internet, ou une connexion insuffisante, il ne pourra pas accéder à sa plateforme de travail. L'idée dans ce cas est de mettre le travail sur une application locale qui synchronise ensuite les données avec le serveur dès que l'utilisateur a à nouveau accès au réseau. Le problème de la sécurité des données en local se pose donc à nouveau.

Si la connexion ne dispose pas d'un débit garanti, une coupure peut survenir, privant l'entreprise de tous les accès au Cloud.

- ***Consommation d'électricité***

Le plus gros problème du Cloud Computing c'est la consommation élevé de l'électricité et l'énergie dépensée par les Datacenter.

- ***Localisation des données***

La dématérialisation des données sur différents sites physique de stockage peut conduire à un éclatement des données et leur répartition sur différentes zones géographiques.

- ***Sécurité et privacité***

C'est la plus grande préoccupation concernant le Cloud Computing. Étant donné que la gestion des données et la gestion de l'infrastructure dans le Cloud sont fournies par des tiers, il est toujours possible de transférer les informations sensibles à ces fournisseurs. Bien que les fournisseurs du Cloud Computing garantissent des comptes protégés par un mot de passe plus sécurisé, tout signe de violation de sécurité entraînerait la perte des clients et des entreprises.

# *II. Le Fog Computing*

## *II.1 Définitions et concepts du Fog Computing*

La norme IEEE[5] définit le « *fog computing* » comme « une architecture horizontale au niveau du système qui distribue les ressources et les services de calcul, de stockage, de contrôle et de mise en réseau à travers tout le continuum Cloud-objets » ***[I.12]***

- ***D'après Flavio Bonomi[6]***

« Egalement appelé Fog networking ou fogging, le Fog Computing renvoie à une infrastructure matérielle et applicative distribuée, taillée pour stocker et traiter les données issues des objets connectés. Au lieu de centraliser dans le Cloud les informations produites par les capteurs, l'idée à travers cet environnement est de faire appel aux équipements situés à la périphérie du réseau (routeurs, passerelles, commutateurs, appareils mobiles…) pour

[5] **IEEE :** est un comité de l'**IEEE** qui décrit une famille de **normes** relatives aux réseaux locaux (LAN) et métropolitains (MAN) basés sur la transmission de données numériques par des liaisons filaires ou sans fil.

[6]**M. Flavio Bonomi :** est chef de la direction et fondateur de Nebbiolo technologies, M.Bonomi était le fondateur et le directeur de la technologie chez loXWorks,ou il a exercé des fonction de conseil auprès de plusieurs entreprises et startup dans le domaine l'idO avant rejoindre IoXWork il était follow Cisco, Vice-prisident et chef de l'organisation pour l'architecture avancée et la recherche au sein de Cisco Système à San jose.

réaliser les traitements. En créant cette surcouche intermédiaire au plus près de la production des données, l'objectif est à la fin d'optimiser les délais de réponse des applications.

Le Fog Computing est considéré comme une extension locale du Cloud. Par contraste avec le Cloud, ou nuage, qui se forme dans le ciel (traduisant l'idée d'une informatique distante), le terme anglais Fog (brouillard en français) se forme au-dessus du sol. L'expression Fog Computing renvoie ainsi à la notion d'une informatique plus proche du monde physique, des terminaux et de l'IoT ».

## *II.2 L'architecture du Fog Computing*

L'architecture informatique Fog est constituée d'éléments **physiques** et **logiques** sous forme de matériel et de logiciels permettant la mise en œuvre d'un réseau IoT[7] (Internet of Things). Comme le montre la figure 13, il est composé de périphériques IoT, de nœuds de brouillard, de nœuds d'agrégation de brouillard à l'aide de services de données de brouillard, d'un stockage en Cloud distant et d'un serveur Cloud de stockage de données local.

On définit les composants de l'architecture informatique de brouillard comme suite : ***[I.13]***

- **Appareils IoT :** il s'agit d'appareils connectés sur un réseau IoT utilisant diverses technologies filaires et sans fil. Ces appareils produisent régulièrement des données en très grande quantité. Les protocoles IoT utilisés incluent IPv4, IPv6, MQTT, …etc.
- **Nœuds de brouillard** : Tout périphérique doté d'une connectivité informatique, de stockage et réseau est appelé nœud de brouillard. Plusieurs nœuds de brouillard sont répartis sur une plus grande région afin de prendre en charge les périphériques finaux **(appareils IoT)**. Les nœuds de brouillard sont connectés en utilisant différentes topologies. Par exemple au sol d'une usine, au sommet d'un poteau électrique, le long de la voie de chemin de fer, dans des véhicules. Des exemples de nœuds de brouillard sont les commutateurs, les serveurs intégrés, les contrôleurs, les routeurs, etc. Les données hautement sensibles sont traitées sur ces nœuds de brouillard.
- **Nœuds agrégés de brouillard :** chaque nœud de brouillard a son nœud de brouillard Agrégé. Il analyse les données en secondes à minutes. Le stockage de données IoT sur ces nœuds peut durer en heures ou en jours. Sa couverture géographique est plus large. Les services de données de brouillard sont implémentés pour implémenter de

[7] **IoT :** L'**Internet des objets**, ou **IdO** est l'interconnexion entre Internet et des objets, des lieux et des environnements physiques. L'appellation désigne un nombre croissant d'objets connectés à Internet permettant ainsi une communication entre nos biens dits physiques et leurs existences numériques

tels points de nœud agrégés. Ils sont utilisés pour traiter les données sensibles moyennes.

- **Cloud distant :** tous les nœuds de brouillard agrégés sont connectés au Cloud. Les données insensibles au temps où les données moins sensibles sont traitées, analysées et stockées sur le Cloud.
- **Serveur local et Cloud :** L'architecture informatique en brouillard utilise souvent un Serveur Cloud privé pour stocker les données confidentielles de la société. Ce stockage local est également utile pour assurer la sécurité et la confidentialité des données.

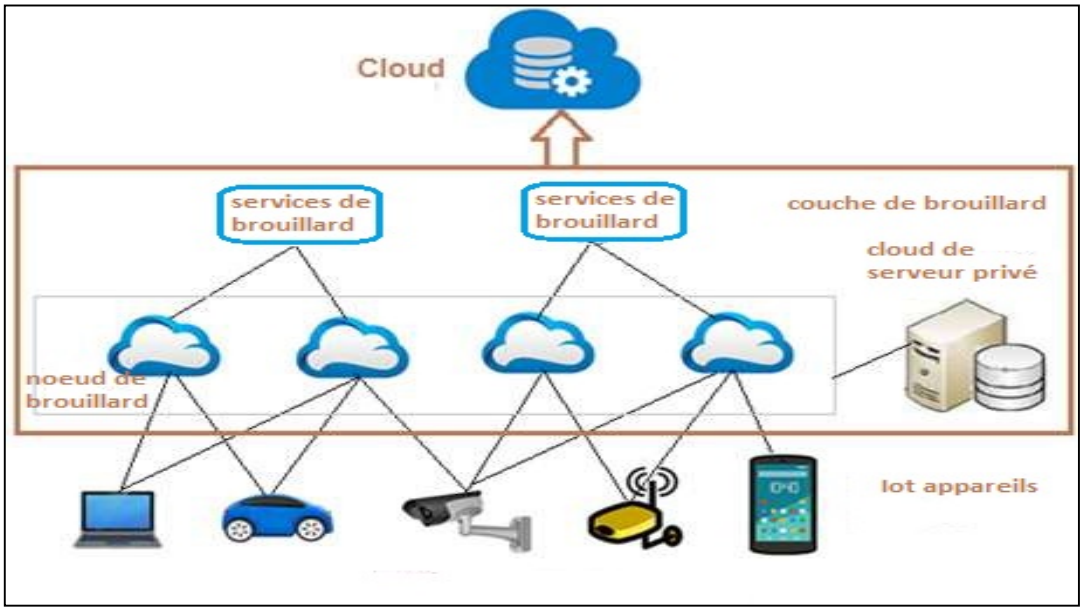



**Figure I.13 : L'architecture du Fog Computing *[I.13]***

## *II.3 L'évolution dans le Fog Computing*

En 2012, la nécessité d'étendre l'informatique en nuage avec l'informatique en brouillard est apparue, afin de faire face à un nombre considérable d'appareils IoT et de volumes de données volumineux pour les applications en temps réel à faible latence. ***[I.14][I.15]***

Le 19 novembre 2015, Cisco Systems, ARM Holdings, Dell, Intel, Microsoft et l'Université de Princeton ont fondé le consortium OpenFog [8] pour promouvoir les intérêts et le

[8]**Consortium OpenFog :** est un consortium d'entreprises de haute technologie et d'institutions universitaires du monde entier visant à la normalisation et à la promotion de l'informatique dans le brouillard dans divers domaines et capacités. Il a été fondé par Cisco Systems, Intel, Microsoft, l'Université de Princeton, Dell et ARM Holdings en 2015 le consortium OpenFog a fusionné avec l'Industrial Internet Consortium le 31 janvier 2019.

développement dans le domaine de l'informatique dans le brouillard. Le directeur principal de Cisco, Helder Antunes, est devenu le premier président du consortium et le stratège en chef, Intel, Jeff Fedders, son premier président. ***[I.16][I.17]***

L'IOT d'aujourd'hui ne peut être comparé à celui de demain. Dès 2020, l'Internet des objets comportera environ 50 milliards d'appareils, d'après une étude de Cisco. Cela impliquera une croissance considérable du volume de données à stocker, analyser et traiter. ***[1.18]*** L'usine intelligente est loin d'être le seul domaine où le Fog Computing peut aider à alléger cette charge. D'autres projets futurs, tels que les voitures connectées (voitures semi-autonomes ou autonomes) ou la ville en réseau dotée de réseaux de distribution intelligents, nécessitent une analyse des données en temps réel. Cela ne peut pas être mis en œuvre avec le Cloud Computing classique. Par exemple, un véhicule intelligent collecte des données sur l'environnement, les conditions de conduite qui doivent être évaluées sans latence afin que les réactions à des incidents imprévisibles puissent également être prises en temps voulu. Dans un tel scénario, le Fog Computing permet le traitement des données du véhicule aussi bien à bord du véhicule que chez le fournisseur de service. ***[I.19]***

## *II.4 Les domaines d'utilisation du Fog Computing*

- ***Soins de santé et suivi des activités***

Le brouillard informatique pourrait être utile dans les soins de santé, dans lesquels le traitement en temps réel et la réaction aux événements sont essentiels. Un système proposé utilise l'informatique par brouillard pour détecter, prédire et prévenir les chutes par accident vasculaire cérébral ***[I.20]***

- ***Services publics intelligents***

Le brouillard informatique peut être utilisé avec des services publics intelligents dont l'objectif est d'améliorer la production, la distribution et la facturation de l'énergie. Dans de tels environnements, les périphériques de périphérie peuvent communiquer aux périphériques mobiles des utilisateurs des informations détaillées sur la consommation d'énergie (par exemple, des lectures horaires et quotidiennes plutôt que mensuelles) par rapport aux services publics traditionnels. Ces appareils périphériques peuvent également calculer le coût de la consommation d'énergie tout au long de la journée et suggérer quelle source d'énergie est la

plus économique à un moment donné ou à quel moment les appareils domestiques doivent être allumés pour minimiser l'utilisation des services publics. ***[I.20]***

## *II.5 Différence entre le FOG et le CLOUD*

D'ici 2020, il y aura 50 milliards d'appareils IoT dans le monde et, en 2025, ce nombre dépassera 75 milliards d'objets connectés, selon Statista. Tous ces appareils produiront d'énormes quantités de données qui devront être traitées rapidement et de manière durable. Pour répondre à la demande croissante de solutions IoT, le brouillard informatique entre en action au même titre que le Cloud Computing. Le brouillard est encore meilleur à certaines choses. ***[I.21]***

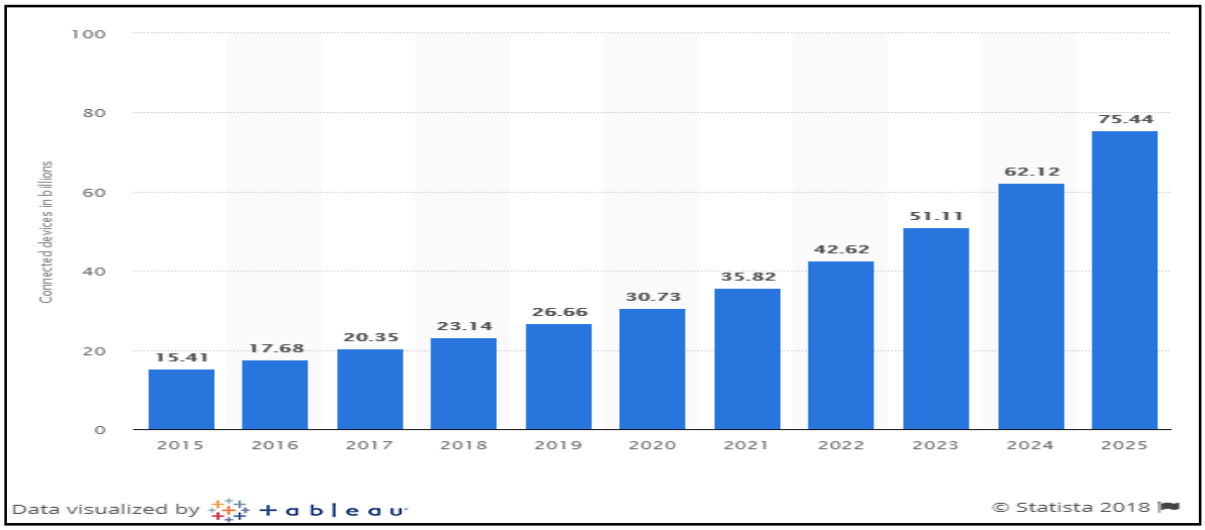



**Figure I.14** : **Nombre d'appareils connectés dans le monde entier de 2015 à 2025 (En milliards)** ***[I.21]***

Le brouillard informatique est un médiateur entre le matériel et les serveurs distants. Il détermine quelles informations doivent être envoyées au serveur et lesquelles peuvent être traitées localement. En ce sens, le brouillard est une passerelle intelligente qui décharge les nuages et permet un stockage, un traitement et une analyse des données plus efficaces. Il convient de noter que la mise en réseau en brouillard n'est pas une architecture distincte et qu'elle ne remplace pas l'informatique en nuage mais la complète, en se rapprochant le plus possible de la source d'informations. ***[I.21]***

Le tableau ci-dessous permet de mieux comprendre la différence entre le brouillard et le nuage, en résumant leurs caractéristiques les plus importantes.

| Caractéristiques | **Nuage** | **Brouillard** |
|---|---|---|
| **Utilisateur Cible** | Utilisateur internet Générale | Utilisateurs mobiles |
| **Communication avec des appareils** | D'une certaine distance<br><br>(plusieurs sauts) | Directement du bord<br><br>(un seul saut) |
| **Type de service** | Informations globales collectées dans le monde entier | Service d'information localise limité à un emplacement de déploiement spécifique |
| **Connectivité** | Internet | Divers protocoles et standards |
| **Nombre de ressource** | Elevée | Inferieure |
| **Grand fournisseur de services** | Amazon, Microsoft, IBM | Cisco IOx |
| **Environnement de travail** | A l'intérieur avec un espace énorme et une ventilation | Environnement de travail à l'extérieur (c'est-à-dire les pistes de rue) ou intérieur (la maison, les centres commerciaux, les restaurants) |

**Tab I.1 : La différence entre le Fog et le Cloud *[I.21] [I.20]***

## *II.6 Avantages du FOG:*

Les infrastructures du Fog permettent une : ***[I.19]***

- réduction significative de la circulation des données à travers le réseau. Donc réduction de la congestion, du coût et de la latence. Elimination des goulets d'étranglement résultant de systèmes informatiques centralisés. Amélioration de la sécurité des données chiffrées qui restent plus proche de l'utilisateur final en réduisant leur exposition. Meilleure évolutivité découlant de système virtualisé.

- ***Latence réduite***

Le Fog Computing raccourcit les voies de communication et accélère ainsi les processus automatisés d'analyse et de décision.

Le Fog Computing s'avère donc plus performant que le Cloud Computing pour faire face aux nouvelles demandes induites par l'apparition de l'IoT. Mais il ne remplacera jamais totalement le Cloud Computing, qui restera privilégié pour le traitement de gros volumes de données, un scénario très répandu. En fait, l'industrie s'oriente vers une complémentarité des deux systèmes en fonction de besoins spécifiques.

- ***Sécurité des données***

Quand on a recours au Fog Computing, le prétraitement des données de l'appareil à lieu sur le réseau local. Cela permet une implémentation dans laquelle les données sensibles restent dans l'entreprise ou peuvent être cryptées ou rendues anonymes avant d'être téléchargées sur le Cloud.

- ***Disponibilité hors ligne***

Les appareils IdO sont également disponibles hors ligne quand ils sont intégrés dans une architecture de Fog Computing. ***[I.19]***

## *III. L'ordonnancement dans le Cloud Computing*

### *III.1 Définition*

Ordonnancer un ensemble de tâches, c'est programmer leur exécution en leur allouant les ressources requises et en fixant leurs dates de début et de fin.

Tout problème d'ordonnancement se définit donc par un ensemble de tâches, à exécuter sur un ensemble de ressources. Il s'agit d'affecter, à chaque tâche, une date d'exécution et un sous-ensemble de ressources de façon à satisfaire à la fois des contraintes temporelles (par exemple, respecter un ordre partiel sur l'ensemble des tâches) et des contraintes d'accès aux ressources (par exemple, exécuter sur une ressource au plus une tâche à chaque instant). Enfin, l'ordonnancement des tâches doit généralement répondre à un critère d'optimisation (par exemple, minimiser la durée totale de l'ordonnancement). ***[I.22]***

## *III.2 Les données d'un problème d'ordonnancement*

Dans un problème d'ordonnancement quatre notions fondamentales interviennent. Ce sont les tâches, les ressources, les contraintes et les objectifs. Ainsi, étant donnés un ensemble de tâches et un ensemble de ressources, il s'agit d'ordonnancer les tâches et affecter les ressources de façon à optimiser un ou plusieurs objectifs (un objectif correspondant à un critère de performance), en respectant un ensemble de contraintes : ***[I.23] [I.24] [I.25]***

- ***Les tâches***

Une tâche est une entité élémentaire de travail localisée dans le temps par une date de début et une date de fin d'exécution et qui consomme des ressources avec des quantités déterminées. Un coût (ou poids) est attribué à une tâche pour estimer sa priorité, son degré d'urgence.

On distingue deux types de tâches : tâches dépendantes et tâches indépendants

- ***Les ressources***

Les ressources dans le Cloud Computing ne sont que la source pour répondre aux besoins informatiques requis. Ce sont deux types, ressources matérielles et ressources logicielles. Les serveurs et les centres de données sont des ressources matérielles. Les logiciels fonctionnant sur le côté du client et du serveur, les informations et les applications sont des ressources logicielles.

- ***Les contraintes***

Les contraintes représentent les limitations imposées par l'environnement, les ressources ou les applications. Suivant la disponibilité des ressources et suivant l'évolution temporelle, deux types de contraintes peuvent être distingués : contraintes de ressources et contrainte des applications.

### *1. Les contraintes de ressources*

Les contraintes d'utilisation de ressources qui expriment la nature et la quantité des moyens utilisés par les tâches, ainsi que les caractéristiques d'utilisation de ces moyens.

Les contraintes de disponibilité des ressources qui précisent la nature et la quantité des moyens disponibles au cours du temps.

### 2. *Les contraintes des applications*

Exemple : contrainte temporelle, contraintes liés à l'énergie ou au débit, contraintes liés aux coûts (Budget).

- Les contraintes temporelles représentent des restrictions sur les valeurs que peuvent prendre certaines variables temporelles d'ordonnancement. Ces contraintes peuvent être :
  - Des contraintes de dates butoirs, certaines tâches doivent être achevées avant une date préalablement fixée.
  - Des contraintes de dates au plus tôt, liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.

- ***Les objectifs***

Les critères d'évaluation sont les indicateurs de performance sur lesquels se base le choix d'un ordonnancement satisfaisant. Les critères à optimiser consistent à minimiser ou maximiser une fonction objective[9]. Cette fonction objective est généralement liée aux temps total d'exécution, à l'énergie consommée, aux ressources ou bien aux coûts monétaire nécessaire.

## *III.3 Types d'ordonnancement*

Les problèmes d'ordonnancement peuvent être classés en deux grandes catégories : ***[I.23] [I.26]***

- Les problèmes d'ordonnancement dynamique (en ligne)

Pour lesquels la date d'arrivée des jobs n'est pas connue à l'avance.

- Les problèmes d'ordonnancement statique (hors ligne)

Pour lesquels les dates d'arrivée des jobs (généralement ils sont tous prêts à t=0 et toutes leurs caractéristiques sont connues avant l'ordonnancement.

[9] Fonction objective : Le terme **fonction objectif** pour désigner une **fonction** qui sert de critère pour déterminer la meilleure solution à un problème d'optimisation. Concrètement, elle associe une valeur à une instance d'un problème d'optimisation.

## *III.4 La dépendance des tâches d'une application*

Les dépendances des tâches permettent de définir le moment où une tâche peut être lancée sur une machine. La dépendance à un impact crucial pour la conception des algorithmes d'ordonnancement. ***[I.27]***

### ➢ *DAG (Graphe Acyclique Dirigé)*

Un **DAG**, comme les trois mots le décrivent : ***[I.28]***

**Graphe :** un ensemble de nœuds **N** qui sont connectés par des arcs.

**Dirigé :** il n'y a aucun chemin dirigé dans le graphique des tâches qui peut commencer et finir à un nœud particulier **n.**

**Acyclique :** il n'y a pas de cycle dans n'importe quel chemin du graphe.

Soit le graphe G = (N ; A) où *N* = {n1, …, nt} est un ensemble de t nœuds *A*= {ai,j} est un ensemble d'arcs tel que ai,j est un arc partant depuis le nœud ni vers le nœud nj. S'il n'existe pas des cycles dans G alors il est appelé un DAG.

La représentation d'une application par un DAG se fait comme suit [I.15] :

- Les nœuds du DAG représentent les tâches qui composent l'application.
- L'existence d'un arc entre deux nœuds représente l'existence d'une contrainte de précédence entre les deux tâches associées aux nœuds.
- Chaque nœud du DAG est pondéré en fonction du temps d'exécution de chaque Tâche.
- En plus des précédences, les tâches peuvent nécessiter des transferts de données pour prendre en compte les transferts de données dans la représentation, chaque arc est annoté d'une valeur qui correspond à la quantité de données.
- La tâche source dans le DAG est un nœud n'ayant aucun arc entrant, et la tâche puits est un nœud n'ayant aucun arc sortant.

La figure ci-dessus représente la structure d'un DAG :

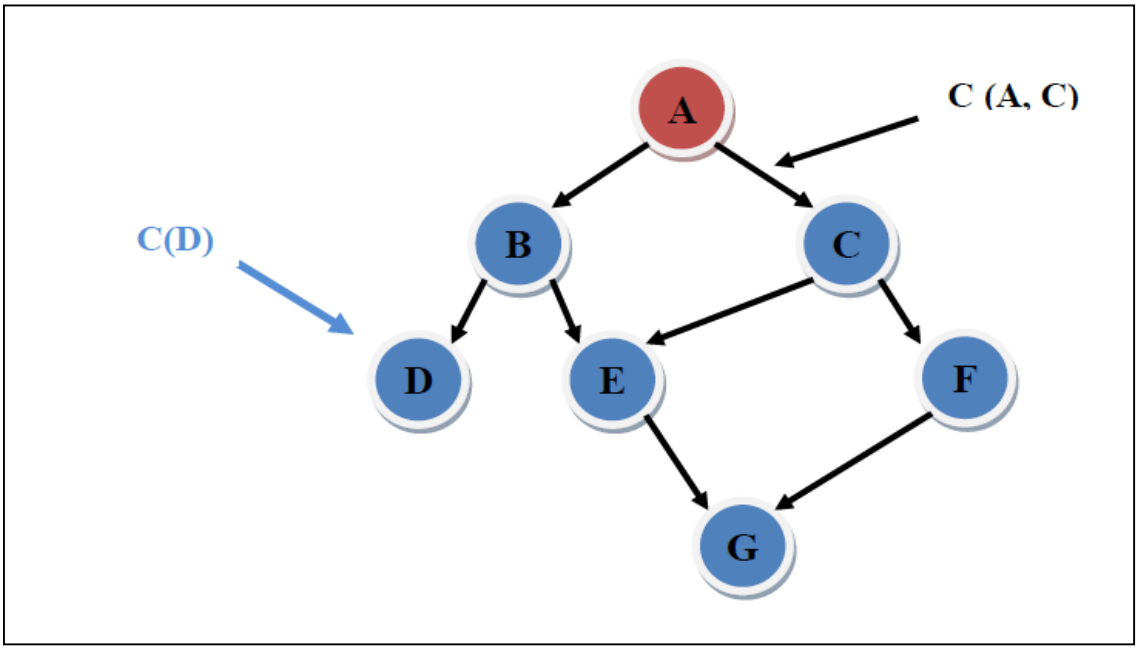



**Figure I.15 : Structure d'un DAG** ***[I.28]***

Chaque nœud a un poids nommé coût de calcul et noté par **C (ni).**

- Les arcs du **DAG** représentent les messages de communication entre un nœud et ses fils pour commencer l'exécution.
- Le poids d'un arc est appelé coût de communication et noté par **C (ni, nj).**

### 1. *Ordonnancement des tâches indépendantes*

Dans ce type d'ordonnancement, toutes les informations sont connues à l'avance comme le temps d'exécution sur les différentes ressources, …etc. Dans ce cas, les tâches s'exécutent en parallèle, indépendamment les unes des autres.

### 2. *Ordonnancement des tâches dépendantes*

Dans ce type d'ordonnancement, le système doit placer des tâches en fonction de la charge de ressources et des dépendances entre les tâches, décrites par un DAG (Directed Acyclic Graph) qui se basent sur l'analyse des dépendances du graphe de tâches entier, afin de compléter les tâches interdépendantes au plus tôt.

## *III.5 Les principaux algorithmes classiques d'ordonnancement*

Nous présentons dans ce qui suit, les principaux algorithmes d'ordonnancement cités dans les littératures : ***[I.27] [I.23]***

- ***Algorithme Min-min***

L'algorithme commence par calculer le temps d'exécution minimale pour toutes les tâches puis la valeur minimale entre ces temps minimum est choisie qui représente le temps minimum d'exécution parmi toutes les taches sur les ressources. Ensuite, en fonction de ce temps minimum, la tâche est ordonnancée sur la machine correspondante. Puis le temps d'exécution pour toutes les autres tâches sont mises à jour sur cette machine en ajoutant le temps d'exécution de la tâche assignée à des temps d'exécution des autres tâches sur cette machine/ressource et la tâche assignée. Ensuite, la même procédure et répétée jusqu'à ce que toutes les tâches soient assignées sur les ressources.

- ***Algorithme Max-min***

L'algorithme Max-min suit le même principe que l'algorithme Min-min à l'exception des propriétés suivantes : après avoir calculé les temps d'exécution minimum, la valeur maximale est sélectionnée, qui est la durée maximale parmi toutes les tâches sur les ressources. Ensuite en fonction de ce temps maximum, la tâche est ordonnancée sur la machine correspondante. Puis le temps d'exécution pour toutes les autres tâches sont mises à jour sur cette machine en ajoutant le temps d'exécution de la tâche assignée à des temps d'exécution des autres tâches sur la machine qui acquise la tâche sélectionnée et la tâche assignée est supprimée de la liste des tâches. La même procédure est répétée jusqu'à ce que toutes les tâches soient assignées sur les ressources.

- ***La méthode du chemin critique (Critical Path Method)***

La méthode du chemin critique (CPM), ou analyse du chemin critique (CPA), est un algorithme permettant de planifier un ensemble d'activités de projet. Il est couramment utilisé conjointement avec la technique d'évaluation et de révision de programme

(PERT). Un chemin critique est déterminé en identifiant le plus long tronçon d'activités dépendantes et en mesurant le temps nécessaire pour les terminer du début à la fin.

Aujourd'hui, il est couramment utilisé dans toutes les formes de projets, y compris :

- la construction
- le développement logiciel
- les projets de recherche
- le développement de produits
- l'ingénierie et la maintenance des installations ***[I.29]***

- ***HEFT*** *(Heterogeneous-Earliest-Finish-Time)*

HEFT est parmi les algorithmes d'ordonnancements de liste les plus répandus. Il détermine un ordonnancement totalement statique d'un DAG sur un environnement hétérogène de manière à minimiser la durée totale d'exécution de l'application (makespan).
Il est caractérisé par :

**Rang d'une tâche:** c'est la hauteur (la hauteur d'une tâche est la longueur du chemin le plus long partant de celle-ci vers le puits) ou la distance maximale de la tâche par rapport au puits du DAG.

**EFT (Earliest Finish Time):** c'est une fonction qui cherche la ressource qui exécute la tâche le plus tôt possible. Lors de l'ordonnancement, la liste de tâches prêtes est triée en fonction de rang des tâches. Plus une tâche à un rang important, plus elle est prioritaire. L'objectif de cet ordre est d'exécuter les tâches les plus éloignées de puits (nœud ayant aucun arc sortant) du DAG en premier pour éviter d'importants temps d'inactivité en fin d'exécution c.à.d. les ressources en attente d'une nouvelle tâche.

## *Conclusion :*

Cloud Computing s'est imposée ces dernières années comme un paradigme majeur d'utilisation des infrastructures informatiques. Celui-ci répond à des besoins et à des demandes croissantes en termes de disponibilité et de flexibilité. Une nouvelle technologie fait son apparition nommé FOG COMPUTING, le brouillard informatique est un paradigme distribué qui fournit des services de type Cloud à la périphérie du réseau. Il permet de transformer le centre de données en une plateforme Cloud.

L'adoption du modèle Cloud Computing, soulève un certains nombres de défis et de contraintes relatifs aux problèmes d'ordonnancement de tâches.

Le mécanisme d'ordonnancement de tâches dans le système du Cloud Computing suscite une attention particulière, c'est un aspect important dans le fonctionnement efficace du Cloud.

Le problème d'ordonnancement devient encore plus difficile lorsque les critères à prendre en considération pour l'optimisation sont multiples.

# *Chapitre II :*

## *Les métaheuristiques AG et BLA*

## Introduction

Les métaheuristiques sont des procédures de haut niveau conçues pour résoudre des problèmes d'optimisation dits difficiles. Ce sont en général des problèmes aux données incomplètes, incertaines, bruitées ou confrontés à une capacité de calcul limitée. Les métaheuristiques ont rencontré le succès dans une large variété de domaines. Cela découle du fait qu'elles peuvent être appliquées à tout problème pouvant être exprimé sous la forme d'un problème d'optimisation de critère(s). Ces méthodes sont, pour la plupart, inspirées de la physique (recuit simulé[1]), de la biologie (algorithmes évolutionnaires) ou de l'éthologie (essaims particulaires, colonies de fourmis). Dans ce qui suit, nous présentons brièvement les principales métaheuristiques d'optimisation.

## I- Etat de l'art des métaheuristiques d'optimisation

### I.1 L'optimisation

L'optimisation est un aspect fondamental de l'ingénierie et de la résolution de problèmes. L'objectif de l'optimisation est de chercher les valeurs d'un ensemble de paramètres pour maximiser ou minimiser les fonctions objectives[2] soumises à certaines contraintes. Un choix des valeurs, pour l'ensemble de paramètres, qui satisfont toutes les contraintes, est appelé une solution faisable. Les solutions faisables avec des valeurs de la fonction objective qui sont meilleures que les valeurs de toutes les autres solutions possibles, sont appelées les solutions optimales. ***[II.1]***

### I.2 Qu'est-ce qu'un problème d'optimisation ?

Un problème d'optimisation peut être formulé sous la forme d'un problème de minimisation ou d'un problème de maximisation. Parfois, nous essayons de minimiser une fonction et, parfois, nous essayons de maximiser une fonction. Ces deux problèmes sont facilement convertis l'un à l'autre : ***[II.2]***

[1]***La méthode de recuit simulé*** s'inspire du processus de recuit physique. Ce processus utilisé en métallurgie pour améliorer la qualité d'un solide cherche un état d'énergie minimale qui correspond à une structure stable du solide.

[2]***Une fonction objective*** : Une fonction objective sert de critère pour déterminer la meilleure solution à un problème d'optimisation. Elle peut être à minimiser ou à maximiser.

$$min_x f(x) \Leftrightarrow max_x[-f(x)]$$
$$max_x f(x) \Leftrightarrow min_x[-f(x)] \quad (1)$$

La fonction f(x) est appelée la fonction objective, Comme le montre l'équation (1), tout algorithme qui est conçu pour minimiser une fonction peut facilement être utilisé pour maximiser une fonction, et tout algorithme qui est conçu pour maximiser une fonction peut être facilement utilisé pour minimiser une fonction.

### *Complexité problématique*

La complexité problématique dépend de la difficulté du problème à résoudre et du nombre d'opérations élémentaires qu'un algorithme peut effectuer pour trouver l'optimum en fonction de la taille du problème.

Les problèmes d'optimisation peuvent être classés en deux classes principales : la classe P et la classe NP. ***[II.26]***

#### ➢ *Les problèmes NP-Complets*

La classe NP-Complet regroupe les problèmes de décision pour lesquels il n'existe pas d'algorithme permettant leur résolution en un temps polynomial. Un problème X est un problème NP-Complet s'il appartient à la classe NP, et si quelque soit le problème qui appartient aussi à la classe NP, on peut le réduire au problème X en un temps polynomial. ***[II.26]***

#### ➢ *Les problèmes NP-Difficiles*

La classe de problèmes NP-Difficiles englobe les problèmes de décision et les problèmes d'optimisation. Les problèmes NP-Difficiles sont aussi difficiles que les problèmes NP-Complets. Si un problème de décision associé à un problème d'optimisation P est NP-Complet alors P est un NP-Difficile. Par conséquent, afin de prouver qu'un problème d'optimisation est NP-Difficile, il suffit de montrer que le problème de décision associé à P est NP-Complet.

- **La classe P (Polynomiale)** : les problèmes appartenant à la classe P sont ceux dont le problème de décision correspondant peut être résolu à l'optimum, à l'aide d'un algorithme en temps polynomial. C'est en quelque sorte la classe des problèmes dits « facile».

- **La classe NP (Non-déterministe Polynomial)** : cette classe a un nom trompeur, NP ne correspond pas à Non Polynomial, mais à Polynomial Non-déterministe ou en anglais « Non-déterministe Polynomial ». Cette classe est une extension de la classe P, elle représente la classe des problèmes de décision pour lesquels un algorithme non-déterministe peut vérifier en temps polynomial la validité d'une solution du problème traité.

  La classe NP contient donc des problèmes plus « difficiles » que la classe P. ***[II.26]***

## *I.3 Algorithme d'optimisation*

Un algorithme d'optimisation permet de trouver la solution optimale, différentes méthodes d'optimisation existent et en sont présentées.
Les méthodes d'optimisation peuvent être réparties en deux catégories :

- **Les méthodes exactes** fournissent systématiquement une solution **(optimale)** au problème traité si une telle solution existe. Dans le cas contraire, ce type de méthode permet d'affirmer qu'il n'existe pas de solution au problème traité.

- **Les méthodes approchées** fournissent une solution approchée au problème traité. Elles sont en général conçues de manière à ce que la solution obtenue puisse être située par rapport à la valeur optimale. De telles méthodes permettent d'obtenir des bornes inférieures ou supérieures de la valeur optimale tel que :

**1- Méthodes Heuristiques.**

**2- Méthodes Méta heuristiques.**

### I.3.1 La méthode heuristique

L'heuristique est une méthode, une technique ou un critère de guidage ou de décision, en général empirique ou obtenu par approximation, permettant de choisir la voie la plus prometteuse de recherche de la solution au problème posé, ou d'éliminer les voies les moins intéressantes, sans garantie sur la validité ou la précision de l'information ainsi fournie.

Entrer dans le domaine des heuristiques, c'est se départir d'emblée les schémas classiques. En effet, alors que la démarche classique mathématique est centrée sur l'objet de l'étude, sur la compréhension de sa structure et de sa logique, la démarche heuristique repousse le problème lui-même au rang d'illustration pour dégager des schémas de pensée plus généraux et donc originaux.

Les heuristiques disposent d'une simplicité et donc d'une rapidité dans leur exécution plus élevée que les Algorithmes classiques. Ces règles s'appliquant à un ensemble particulier la recherche des faits ce voit simplifiée et accélérée (moins de possibilité). D'où une analyse des situations améliorées. Mais une méthode heuristique trop simplifiée ou au contraire trop générale peut conduire à des biais cognitifs, générant des erreurs de décision. ***[II.3] [II.4]***

### I.3.2 La méthode métaheuristique

Les métaheuristiques sont une famille d'algorithmes stochastiques destinés à la résolution des problèmes d'optimisation est une méthode approchée qui permet de trouver, pour un problème NP-difficile, une solution réalisable pas forcément optimale en un temps polynomial. Leurs particularités résident dans le fait que celle-ci sont adaptables à un grand nombre de problèmes sans changement majeur dans leurs algorithmes, d'où le qualificatif '**méta**'.

Leur capacité à résoudre un problème à partir d'un nombre minimale d'informations est contrebalancée par le fait qu'elles n'offre aucune garantie quant à l'optimalité de la meilleure solution trouvée. Seule une approximation de l'optimum global est donnée. Cependant ce constat n'est pas forcément un désavantage, étant donné qu'on préférera toujours une approximation de l'optimum global trouvé rapidement qu'une valeur exacte trouvé dans un temps rédhibitoire.

La majorité des métaheuristiques sont inspirées des systèmes naturels, nous pouvons citer à titre d'exemple : le recuit simulé qui est inspiré d'un processus métallurgique, les algorithmes évolutionnaires et les algorithmes génétiques qui sont inspirés des principes de l'évolution

Darwinienne[3] et de la biologie, la recherche tabou qui s'inspire de la mémoire des êtres humains, les algorithmes basés sur l'intelligence d'essaim comme l'algorithme d'optimisation. par essaim de particules, l'algorithme de colonies de fourmis, l'algorithme de colonies d'abeilles, la recherche coucou et l'algorithme d'optimisation par coucou qui s'inspirent du comportement social de certaines espèces évoluant en groupe.

Nous pourrons classer les métaheuristiques en trois catégories à savoir la recherche locale, les métaheuristiques à population ainsi que les méthodes hybrides. **La figure II.1** présente ces trois catégories.

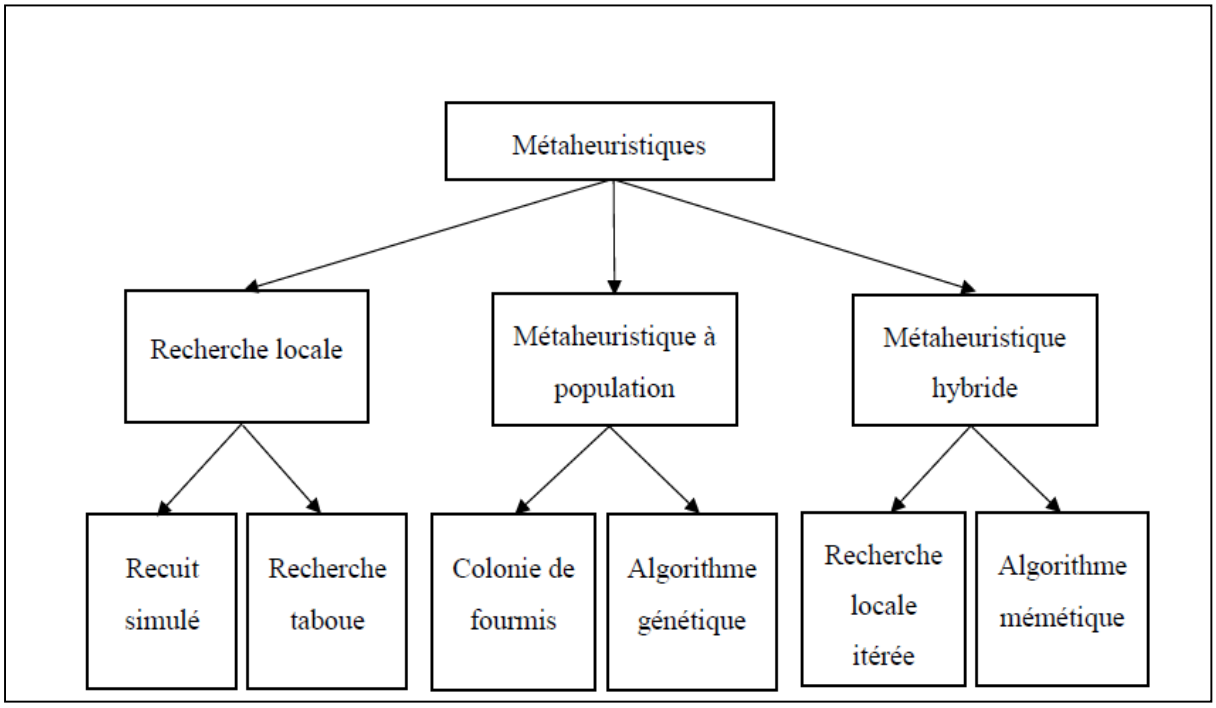



**Figure II.1 :** ***Classification des métaheuristiques [II.23]***

[3]***La théorie de Charles Darwin*** est une partie d'une théorie scientifique : celle de l'évolution des espèces. Selon lui, les espèces animales et végétales ont dû changer pour survivre. Elles ont dû s'adapter aux variations de leur environnement. Seuls ceux qui survivent et se reproduisent ont des descendants : c'est la sélection naturelle. A la fin de cette longue évolution les humains sont apparus parmi les autres animaux.

## 1 Les métaheuristiques à solution unique

Les méthodes à solution unique, plus connues sous le nom de *méthodes de recherche locale*[4]ou encore *méthodes de trajectoire*, sont basées sur l'évolution d'une seule solution dans l'espace de recherche[5]. Typiquement, les méthodes de recherche locale démarrent d'une solution unique, puis à chaque itération, la solution courante est déplacée dans un voisinage local en espérant améliorer la fonction objectif. Les méthodes à solution unique englobent principalement la méthode de descente, la recherche locale itérée, la recherche à voisinage variable, la méthode du recuit simulé et la recherche tabou. Ces méthodes diffèrent essentiellement dans leur manière d'exploiter le voisinage local, i.e. choix des candidats et critères de déplacement.

## 2 Les métaheuristiques à population de solutions

À l'inverse des méthodes de recherche à solution unique, les métaheuristiques à population de solutions sont des méthodes qui font évoluer simultanément un ensemble d'individus (solutions) dans l'espace de recherche, où chacun profite de l'expérience du groupe, de manière directe ou indirecte. Ces méthodes sont principalement inspirées du vivant. On peut distinguer deux catégories de métaheuristiques à population : les algorithmes évolutionnaires inspirés de la théorie de l'évolution de C. Darwin et les algorithmes d'intelligence en essaim inspirés de l'éthologie et de la biologie. ***[II.5]***

### 2.1 Algorithmes évolutionnaires

La naissance des algorithmes évolutionnaires fût marquée dans les années 60 et 70. Cette naissance est due à la recherche qui a abouti aux méthodes approchées. Ces méthodes ont été essayées pour résoudre une catégorie spécifique de problèmes. En effet, il existe une catégorie de problème pour lesquels il est difficile, de trouver une solution en un temps limité. Il est alors utile de trouver une technique permettant la localisation rapide de solutions sous optimales, sachant que l'espace de recherche a une taille et une complexité suffisamment

[4]***La recherche locale*** est une méthode générale utilisée pour résoudre des problèmes d'optimisation, c'est-à-dire des problèmes où l'on cherche la meilleure solution dans un ensemble de solutions candidates.

[5]Dans certaines méthodes d'optimisation, tels que les stratégies d'évolution, ***l'espace de recherche est infini*** : seule la population initiale est confinée dans un espace fini. Mais dans le cas des algorithmes de type génétique, il est généralement nécessaire de définir un espace de recherche fini. Cette limitation de l'espace de recherche n'est généralement pas problématique. En effet, ne serait-ce que pour des raisons technologiques ou informatiques, les intervalles de définition des variables sont en général naturellement limités. De plus, on a souvent une idée des ordres de grandeur des variables du problème.

importantes pour éliminer toute garantie d'optimalité. Pour cela, un système capable de s'auto modifier au cours du temps, tout en améliorant sa performance dans l'accomplissement des tâches auxquelles il est confronté, semble ouvrir la voie à une recherche intéressante. Il s'agit des algorithmes évolutionnaires comptés parmi les méthodes de l'intelligence computationnelle[6]. La Figure II.2 en est une description. ***[II.6]***

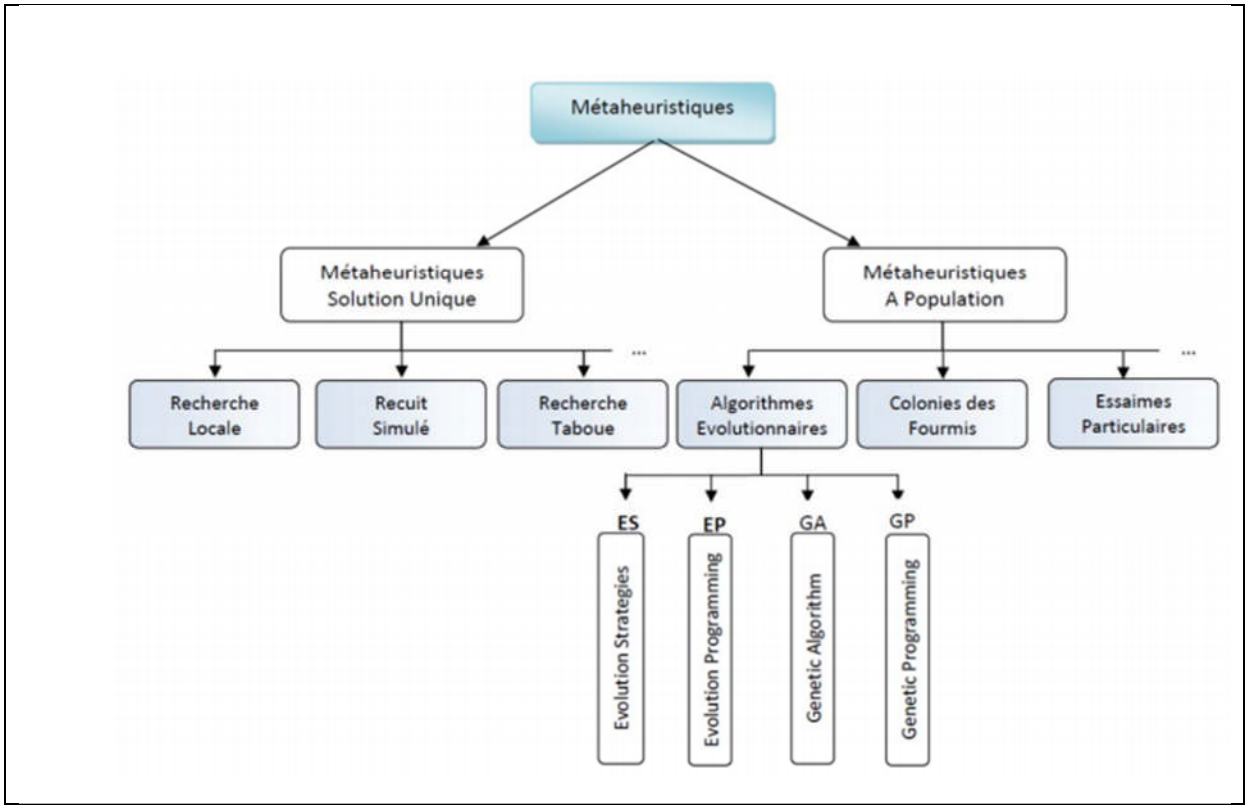



**Figure II.2 : Différentes classes des métaheuristiques.** ***[II.24]***

Les AE sont inspirés du concept de sélection naturelle élaboré par Charles Darwin. Le vocabulaire employé est directement calqué sur celui de la théorie de l'évolution et de la génétique.

[6]L'**intelligence computationnelle** (IC) est un ensemble de calculs inspirés de la nature et des approches pour résoudre les problèmes complexes du monde réel, dont la modélisation mathématique classique n'est pas efficace. Les méthodes utilisées sont proche de la manière du raisonnement de l'humain à savoir la logique floue, les réseaux neuronaux et les algorithmes génétiques.

### 2.1.1 La sélection naturelle

S'il n'existe pas de preuve générale de l'efficacité des algorithmes évolutionnaires il est par contre aisé de constater l'efficacité de la sélection naturelle dans le monde vivant. Si l'on adhère à ce paradigme, il est clair que l'évolution a permis l'émergence d'organismes étonnamment adaptés à leur environnement. Les AE sont conçus par analogie avec ce processus d'évolution biologique et tirent leur puissance des mêmes mécanismes. Dans " The Origin of Species (1859) ", Darwin montre que l'apparition d'espèces distinctes se fait par le biais de la sélection naturelle de variations individuelles. Cette sélection naturelle est fondée sur la lutte pour la vie, due à une population tendant naturellement à s'étendre mais disposant d'un espace et de ressources finis. Il en résulte que les individus les plus adaptés (the fittest) tendent à survivre plus longtemps et à se reproduire plus aisément. Le terme "adapté" se réfère à l'environnement, que l'on peut définir comme étant l'ensemble des conditions externes à un individu, ce qui inclut les autres individus. ***[II.7]***

### 2.1.2 Principes généraux

Les algorithmes évolutionnaires sont basés sur des principes simples. En effet, ils font évoluer une population d'individus, dont l'objectif de trouver la solution optimale, où seulement les mieux adaptés à un environnement donné peuvent survivre et se reproduire. En termes mathématiques, l'environnement est la fonction de performance à optimiser, et elle constitue la seule information nécessaire aux AE, ce qui leur permet de traiter une grande variété de problèmes numériques difficiles à traiter avec des méthodes d'optimisation classiques. Ces algorithmes reposent sur une vision darwinienne relativement simpliste et une optimisation stochastique résumée dans le diagramme de la Figure II.3 ***[II.8] [II.9]***

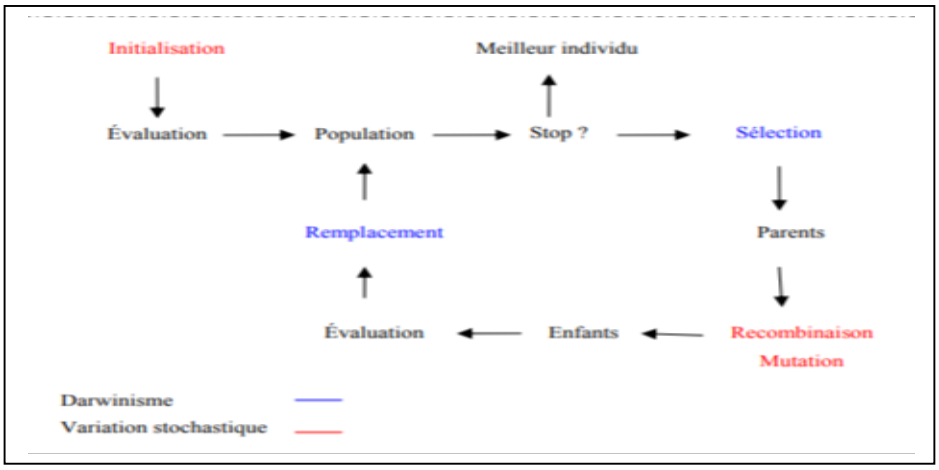



**Figure II.3 : Notions de base de l'algorithme évolutionnaire *[II.23]***

La **Figure II.3** explique alors qu'à chaque population, les individus les plus aptes sont ceux qui survivent pour être croisés et mutés par la suite constituant ainsi une nouvelle population. Ce processus est répété pendant un certain nombre de générations, en espérant que les optima de la fonction apparaissent dans la population. Quel que soit le type de problème à résoudre, les AE opèrent selon les principes suivants : la population est initialisée de façon dépendante du problème à résoudre (l'environnement), puis évolue de génération en génération à l'aide d'opérateurs de sélection, de croisement et de mutation. Dans cette évolution les générations successives des différentes populations préservent une taille constante (mais ce n'est pas une règle). Pour l'environnement, il a pour charge d'évaluer les individus en leur attribuant une performance (fitness). Cette valeur favorisera la sélection des meilleurs individus, en vue, d'améliorer les performances globales de la population. ***[II.8] [II.9]***

## *II- Concept et généralité sur les algorithmes génétiques (AG)*

John Holland[7], ses collègues et ses étudiants ont développé à l'université de Michigan les Algorithmes Génétiques (AGs), métaphores biologiques inspirées des mécanismes de l'évolution darwinienne (sélection naturelle) et de la génétique. Ces métaphores prennent la forme d'algorithmes de recherche appelés "algorithmes génétiques". Ces algorithmes font partie de la classe des algorithmes dits stochastiques. En effet une grande partie de leur fonctionnement est basée sur le hasard. Bien qu'utilisant le hasard, les AGs ne sont pas purement aléatoires. Ils exploitent efficacement l'information obtenue précédemment pour spéculer sur la position de nouveaux points à explorer, avec l'espoir d'améliorer la performance. Les algorithmes génétiques permettent à une population de solutions de converger vers les solutions optimales. Pour ce faire, ils vont utiliser un mécanisme de sélection des individus de la population (les solutions potentielles). Les individus sélectionnés vont être croisés entre eux (exploitation), et certains vont être mutés (exploration). Ces mécanismes d'exploitation et d'exploration vont permettre de converger vers les bonnes solutions en évitant, autant que faire se peut, les optima locaux. ***[II.10] [II.11]***

### *II.1 Définition des Algorithme génétiques AG*

Les GA sont une technique de recherche ou d'optimisation de solution heuristique, motivée à l'origine par le principe darwinien de l'évolution par sélection (génétique). Une AG utilise une version très abstraite des processus évolutifs pour élaborer des solutions à des problèmes

[7] **John Henry Holland** est un scientifique américain, professeur de psychologie et professeur d'ingénierie électrique et de sciences informatiques à l'université du Michigan (Ann Arbor). Il a été un pionnier dans les systèmes complexes et non linéaires. Il est le père des algorithmes génétiques.

donnés. Chaque AG opère sur une population de chromosomes artificiels. Ce sont des chaînes dans un alphabet fini (généralement binaire). Chaque chromosome représente une solution à un problème et possède une aptitude, un nombre réel qui mesure la qualité de la solution apportée au problème considéré.

Les algorithmes génétiques sont utilisés dans la découverte de connaissances dirigée. Ils permettent de résoudre des problèmes divers, notamment d'optimisation, d'affectation ou de prédiction. Leur fonctionnement s'apparente à celui du génome humain. Le principe de fonctionnement est le suivant : les données sont converties en chaînes binaires (comme les chaînes d'ADN - acide désoxyribonucléique-). Celles-ci se combinent par sélection, croisement ou mutation et donnent ainsi une nouvelle chaîne qui est évaluée.

## *II.2 Terminologie propre aux algorithmes génétiques*

Les AGs utilisent les éléments suivant pour se définir :

- ✓ **Codage :** de l'élément de population : cette étape associe à chacun des points de l'espace d'état une structure de données. La qualité du codage des données conditionne le succès des algorithmes génétiques.

- ✓ **Un gène** : est un ensemble de symboles représentant la valeur d'une variable. Dans la plupart des cas, un gène est représenté par un seul symbole (un bit, un entier, un réel ou un caractère).
- ✓ **Un chromosome** : représente le codage d'une solution potentielle au problème à résoudre. Il est constitué d'un ensemble d'éléments appelés gènes, pouvant prendre plusieurs valeurs binaire ou réelle.
- ✓ **Un individu** : est composé d'un ou de plusieurs chromosomes. Il représente une solution possible au problème traité.
- ✓ **Une population** : est représentée par un ensemble d'individus (i.e. l'ensemble des solutions du problème).
- ✓ **Une génération** : est une succession d'itérations composées d'un ensemble d'opérations permettant le passage d'une population à une autre

- ✓ **Environnement** : l'espace de l'exploration et de la recherche.

- ✓ **Fonction d'évaluation (fitness)** : la fonction objective qui sert à évaluer la précision de la solution du problème.

- ✓ **fonction d'adaptation ou fonction de performance :** fonction regroupant plusieurs fonctions objectives et quantifiant les performances d'un individu avec une valeur numérique.

## *II.3 Fonctionnement des algorithmes génétiques*

Tout algorithme génétique à une structure commune, il opère de la même manière et suivent la même méthodologie. Chaque algorithme génétique est déterminé par son cycle de base, qui est représenté par la **figure (II.4)**. Tout d'abord, un ensemble de solutions ou d'individus est généré afin de former la population initiale. Par la suite, les individus de la population sont évalués et sont classés dans la population. Ensuite, un sous-ensemble de parents est créé de façon à favoriser les meilleures solutions et à partir de cet ensemble, un groupe d'enfants est généré par des mécanismes de reproduction (mutation et croisement).

Après avoir créé l'ensemble d'enfants, la performance de ceux-ci est évaluée, afin de comparer leur degré de performance par rapport au reste de la population et pour ainsi guider a construction de la nouvelle génération. Ensuite, le processus se répète à partir de cette nouvelle génération jusqu'à ce que soit atteint le critère d'arrêt défini au départ. ***[II.12]***

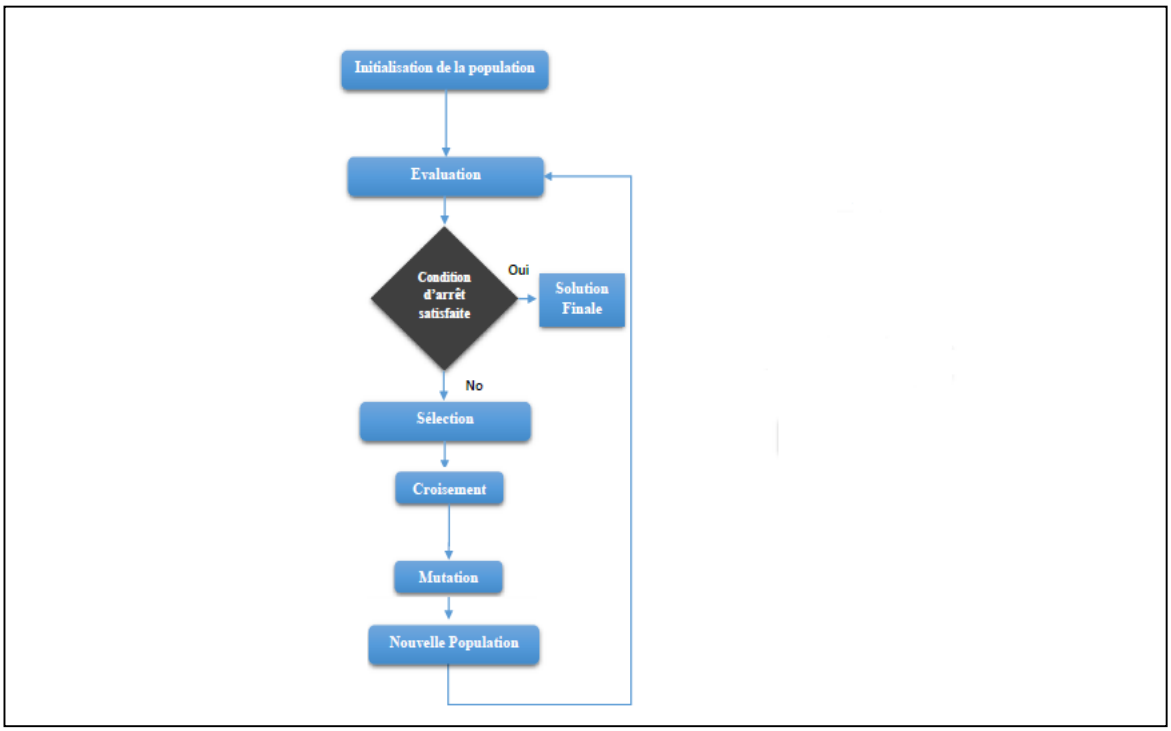



**Figure II.4** : **Structure de l'algorithme génétique** ***[II.25]***

### II.3.1 Codage d'individus

Avant de passer à l'explication des différents processus génétiques, il faut tout d'abord expliquer le codage des individus. La procédure normale pour coder un algorithme génétique ayant plusieurs paramètres est de coder chaque paramètre comme une séquence de bits. Les séquences sont ensuite tronquées l'une après l'autre pour former une grande séquence, le chromosome, qui représente le vecteur des paramètres. Chaque séquence du vecteur total représente un gène, et la valeur de chaque gène est un allèle. ***[II.13]***

Example : Soit un vecteur $\bar{x}$ composé de trois paramètres $x_1$, $x_2$, et $x_3$, codés à 4 bits.

$$x_1 = 0011$$
$$x_2 = 1010$$
$$x_3 = 0100$$

La séquence totale serait la suivante :

$$\bar{x} = \{\overset{x_1}{0011}|\overset{x_2}{1010}|\overset{x_3}{0100}\}$$

### II.3.2 Génération de la population initiale

Le processus de l'optimisation par AG commence par choisir aléatoirement dans l'espace de recherche un nombre fini d'individus qui vont constituer la population initiale.

### II.3.3 Evaluation des individus

Un opérateur d'évaluation, qui mesure l'adaptation des individus aux conditions imposées, et qui dépend donc du problème à traiter normalement, cette évaluation est faite à l'aide d'une fonction objective (fitness) qui mesure la qualité de l'individu comme solution au problème.

### II.3.4 Les opérateurs

Un algorithme génétique simple utilise les trois opérateurs suivants : la sélection, le croisement et la mutation.

#### a) Sélection

Comme son nom l'indique, la sélection permet d'identifier statistiquement les individus de la population courante qui seront autorisés à se reproduire. Cette opération est fondée sur la performance des individus, estimée à l'aide de la fonction objective. Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se

reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population.

Un opérateur simple de sélection est la technique de la roulette (la roue biaisée), avec cette méthode chaque individu a une chance d'être sélectionné proportionnelle à sa performance, donc plus les individus sont adaptés au problème, plus ils ont de chances d'être sélectionnés. Ainsi un individu Xi a la probabilité suivante d'être sélectionné :

$$P_s(X_i) = \frac{F(x_i)}{\sum_{k=1}^{N} F(x_k)} \qquad (2)$$

Où **Ps** est la proportion que fait la valeur de la fonction objective f (***xi***) d'un individu ***xi***, par rapport à la somme de tous les objectifs des ***N*** individus de la population. De cette façon, plus un individu a une forte qualité ***F*** par rapport au reste de la population, plus celui-ci a de chances de se reproduire. Toutefois, comme la sélection se fait aléatoirement, les moins bons individus ont tout de même une chance de faire partie de l'ensemble de reproduction.

### b) *Le croisement*

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants. Il consiste à échanger les gènes des parents afin de donner des enfants qui portent des propriétés combinées. Bien qu'il soit aléatoire, cet échange d'informations offre aux algorithmes génétiques une part de leur puissance : quelque fois, de bons gènes d'un parent viennent remplacer les mauvais gènes d'un autre et créent des fils mieux adaptés aux parents, il existe plusieurs variantes de cet opérateur, parmi les opérateurs de croisement les plus connus dans la littérature, nous citons : ***[II.14]***

- ✓ ***Le croisement 1-point***

Le croisement 1-point consiste à choisir un seul point de coupure, puis échanger les fragments situés après ce point de coupure. Comme la montre la figure ci-dessus.

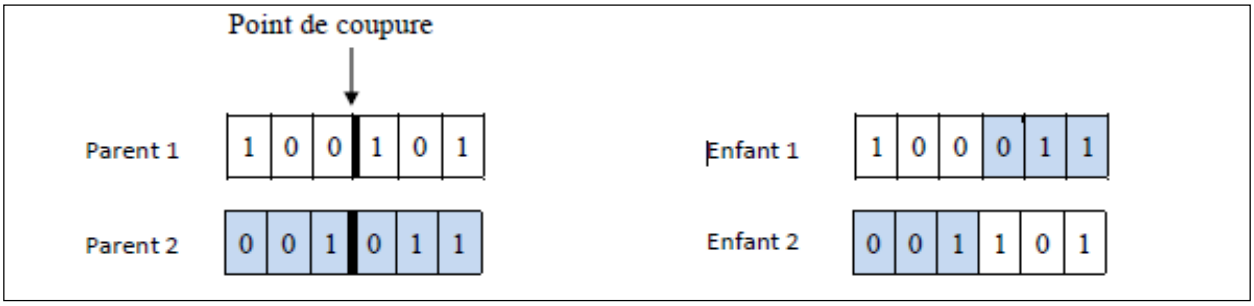



**Figure II.5 : croisement à 1-point.** ***[II.26]***

- ✓ ***Le croisement 2-point***

Le croisement 2-point consiste à choisir deux points de coupure, puis échanger les fragments situés entre ces deux points comme le montre la figure suivante.

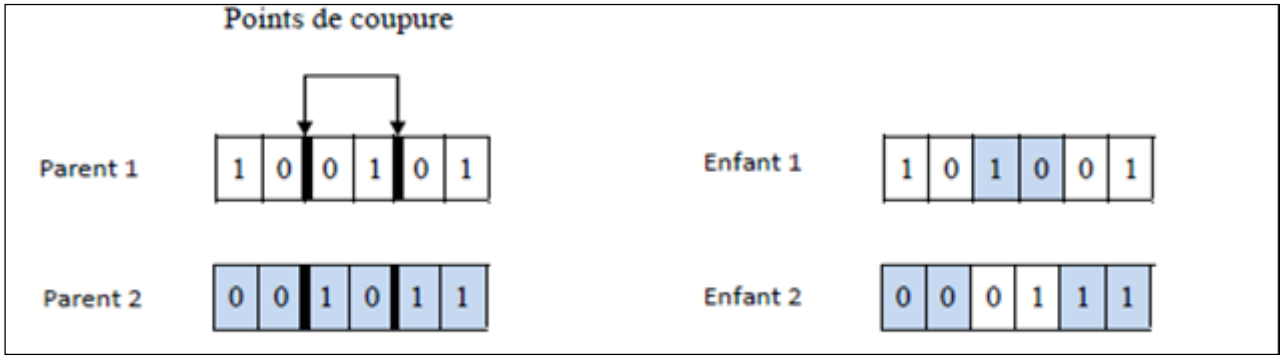



**Figure II.6 : Croisement 2-points.** ***[II.26]***

### *c) La mutation*

La mutation est une modification aléatoire de la valeur d'un gène qui se produit avec une probabilité fixée. Le but de la mutation est de préserver la capacité de l'AG à trouver de nouvelles bonnes solutions en gardant une certaine diversité génétique dans la population.

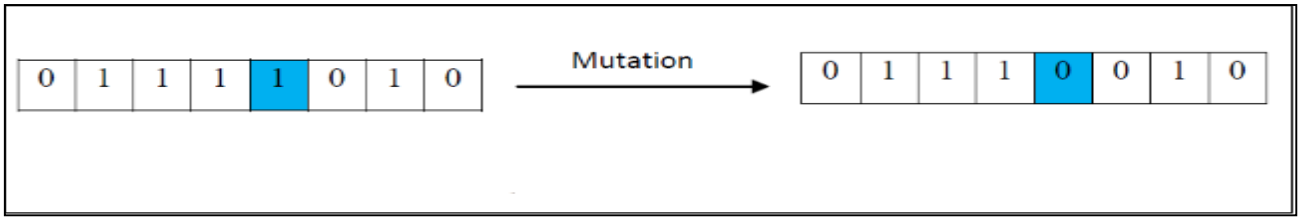



**Figure II.7 : Mutation d'un gène** ***[II.26]***

### *d) Retour*

Tant que la condition d'arrêt n'est pas satisfaite, l'algorithme recommence à partir de la phase d'évaluation.

## *II.4 Les paramètres des algorithmes génétiques*

La rapidité et la qualité de convergence d'un algorithme génétique sont influencées par de nombreux paramètres :

**1. taille de la population** (notée **Npop**) : plus le nombre d'individus est grand, plus la probabilité de convergence est élevée, mais plus l'algorithme est lent pour calculer chaque génération.

**2. le nombre de générations maximales** que doit effectuer l'algorithme, qui se définit le plus souvent en fonction de la taille de la population et du temps imparti pour résoudre le problème.

**3. Le taux de croisement (Pc) :** plus ce taux est grand, plus de nouvelles structures sont introduites dans les nouvelles générations par conséquent les structures performantes sont très fréquemment détruites. Par contre, si ce taux est trop bas, la population n'évolue pas assez vite. Il est généralement fixé entre 0.6 et 0.9.

**4. Le taux de mutation (Pm**) : souvent ce taux est fixé entre 0.001 et 0.01 ou dépend de la taille L de chromosome. Il est possible d'associer une probabilité différente à chaque gène. Ces probabilités peuvent êtres fixes ou évoluer dans le temps.

**5. l'élitisme** de l'algorithme, qui consiste à forcer la conservation d'un ou des meilleurs individus trouvés, tant qu'ils ne sont pas dépassés par d'autres. Cette stratégie assure la croissance de la fonction d'adaptation du meilleur individu de la population, mais peut nuire à la diversité de la population.

Le réglage de ces paramètres est souvent fastidieux, il nécessite beaucoup d'essais et dépend fortement du problème (de la fonction) à optimiser. ***[II.15]***

## II.5 Propriétés des algorithmes génétiques

Les propriétés qui caractérisent les AGs sont : ***[II.16]***

- ✓ Le mécanisme d'évolution et de sélection est indépendant du problème à résoudre : souvent un même algorithme peut être utilisé dans des domaines différents. Pour cela il suffit d'adapter les opérateurs utilisés au problème traité.
- ✓ équilibre entre l'exploration et l'exploitation : nombreux sont les problèmes que résout l'AG et qui ont été difficile à résoudre par la programmation classique.
- ✓ Adaptabilité dans le temps : l'exploration étant éventuelle (probabilité non nulle) dans toutes les directions de recherche. Les AGs sont ainsi capables de retourner un optimum variable dans le temps
- ✓ Non optimalité : l'optimum ne peut être atteint malgré que l'AG approche la solution, c'est dans ce contexte que ce dernier recherche la solution au voisinage de l'optimum.
- ✓ Exploration parallèle : l'AG agit sur un ensemble de solutions, ce qui permet d'effectuer une recherche parallèle tout en échangeant des informations entre les différentes recherches (croisement). En manipulant k individus, l'AG explore k direction de recherches.

## II.6 Convergence (Critère d'arrêt)

Après ces diverses opérations génétiques, un nouvel ensemble de variables est obtenu. A ces nouveaux individus, la fonction d'adaptation F est appliquée pour déterminer le maximum.
L'AG travaille par générations successives jusqu'à ce qu'un critère d'arrêt soit vérifié. Ce critère d'arrêt peut être de nature diverse : ***[II.17]***

- la nième génération est atteinte
- le meilleur élément de la dernière génération a atteint un seuil de qualité fixé au départ.
- aucune évolution du meilleur individu n'est perceptible après un nombre donné de générations.

## III- Algorithme de la vie des abeilles (Bees Life Algorithm)

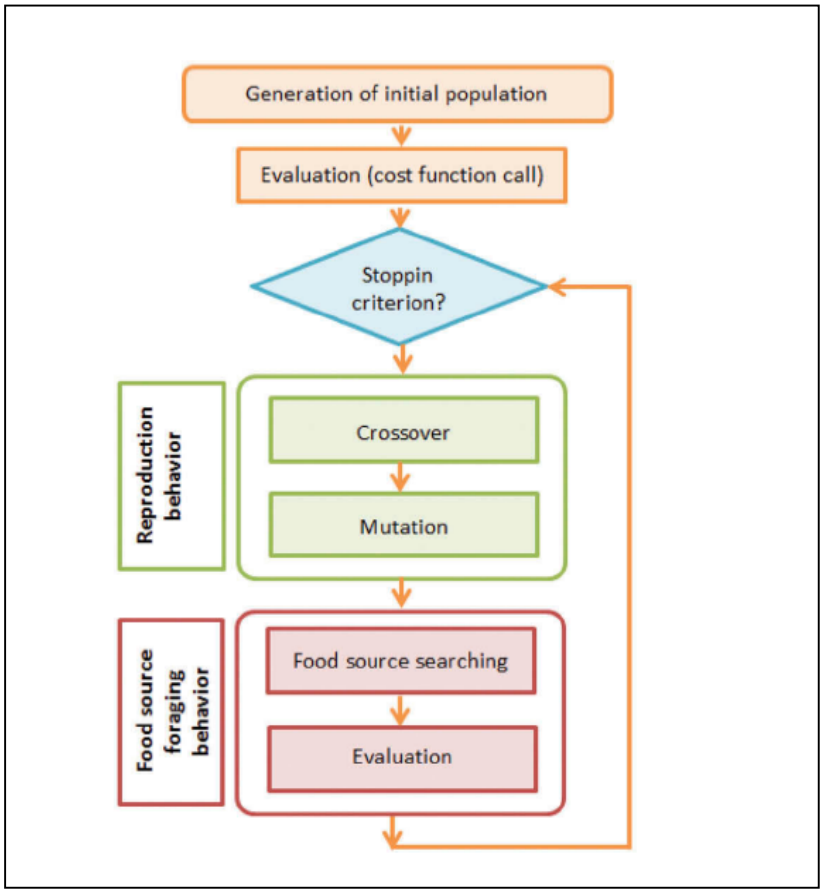
L'algorithme de vie d'abeille (BLA) s'inspire du comportement naturel des abeilles, à savoir le couplage (reproduction) et la recherche de nourriture. **La figure II.8** montre le pseudo-code pour un algorithme d'abeille simple.

L'algorithme commence par choisir N abeilles (individus) au hasard à partir de l'espace de recherche pour former la population initiale. Ensuite, cette population initiale est évaluée en appelant une fonction de coût. Après avoir trié cette population, l'abeille la plus apte est considérée comme la reine, les drones «D» sont les abeilles les plus aptes, tandis que les autres abeilles constituent les ouvrières (leur nombre est «W»).

Nous mentionnons que les valeurs de 'N', 'D' et 'W' sont fixées par l'utilisateur en tant que valeurs expérimentales. L'étape suivante consiste à exécuter un certain nombre d'itérations BLA jusqu'à ce que le critère d'arrêt soit atteint. Chaque itération représente le cycle de vie des abeilles (itération BLA), elle consiste en les deux principaux comportements des abeilles (c.-à-d. reproduction et alimentation).

Pendant la phase de reproduction, un vol d'accouplement est effectué où la reine s'accouple avec un ensemble de drones par les opérateurs de croisement et de mutation. Par conséquent, "N" couvées sont pondues par la reine. Tous les individus de la ruche (c.-à-d. Les «N» abeilles originales et les «N» couvées) sont évalués et triés en appelant la fonction de coût par conséquent, les meilleures abeilles remplaceront le Reine précédente, les abeilles D les plus aptes sont sélectionnées pour former les nouveaux drones et le Les «meilleures abeilles» suivantes sont considérées comme les nouveaux travailleurs.

Maintenant, la phase de recherche de nourriture est exécutée par les ouvrières. Chaque travailleur est responsable de trouver une source de nourriture (région de fleurs) puis de recruter plusieurs autres travailleurs pour récupérer la nourriture trouvée. Nous notons qu'il y a plus de travailleurs («FBest») recrutés dans des régions plus prometteuses «B» où il y a plus de nourriture par rapport aux nombre de travailleurs (« FOther ») recrutés dans les régions restantes. Ensuite, le meilleur ouvrier de chaque région reste en vie dans la population suivante. Ce choix du meilleur ouvrier se fait avec l'appel de fonction de coût. ***[II.18]***




**La figure II.8 : pseudo-code pour un algorithme d'abeille simple *[II.22]***

## *III.1 Les abeilles en nature*

Les abeilles possèdent des propriétés assez différentes de celles des autres espèces d'insectes. Elles vivent en colonies, en construisant leurs nids dans des troncs d'arbres ou d'autres espaces clos similaires. Généralement, une colonie d'abeilles contient une femelle reproductrice appelée reine, quelques centaines de mâles connus sous le nom de faux bourdon, et de 10.000 à 80.000 femelles stériles qui s'appellent les ouvrières. Après accouplement avec plusieurs faux-bourdon, la reine reproduit beaucoup de jeunes abeilles appelées les couvées.

- ***La Reine***

Dans une colonie d'abeilles, il y a une seule reine qui est la femelle reproductrice avec l'espérance de vie entre 3 et 5 ans. Son rôle principal est la reproduction, elle s'accouple avec (7-20) faux-bourdon dans une opération de reproduction appelée le vol nuptial (mating

flight).Elle conserve les spermes dans sa spermathèque et puis pond jusqu'à 2000 œufs par jour. Les œufs fertilisés vont bien devenir femelles (ouvrières) et les œufs non-fertilisés deviennent mâles (bourdons). ***[II.19][II.20]***

- ***Le male (faux-bourdon)***

Au sein de la famille des Apidés, se trouvent plusieurs genres, et notamment les bourdons, qu'il ne faut pas confondre avec les faux-bourdons, les mâles de l'abeille domestique. Les faux-bourdons représentent les mâles, variant entre 300 et 3000 dans une ruche. Selon la taille de l'alvéole, dans une grande cellule hexagonale, la reine dépose un œuf sans que son réceptacle séminal laisse sortir de spermatozoïde. L'œuf non fécondé donnera ainsi naissance à un faux-bourdon. Ce dernier a une espérance de vie de 90 jours. Après l'accouplement, qui a lieu en vol, le male meurt rapidement.

- ***Ouvrières***

Les ouvrières sont les abeilles femelles mais elles ne sont pas reproductrices, elles vivent de 4 à 9 mois dans une saison froide et leur nombre arrive jusqu'à 30.000 dans une colonie (ruche). Cependant, en été, leur durée de vie est de 6 semaines et leur nombre atteint jusqu'à 80.000.L'ouvrière est responsable de la défense de la ruche utilisant sa piqûre barbelée. En conséquence, elle meurt après avoir piquée. On peut énumérer les activités des ouvrières par le critère des jours de sa vie comme suit : nettoyage de cellules (jour 1-2), soigner les abeilles (jour 3-11), production du cire (jour 12-17), surveiller les autres abeilles (jour 18-21), et recherche de nourriture (jour 22-42). L'ouvrière assure les activités habituelles de la colonie d'abeilles telles que l'emballage de pollen, éventer les abeilles, porter l'eau, déplacement des œufs, s'occuper de la reine, nourrir les bourdons, et construction du nid d'abeilles. ***[II.19][II.20]***

- ***Couvées***

Les jeunes abeilles sont nommées les couvées. Elles sont nées après que la reine pond un œuf fécondé par un spermatozoïde libéré par la spermathèque dans des cellules spéciales du nid d'abeilles appelées trames de couvée (brood frames). L'œuf fécondé donne naissance à une abeille ou à une reine en fonction des soins et de la nourriture apportés aux larves. Peu de larves femelles sont choisies pour être des futures reines. Les œufs non fertilisés donnent naissance aux couvées. Les jeunes larves tournent par le cocon, couvrant la cellule par les ouvrières les plus âgées. C'est l'étape de chrysalides. Puis, elles atteignent l'étape de

développement dans laquelle elles reçoivent le nectar et le pollen des abeilles chercheurs jusqu'à ce qu'elles quittent la ruche et passent leurs vies comme chercheurs de nourriture. ***[II.19][II.20]***

## *III.2 Recherche de nourriture chez les abeilles*

Les scientifiques ont effectué beaucoup de recherches pour déterminer comment l'ordre est maintenu dans la ruche où vivent des dizaines de milliers d'abeilles. Un grand nombre d'études académiques ont été aussi effectuées à cette fin. Un éminent expert et professeur à l'Université de Munich, le zoologiste autrichien Karl Von Frisch, a consacré un livre de 350 pages à la communication des abeilles, "The dance language and orientation of bees" (Le langage de la danse et l'orientation des abeilles). ***[II.21]***

### *III.2.1 Exploration des sources de nourriture*

L'abeille qui cherche à manger, appelée « éclaireuse », se trouve devant une tâche formidable. Elle quitte la ruche, cherche dans de vastes zones et vole de longues distances, elle doit donc reconnaitre les bonnes sources de nourriture et avant que les abeilles butineuses retournent à la ruche, elle doit enregistrer sa position par rapport à la ruche et déposent une odeur spéciale sur leur source de nourriture. Une fois qu'une source de nourriture a été localisée et marquée, elle doit retourner à la ruche pour informer les autres ouvrières de sa découverte. Pour cela, elle doit utiliser des informations sur la direction de la ruche même si, pour arriver où elle est, Elle fait tout un circuit. Après, elle doit communiquer aux autres ouvrières la direction et la distance de la source de nourriture par rapport à la ruche, et elle doit donner certaines indications sur sa qualité. Ceci doit être fait de la façon la plus économique possible. Elle doit donc attirer l'attention des autres ouvrières, qui peuvent déjà être occupées à autre choses ou avoir reçu des messages d'autres abeilles. La question posée est alors : « Comment les abeilles communiquent elles ? ***[II.21]***

### *III.2.2 Méthode de communication chez les abeilles*

Karl Von Frisch a construit une ruche avec une paroi en verre par laquelle il pouvait observer le comportement des abeilles à l'intérieur. Il remarqua que dans le retour des abeilles employeuse à la ruche, elles exécutent la danse. Si une seule abeille l'exécute, ce n'est pas toute la ruche qui passe à l'action. Premièrement, un groupe d'éclaireuses quitte la ruche. Si, à

son retour, ce groupe exécute aussi la danse, alors un groupe plus important d'abeilles se dirigera vers l'objectif. Meilleure est la source de nourriture qu'elles trouvent, plus longtemps dure la danse et plus grand sera le nombre d'abeilles qui les suivent. De cette façon, l'attention des butineurs est toujours fixée sur la source la plus productive.

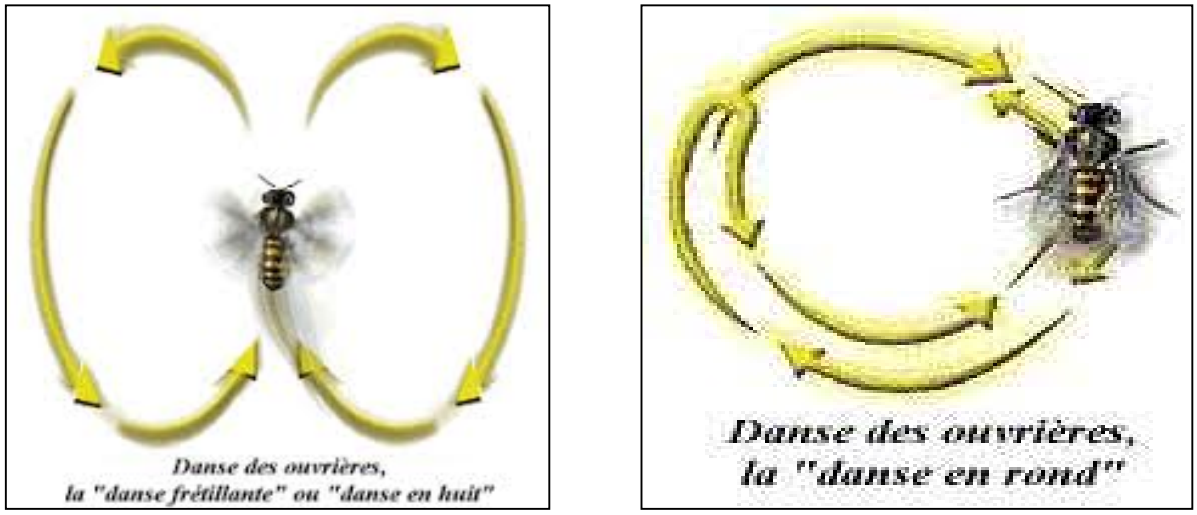



**Figure II.9 : la danse des ouvrières** ***[II.21]***

Si la source de nourriture qu'elles trouvent est très riche, la danse que les abeilles exécutent est très enthousiaste. Si la source est à proximité, elles décrivent son emplacement en exécutant la "danse en rond" illustrée à droite. Pour les sources de nourriture plus éloignées elles exécutent la danse en huit illustrée à gauche, avec des mouvements frétillants.

Dans la danse frétillante, l'abeille tourne alternativement d'un côté et de l'autre effectuant un demi-tour circulaire qui la ramène au point de départ.

Les ouvrières suivent la danseuse au cours de ses déplacements, et la touchent des antennes. Cette danse est très riche en information, en particulier, elle donne aux abeilles observatrices deux indications essentielles : la quantité de nourriture, la distance à parcourir et la direction à suivre. ***[II.21]***

Concernant la distance, plusieurs paramètres sont utilisés simultanément, le plus important est le tempo de la danse, c'est-à-dire le nombre de tours complets effectués par la danseuse par unité de temps. Les mouvements de l'abeille sont d'autant plus durables que la source est riche (quantité de nourriture).

En ce qui concerne la direction, elle est donnée par rapport à l'azimut solaire (en plan horizontal), la position angulaire de la source est indiquée par l'angle que fait la partie frétillante avec la verticale du rayon comme l'indique la **figure II.10**.

Dans le cas où la source de nourriture trouvée serait improductive, les abeilles dansent tout de même, mais elles le font à contre cœur et pendant une plus brève durée. Cela se reflète aussi sur les autres abeilles dans la ruche, les abeilles qui se rassemblent autour de la danseuse se dispersent rapidement et une nouvelle équipe quitte la ruche à la recherche de nourriture.

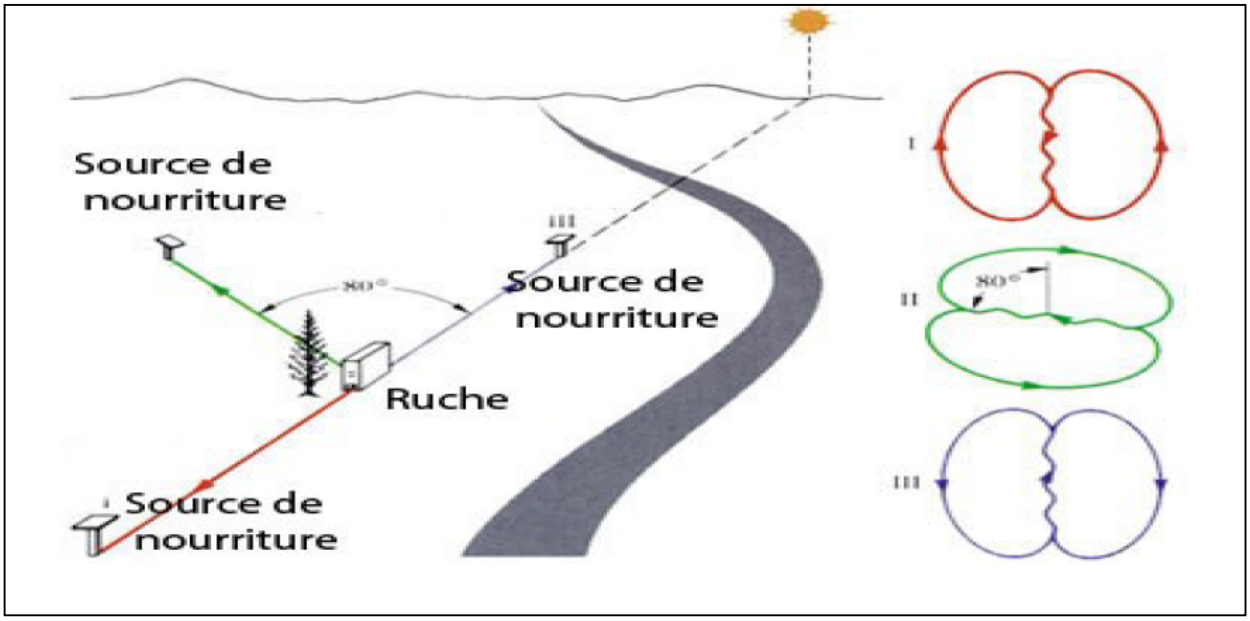



**Figure II.10 : l'indice de la direction *[II.21]***

1- Si la source de nourriture se trouve exactement dans la direction du soleil, ou dans la direction opposée, les frétillements dans la danse seront verticaux sur le rayon.

2 - Si la source de nourriture est 80 degrés à gauche du soleil, cela est indiqué en faisant la trajectoire du frétillement de la danse à un angle correspondant de 80 degrés à gauche de la verticale.

3- Si l'abeille suit une direction vers le haut durant son frétillement, elle signale que la source de nourriture se trouve dans la direction vers le soleil. Et si elle se dirige droit en bas, cela signifie que la source se trouve dans la direction exactement opposée de celle du soleil. ***[II.21]***

### *Optimisation naturelle : Les directions fournies par les abeilles*

Comme il a été déjà mentionné, peu après avoir regardé l'abeille danseuse, les autres ouvrières quittent la ruche et se dirigent vers l'objectif. Cependant, les abeilles sont confrontées à un problème important : l'angle que la danseuse a fourni à ses sœurs est basé sur le soleil. Cependant, le soleil n'est pas fixe dans le ciel, mais il change de position de 1 degré à toutes

les 4 minutes. Si une abeille suivait la ligne d'origine, elle ne serait jamais capable de localiser son objectif à cause du changement de position du soleil. Chaque fois que 4 minutes s'écoulent, cela correspond à une marge d'erreur de 1 degré, laquelle atteindra des dimensions impossibles à corriger au cours d'un long voyage. En réalité, cela ne se produit jamais. Depuis maintenant des millions d'années, les abeilles ont compris les directions qui leur sont fournies par leurs sœurs, malgré le mouvement du soleil et le changement des angles. Les abeilles n'éprouvent aucune difficulté à trouver des sources de nourriture, ce qui indique qu'elles ne font aucune erreur en calculant l'angle par rapport au soleil.

Pour exprimer cela en termes mathématiques, les abeilles calculent que le soleil se déplace de 1 degré à chaque 4 minutes. En conséquence, elles sont capables de se rappeler de l'emplacement exact de la source de nourriture et de le "décrire" aux autres abeilles. Les autres abeilles calculent l'angle suivant la position modifiée du soleil, comprennent les directions données et localisent la source de nourriture en question. ***[II.21]***

***Conclusion***

Dans ce chapitre, nous avons présenté un état de l'art sur les métaheuristiques d'optimisation et quelques métaheuristiques très répandues comme l'algorithme génétique GA. Nous avons également détaillé l'algorithme inspiré de vie des abeilles (BLA) et enfin nous avons abordé les notions de base et le principe de son fonctionnement.

# *Chapitre III :*

# *Application de BLA pour le problème d'ordonnancement*

## *I. Introduction*

Dans ce chapitre, nous présenterons laformulation du problème d'ordonnancement des tâches dans le Cloud, l'initialisation de l'algorithme BLA, les notions général ainsi les opérateurs d'optimisation de l'algorithme. Enfin nous détaillerons l'approche de recherche locale de la métaheuristique BLA.

## *II. Formulation du problème*

Nous définissons un Jobcomme un processus correspondant à une demande de service établie par un utilisateur de cloud computing.Ensuite, un ensemble de "n" jobs pourrait être soumis afin d'être ordonnancé et exécuté par l'infrastructure informatique de cloud. Ces jobs sont représentés par l'ensemble de jobs suivant :

$$\mathbf{Jobs} = \{ j_1, j_2, \dots , j_j , \dots , j_n \}$$

Chaque job «j» parmi «n» jobs (c'est-à-dire $1 \leq j \leq n$) peut être partitionné en un ensemble de «r» tâches. Chaque tâche a les propriétés suivantes : nombre d'instructions, mémoire requise, taille des fichiers d'entrée et de sortie.En supposantque $T_k$indique la $k^{éme}$tâche, à chaque fois, un ensemble de r tâches indépendantes est envoyées au système, comme suit :

$$\mathbf{T} = \{T_1, T_2, T_3, \dots T_k \dots , T_r\}$$

L'infrastructure informatique du Cloud Computing est composée de processeurs, c'est-à-dire de nœuds Cloud qui ont les attributs suivant, le taux de la CPU, les frais d'utilisation de la CPU, les frais d'utilisation de la mémoire et les frais d'utilisation de la bande passante. L'ensemble de m processeurs comprenant les nœuds c du Cloud est exprimé comme :

$$\mathbf{N} = \{N_1, N_2, N_3, \dots , N_m\}$$

Où $N_i$ présente le $i^{ème}$noeud de traitement.

Chaque tâche$T_k$ **($T_k$ ∈ Tasks)**est attribuée au processeur $N_i$**($N_i$ ∈Nodes)**représenté par $T_k^i$

Un processeur peut être affecté pour traiter un ensemble d'une ou plusieurs tâches :

$$N_i Tasks = \{ T_x^i , T_y^i , \dots , T_z^i \}$$

Le problème d'ordonnancement des tâches dans l'environnement informatique Cloud Computing pourrait être formulé comme suit :

Recherche d'un ensemble :

$$\boldsymbol{NodeTasks = \{ T_1^a, T_2^b, T_3^c, \ldots, T_r^m \}}$$

- ***La somme des temps des exécutions (flow time)***

Pour un ensemble de tâches $N_i Tasks$,**La somme des temps des exécutions (flow time)** définie comme le temps d'exécution (EXT) qu'a besoin le nœud $N_i$ pour terminer toutes les tâches qui lui ont été assigné, est calculé comme suit:

$$\textbf{Flow time} = \sum_{T_k^i \in N_i Tasks} \boldsymbol{ExeTime}(T_k^i) \qquad \textbf{(1)}$$

Où$\boldsymbol{ExeTime}(T_k^i)$est le temps d'exécution de $\boldsymbol{T_k}$ traité dans le nœud $\boldsymbol{N_i}$, calculé par:

$$\boldsymbol{ExeTime}(T_k^i) = \frac{\boldsymbol{lenght}(T_k)}{\boldsymbol{CPUrate}(N_i)} (2)$$

Avec longueur ($\boldsymbol{T_k}$) le nombre d'instructions de la tâche $\boldsymbol{T_k}$ et CPUrate ($\boldsymbol{N_i}$) le débit de la CPU du nœud Ni, quiest déterminé en fonction de la fréquence d'horloge, du nombre de noyaux, du parallélisme du niveau d'instruction, …etc.***[III.1]***

## *I.1 La Fonction objective d'évaluation*

L'optimisation est exprimée par une fonction devant minimiser ou maximiser un critère donné. La plupart des travauxde recherche ont porté sur une seule optimisation objective, comme la minimisation du retard du job (Tardness), la réduction de la charge de travail totale, la minimisation du makespan, … etc. En présence d'incertitude associée à la défaillance des machines, les ordonnancements devraient fonctionner dans des environnements dynamiques. Ainsi, l'ordonnancement doit satisfaire deux objectifset qui sont leTemps d'exécution de la CPU et la minimisation du makespan.

## *I.2 Minimisation du makespan*

Dans ce travail, notre premier objectif étant la minimisation du makespan, qui est le temps total nécessaire au système pour exécuter toutes les tâches, défini à partir du moment où la

demande est reçue à l'heure à laquelle la dernière tâche est terminée ou à l'heure à laquelle la dernière machine se termine. La fonction objective est la suivante:

**Makespan = max { $ExeTime$ (Ti) } (3)**

## *III. Formulation de la solution*

Puisque l'ordonnancement des jobs dans le cloud est considéré comme un problème d'optimisation. Nous supposons qu'un job (constitué d'un ensemble de tâches) est représenté par un graphique en commençant par un nœud racine, se terminant par un nœud feuille, et ses tâches sont représentées par un ensemble de nœuds intermédiaires, où la racine exprime le début d'un job et la feuille la fin du job. En conséquence, un chemin dans le graphique partant de la racine jusqu'à la feuille forme une solution (c'est-à-dire un individu), qui est représenté par une chaîne. Par exemple, nous avons un job «j» qui consiste trois tâches ($T_1$, $T_2$, $T_3$). Il existe de nombreuses possibilités pourordonnancer ces tâches par «m» nœuds du cloud. Par exemple, une configuration de solution est que ce job est ordonnancé comme suit: $T_1$dans $N_5$, $T_2$dans $N_7$et$T_3$dans $N_1$, comme illustré [4]

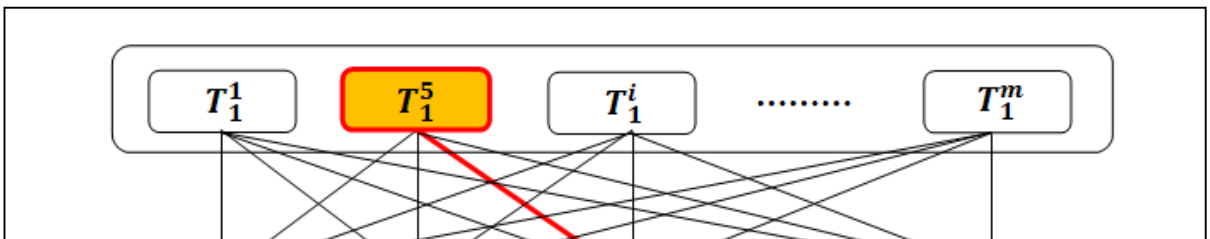


**Figure III.1 : la représentation des individus de BLA.*[III.1]***

## *III.1 Initialisation de l'algorithme BLA*

L'initialisation de l'algorithme BLA génère 'N'individus sélectionnés au hasard pour former la première population. ***[III.1]***

Par exemple, les individus (abeilles) suivantes sont sélectionnés, où l'individu 1 représente un ensemble de quatre tâches $T_w, T_x, T_y, T_z$exécutés par quatre nœuds du Cloud $N_a, N_b, N_c, N_d$ , respectivement.

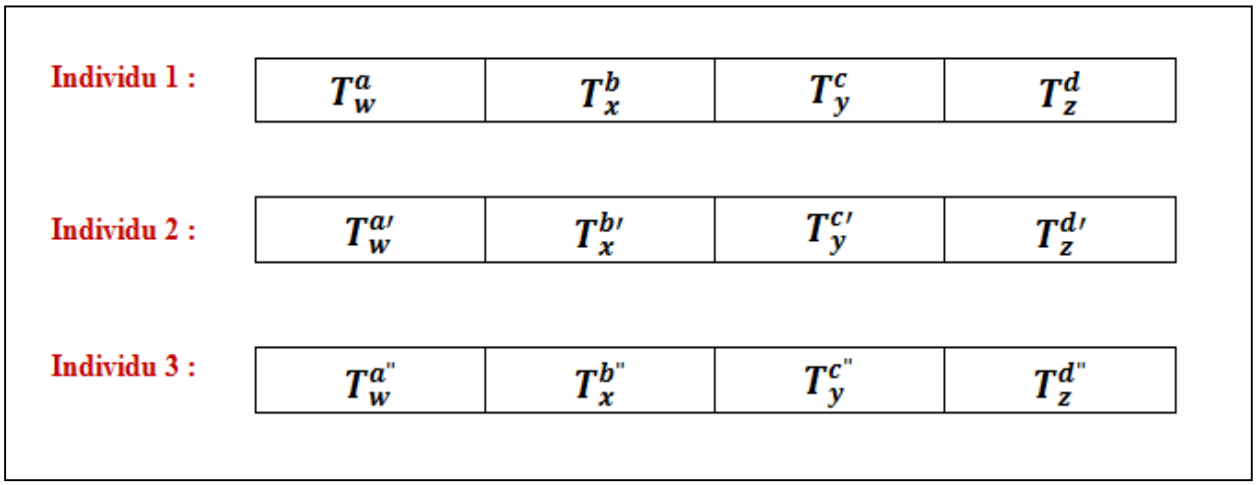



**Figure III.2 : La sélection de 3 individus**

## *II.1 Critère d'arrêt de l'algorithme BLA*

Le critère d'arrêt représente une valeur prédéterminée (MaxIT), fixée par l'administrateur de l'ordonnancement des travaux, en tant que nombre maximal d'itérations de l'algorithmeBLA. MaxIT pourrait correspondre à la limitation de ressources(performances du nœud de cloud) ou à une période de temps pendant laquelle un utilisateur peut prendre en charge avant d'obtenir les résultats. Il est à noter qu'une valeur de convergence pourrait être définie inférieur à MaxIT, ce qui est le cas lorsqu'il n'y a pas d'amélioration significative de la valeur de la fonction objective, d'une génération à l'autre. ***[III.1]***

## *II.2 Opérateurs d'optimisation de BLA*

BLA suggère deux (02) opérateurs génétiques d'être en phase de reproduction afin d'accroître la diversité des individus dans les populations générées. Ces opérateurs sont: le croisement et la mutation.

Cependant, au stade de la recherche de sources de nourriture, une approche de recherche localeest appliquée améliorer les solutions obtenues en phase de reproduction.

Cet algorithme glouton améliore la solution obtenue par un processus itératif en appliquant des modifications locales à l'espace de recherche jusqu'à ce qu'une solution optimale soit découverte lorsqu'il n'y a pas d'amélioration de la valeur de la fonction objective de cette solution. Dans les sections suivantes, nous illustrons l'utilisation de ces opérateurs.***[III.1]***

### *II.3.1 Croisement*

L'opération de croisement s'applique à deux individus de la colonie appelés parents qui sont la reine et un drone. Ces parents sont combinés pour former de nouvelles personnes appelées progénitures. Ce processus est effectué par la reine et les drones «D» afin de produire une descendance «N». La reine et les drones «D» sont sélectionnés parmi les meilleurs individus existants dans la population initiale en appliquant la fonction de coût. De cette façon, une meilleure progéniture devraient apparaître dans la prochaine génération, et ainsi de suite jusqu'à ce qu'une solution globale soit trouvée. Nous notons que le **croisement** est appliqué avec une probabilité de «Pc», ce qui est souvent une valeur élevée. Plusieurs opérateurs de croisement ont été proposés dans la littérature. Dans ce travail, nous sélectionnons l'opérateur de croisement à deux points (comme illustré à la **Figure II.13**). Avec un croisement à deux points, les deux points sont sélectionnés aléatoirement pour la chaîne qui représente chaque parent : dans notre cas, le parent est représenté par un chemin de graphe qui est représenté à son tour par une chaîne (comme décrit précédemment). La première partie (du début au premier point de croisement) et la troisième partie (à partir du deuxième point de croisement à la fin de la chaîne représentant le premier parent (c'est-à-dire la reine) est copié pour former les première et troisième parties de la première progéniture. La deuxième partie (à partir du premier point de croisement au deuxième point de croisement) du deuxième parent (c'est-à-dire un drone) est copié dans forment la deuxième partie de cette première progéniture. Ce premier produit hérite des parties de chaîne de ses deux Parents. De même, la deuxième progéniture est formée en héritant de ses première et troisième parties de la première et troisième partie du deuxième parent, tandis que sa deuxième partie est héritée de la deuxième partie du premier parent.

Nous proposons, pour l'ordonnancement destâches dans le cloud, un opérateur de croisement à deux points, **Figure II.13** Sur cette figure, les premier etsecond points de croisement sont sélectionnés au hasard après le premier et le quatrième représentent les tâches de l'individu. Dans ses première et troisième parties, la première progéniture (Pi') est formée par la première et troisième parties du premier parent (Pi), alors La partie médiane de la progéniture est identique à la partie médiane semblable à celle du deuxième parent (Pj).***[III.1]***

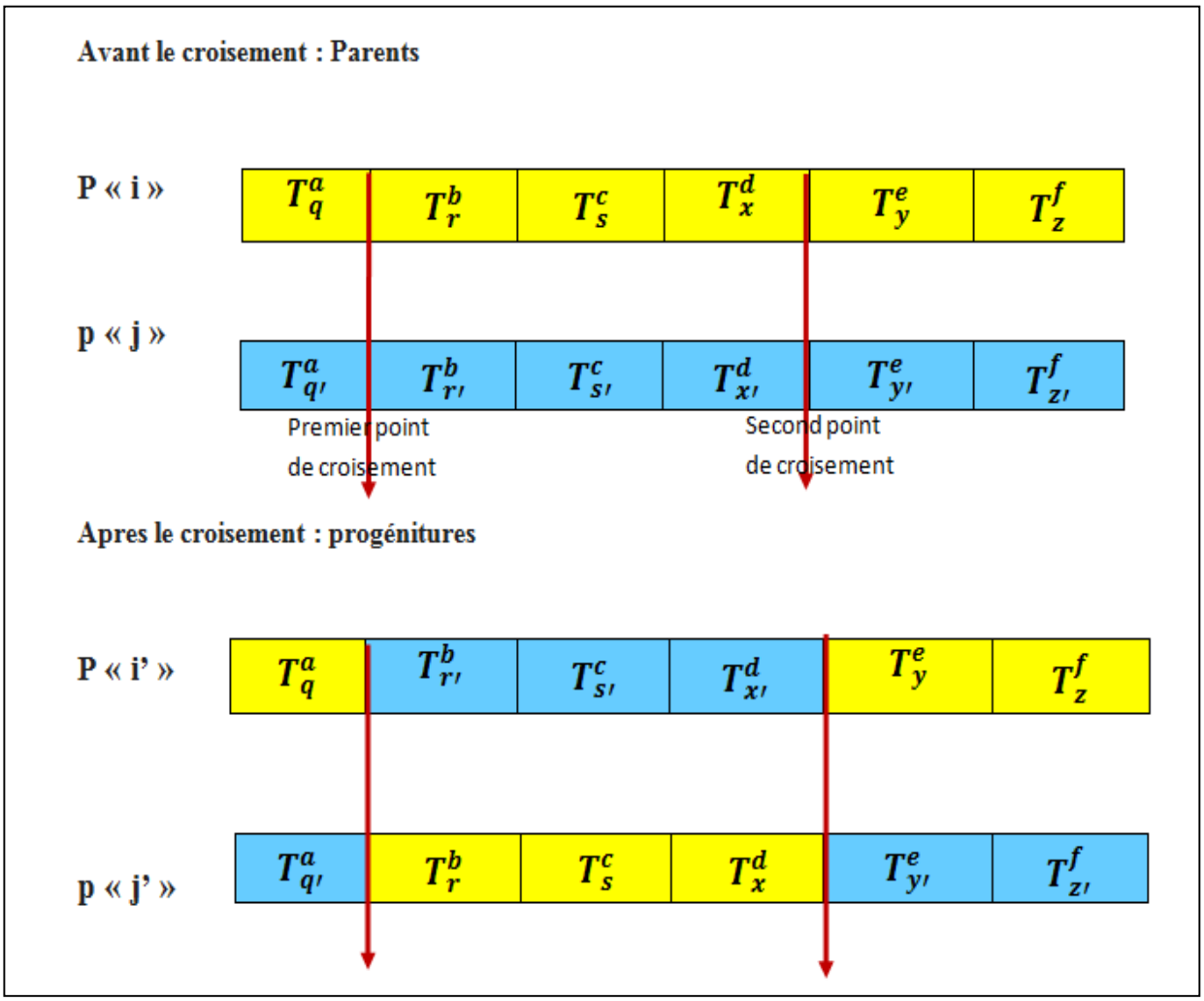



**Figure III.3 : Le croisement de 2 parents *[III.1]***

### *II.3.2 Mutation*

La mutation est un opérateur unaire qui introduit des modifications dans les caractéristiques des enfants résultant de l'opération de croisement. Ces changements sont très petits et aléatoires en fonction de la probabilité de mutation de «Pm», généralement choisie comme une valeur faible. Par conséquent, la nouvelle progéniture ne sera pas différente de l'original. Dans notre proposition, nous proposons un opérateur de mutation appelé tâche inversion. La mutation par inversion de tâches sélectionne une tâche au hasard, qui est ensuite remplacée par une autre tâche également sélectionnée au hasard dans le même nœud du cloud.

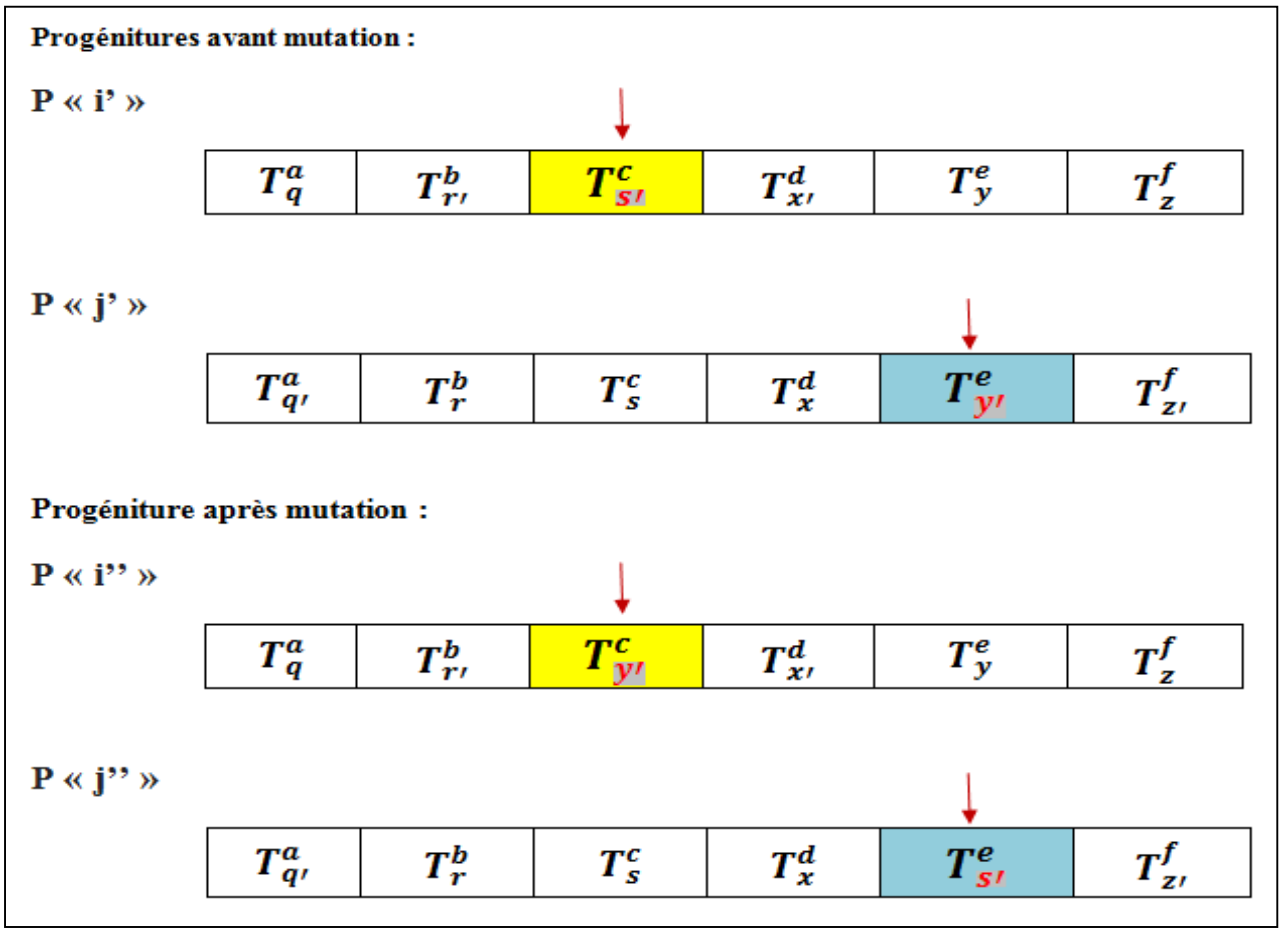



**Figure III.4 : La mutation d'une Progéniture P *[III.1]***

La figure illustre une mutation par inversion de tâches desdeuxprogénitures Pi' et j' qui seront Pi'' et j'' après la mutation, respectivement. Dans cet exemple, la tâche Ts' est remplacé par la tâche Ty' conçu le même nœud du cloud «i».

## *IV. Approche de recherche locale*

Dans la phase de recherche de nourriture, nous avons adopté la démarche de l'algorithme de colonie d'abeilles artificielles (ABC)quenous décrivons brièvement ci-dessous

### *III.1 Description de l'algorithme ABC*

Dans un algorithme d'optimisation par colonies d'abeilles artificielles (ABC), chaque solution représente uneposition de nourriture potentielle dans l'espace de recherche et la qualité de lasolution correspond à la qualité de la position alimentaire. La colonie d'abeilles artificielles est composée de trois types d'abeilles : les ouvrières, les spectatrices et les scoutes :

- L'ouvrière exploite la source de nourriture trouvée. Elle se base sur sa mémoire et essaye d'apporter des modifications à sa position (solution) actuelle pour découvrir une nouvelle position (i.e. source de nourriture).
- La spectatrice attend le retour des ouvrières au champ de danse pour observer leurs danses et recueillir des informations sur les sources de nectar qu'elles ont trouvé.
- L'abeille scoute exploite l'espace de recherche en lançant une recherche aléatoire d'une nouvelle source de nourriture.

Les ouvrières ($\mathbf{N_O}$) sont associés à des solutions actuelles de l'algorithme. À chaque étape de l'algorithme un $\mathbf{N_O}$ tente d'améliorer la solution

La population initiale de solution est constituée d'un nombre $\mathbf{N_{FS}}$ de sources de nourriture générées aléatoirement dans l'espace de recherche. En supposant que la $i^{ème}$ source de nourriture de la population soit représentée par $\mathbf{X_i = (x_{i1}, \dots, x_{ij}, \dots, x_{in})}$ où *n* est la dimension du problème, alors chaque source de nourriture est générée par :

$$x_{ij} = X_{min\,j} + rnd(0,1)\left(X_{max\,j} - X_{min\,j}\right), j = 1, \dots, n, \quad i = 1, \dots, N_{FS} \qquad (4)$$

Ces sources de nourriture sont assignées aléatoirement aux $\mathbf{N_O}$ouvrières de la ruche et les valeurs de la fonction objective correspondantes sont évaluées. Nous supposerons ici que $\mathbf{N_O = N_{FS}}$

De même nous supposerons que le nombre d'abeilles spectatrices $\mathbf{N_S}$est égal à$\mathbf{N_O}$.
Enfin chaque source de nourriture dispose d'un compteur d'abandon initialisé à 0.
La $k^{ème}$itération consiste à réaliser successivement les 3 phases décrites ci-dessous.***[III.2]***

## 1. *Description de l'algorithme BLA*

✓ **Phase ouvrière**

Dans cette phase, chaque abeille employée génère une nouvelle source de nourriture$\mathbf{x_{new}}$choisie au voisinage de la position actuellement trouvée :***[III.2]***

$$x_{newij} = x_{ij} + rnd(-1,1)\left(x_{ij} - x_{kj}\right), \quad i = 1, \dots, N_O \qquad (5)$$

Où $\mathbf{k \in \{1, 2, 3, \dots, N_{FS}\}}$ tel que $\mathbf{k \neq i}$ et $\mathbf{j \in \{1, 2, 3, \dots, n\}}$ sont choisis aléatoirement.

On procède ensuite à une sélection après évaluation des fonctions objectives pour cette nouvelle solution. $\mathbf{X_{new}}$remplace $\mathbf{X_i}$dans la population si $f(\mathbf{X_{new}}) \leq f(\mathbf{X_i})$(et son compteur d'abandon est réinitialisé) sinon $\mathbf{X_i}$est conservée et son compteur d'abandon est incrémenté.

- ✓ **Phase spectatrice**

Dans cette étape les spectatrices récupèrent de la part des ouvrières de l'information sur la quantité de nectar de la source $\mathbf{X_i}$. La probabilité $\mathbf{P_i}$ que les spectatrices choisissent la source $\mathbf{X_i}$est déterminée comme suit :

$$P_i = \frac{fit_i}{\sum_{K=1}^{N_{FS}} fit_k} \qquad (6)$$

Où $fit_i$ est la quantité de nectar de la $i^{ème}$ source de nourriture $\mathbf{X_i}$.

On trouve aussi dans la littérature l'expression suivante :

$$P_i = 0,1 + \frac{0,9 fit_i}{\max_k(fit_k)} \qquad (7)$$

La quantité de nectar associée à la source $\mathbf{X_i}$est déterminée par :

$$fit_i = \begin{cases} \frac{1}{1+f(x_i)} & si\ f(x_i) \geq 0 \\ 1 + |f(x_i)| & si\ f(x_i) \leq 0 \end{cases} \qquad (8)$$

D'après (10) et (11), il est clair que plus $fit_i$est élevée, plus la probabilité desélectionner la source $\mathbf{X_i}$correspondante est importante.

Une fois que la $i^{ème}$source est sélectionnée par les spectatrices, elles en réalisent unemodification en utilisant (8).

Si la source ainsi modifiée est meilleure que $\mathbf{X_i}$alors la source modifiée remplace $\mathbf{X_i}$ dans la population (son compteur d'abandon est alors réinitialisé), sinon $\mathbf{X_i}$est conservéet son compteur d'abandon est incrémenté.

- ✓ **Phase scoute**

La $i^{ème}$ source de nourriture est abandonnée si elle ne peut être améliorée au boutd'un certain nombre prédéterminé d'essais $\boldsymbol{T_{limit}}$ (c'est-à-dire si son compteur d'abandondépasse $\boldsymbol{T_{limit}}$ ) et l'abeille ouvrière correspondante devient alors une abeille scoute.L'abeille scoute génère alors sa source de nourriture aléatoirement :

$$\boldsymbol{x_{ij} = X_{min\,j} + rnd(0,1)\left(X_{max\,j} - X_{min\,j}\right), j = 1, \dots, n} \quad (9)$$

## Pseudo-code de la partie recherche de nourriture

Le pseudo code de la partie recherche de nourriture est donné ci-dessous.

**Pour chaque ouvrière**

Générer une solution initiale $\mathbf{X_i}$ pour chaque ouvrière avec (4),
Evaluer la fonction objective $\boldsymbol{f(X_i)}$
Evaluer la quantité de nectar $\boldsymbol{(fit_i)}$ associée en utilisant (8),
Initialiser un compteur d'abandon $\mathbf{C_i = 0}$.
Tant que le critère d'arrêt n'est pas satisfait, répéter :

**Phase ouvrière**

Pour chaque ouvrière
Générer une nouvelle position $\mathbf{X_{new}}$avec (5),
Evaluer la fonction objective correspondante $\boldsymbol{f(X_{new})}$,
Evaluer sa quantité de nectar $\boldsymbol{(fit_{new})}$ en utilisant (8),
Si $\boldsymbol{f(X_{new}) \leq f(X_i)}$, alors
$\mathbf{X_i \leftarrow X_{new}}$
$\mathbf{C_i = 0}$
Sinon
$\mathbf{C_i = C_i + 1}$
Fin Si
Fin Pour
Evaluer la probabilité de transmission d'information associée à chaque source $\mathbf{P_i}$l'aide de (7),

**Figure III.5 :Pseudo-code de la partie recherche de nourriture *[III.2]***

**Phase spectatrice**

Pour chaque spectatrice
Si **rnd (0,1) <$P_i$**
Générer une nouvelle position **$X_{new}$**avec (5),
Evaluer la fonction objective correspondante ***f* ( $X_{new}$ )**,
Evaluer sa quantité de nectar **($fit_{new}$)** en utilisant (8)
Si ***f* ( $X_{new}$)≤ *f*($X_i$)**, alors
**$X_i \leftarrow X_{new}$**
**$C_i$**= 0
Sinon
**$C_i$**= **$C_i$**+1
Fin Si

**Phase scoute**

Pour chaque scoute
Si **$C_i \geq T_{limit}$**
Générer une nouvelle position avec (9),
Evaluer la fonction objective correspondante
Evaluer sa quantité de nectar **($fit_i$)** en utilisant (8),
**$C_i = 0$**
Retenir la meilleure solution **$X_{best}$**
Fin Tant que

**Figure III.6 :Pseudo-code de la partie recherche de nourriture (suit)** ***[III.2]***

## *Variantes de l'algorithme*

Les paramètres de base de l'algorithme sont le nombre de sources de nourriture $N_{FS}$ ,le nombre d'ouvrières $N_O$, le nombre de spectatrices $N_S$ ainsi que la limite $T_{limit}$ à partirde laquelle la source de nourriture doit être abandonnée.

La version standard de l'algorithme consiste à prendre :

$$N_{FS} = N_O = N_S \quad (10)$$

Enfin dans la littérature, il est préconisé de choisir :

$$T_{limit} = n \times N_{FS} \quad (11)$$

## Conclusion

Dans ce chapitre, nous avons exposé la formulation du problème d'ordonnancement et donné des détails sur la métaheuristique BLA.

Dans le chapitre suivant, nous allons tester notre méthode pour vérifier ses performances.

# *Chapitre IV :*

# *Réalisation et Tests*

## Introduction

Dans le chapitre précédent, nous avons présenté la formulation du problème d'ordonnancement de tâches ainsi que sa modélisation avec la méta-heuristique BLA

Dans ce dernier chapitre, nous allons présenter en premier lieu quelques définitions de base, ensuite nous détaillons les résultats et les tests obtenus après l'implémentation et l'exécution de notre algorithme BLA. Enfin, une comparative avec l'algorithme génétique est réalisée afin d'évaluer l'algorithme BLA que vous avons implémenté.

Nous rappelons que nous avons implémenté la version BLA proposée et décrite dans [référence Bitam].

# I. Définitions de base

## I.1 Datacenter (DC)

Un centre de données (en anglais **data center**) est un lieu ou un service sur lequel se trouve regrouper des équipements informatiques (ordinateurs centraux, serveur, baies de stockage, équipements réseaux et de télécommunications….Etc.

Il peut être interne ou externe à l'entreprise, exploité ou non avec le soutien de prestataires. Il Comprend en général un contrôle sur l'environnement (climatisation, système de prévention contre l'incendie, etc.), une alimentation d'urgence et redondante, ainsi qu'une sécurité physique élevée.

Cette infrastructure peut être propre à une entreprise et utilisé par elle seule ou à des fins Commerciales. Ainsi, des particuliers ou des entreprises peuvent venir y stocker leurs données suivant des modalités bien définies. ***[IV.1]***

Un data center c'est le centre de traitement dans le Cloud Computing regroupant des équipements constituants le système d'information de l'entreprise.

Il doit être protégé et prévenu de toutes malveillances au système, tels que le risque d'incendie, une coupure d'électricité ou l'accès de personnes malveillantes au serveur. Comme exemple des Data Center les plus connues dans le monde informatique, nous pourrions citer ceux de Microsoft, Google et Facebook. Les grands Data center sont souvent mis en œuvre pour soutenir et supporter la demande croissante du calcul et du stockage de l'entreprise. De nos jours il y'a environ plus de 23000 Data center.

En 2014, le marché devait croitre à environ 343.4 milliards de dollars. Pour Facebook par exemple, le nombre de nœuds de calculs dans leur Data Center est à 60000. Ces centres de données à grandes échelles consomment généralement une énorme quantité d'énergie électrique qui entraine des coûts opérationnels élevés et des émissions importantes de dioxyde de Carbonne. ***[IV.2]*** Voici une image d'un exemple de Data Center :

**Figure IV.1 : Data center de Facebook *[IV.3]***

## *I.2 La machine virtuelle (VM)*

Le concept de la machine virtuelles remonte aux années 1960, il a été introduit pas IBM comme un moyen pour fournir un accès interactif et simultané à leur ordinateur centraux. Une machine virtuelle (en anglais Virtual Machine) est une instance de la machine physique (PM) qui donne illusion aux utilisateurs d'avoir un accès direct à la machine physique. Les VM sont utilisées pour permettre le partage de ressources d'un matériel très coûteux et chaque VM est une copie protégée et isolée du système. ***[IV.4]***

C'est une machine qui a les mêmes caractéristiques que les machines physiques ou des ordinateurs exécutants un système d'exploitation, une application, ou des ressources pouvant être clonés, modélisés, signifiants et très facilement manipulables. ***[IV.5]***

## I.3 *Cloudlet*(Task)

Les tâches sont le nombre d'instructions et de données connu à exécuter et à transférer. Nous pourrions dire de manière générale que c'est l'enfant d'un Cloud,car c'est un Cloud en miniature, d'une autre façon c'est un centre de donnée à petite échelle conçue pour fournir des services rapides aux nouveaux appareils mobiles tels que les Smartphones, tablettes, etc.

Il a pour objectif principal d'augmenter le temps de réponses et prendre en charge les applications mobiles gourmandes en ressources, tout cela en utilisant la connexion sans fil et a bande passante élevée en hébergeant des ressources de Cloud comme des VM (Machines Virtuelles), pour éliminer les latences de retard dans les réseaux WAN pouvant s'exécuter sur les Cloud traditionnelles. ***[IV.6]***

## I.4 *Host*

Un **ordinateur hôte** est un terme général pour décrire tout ordinateur relié à un réseauinformatique, qu'il fournisse des services à d'autres systèmes ou utilisateurs (serveur informatique, ou « système hôte ») ou soit un simple client.

On utilise aussi le terme de **système hôte** pour désigner le système qui héberge un système virtuel par opposition à celui-ci. Le système virtuel utilise les ressources du système hôte. Cela s'appelle la virtualisation. ***[IV.7]***

## I.5 *Broker*

Un broker (ou courtier) est un intermédiaire entre les clients et les producteurs (de services). Il existe des courtiers en assurance, en immobilier, en vente de véhicule, etc. L'objectif principal de ce type de broker est de proposer des prix abordables à sa clientèle en jouant sur les quantités du côté du producteur. ***[IV.8]***

- ***Le négociateur***

Il peut être mandaté pour négocier des contrats. Dans ce cas, il peut répartir les services entre divers fournisseurs dans un souci d'économie maximale, malgré les éventuelles complexités engendrées par les négociations impliquant plusieurs fournisseurs. Une fois la recherche terminée, le courtier présente au client une liste de fournisseurs recommandés ainsi qu'une comparaison des caractéristiques du service, des ventilations des coûts, et d'autres critères.

- ***L'agrégateur de services***

Il combine et intègre plusieurs services en un ou plusieurs nouveaux services. Il assure l'intégration des données et les connexions sécurisées entre son client et les multiples fournisseurs de Cloud. Il peut choisir les services de plusieurs fournisseurs, selon les critères déterminés auparavant avec son client.

- ***facilitator***

Il peut améliorer un service donné en optimisant certaines capacités spécifiques ou en proposant des services à valeur ajoutée ou personnalisés. Cette amélioration peut concerner la gestion de l'accès aux services Cloud, le contrôle des identités, les rapports de performance, …etc.

Les différentes missions d'un cloud broker s'avèrent très utiles pour les entreprises. Mais un courtier ne peut justifier le surcoût de sa prestation que par la valeur ajoutée qu'il apportera aux services de base des fournisseurs : non seulement il doit faciliter le travail quotidien de ses clients, mais aussi celui de demain !

Une relation de confiance entre le courtier et son client doit donc s'instaurer afin de définir la stratégie de migration la mieux adaptée. Cet accompagnement doit permettre d'optimiser la politique de transformation numérique. L'offre est pléthorique. Sélectionner les bons services et les bonnes plates-formes est une étape délicate, mais déterminante.

## *II. Exemple de simulation BLA*

Nous commençons par un petit exemple d'un résultat de simulation BLA. Nous illustrons les captures les plus importantes afin de montrer les deux parties fondamentales de notre algorithme (la reproduction et la nourriture).

Nous avons effectué cette simulation avec 10 tâches, 10 machines virtuelles, et 3 itérations.

## 1. La population initiale

Dans cet exemple la population initiale est composée de 10 individus (abeilles), dont chacun représente une solution (un ensemble de tâches affectées aux machines virtuelles) générées aléatoirement.

Après la création de la population initiale, les individus sont évalués par la fonction objective afin de les classer selon leur aptitude (reine, drone et ouvrière).

Ces figures montrent le classement de quelquesindividus après l'évaluation :

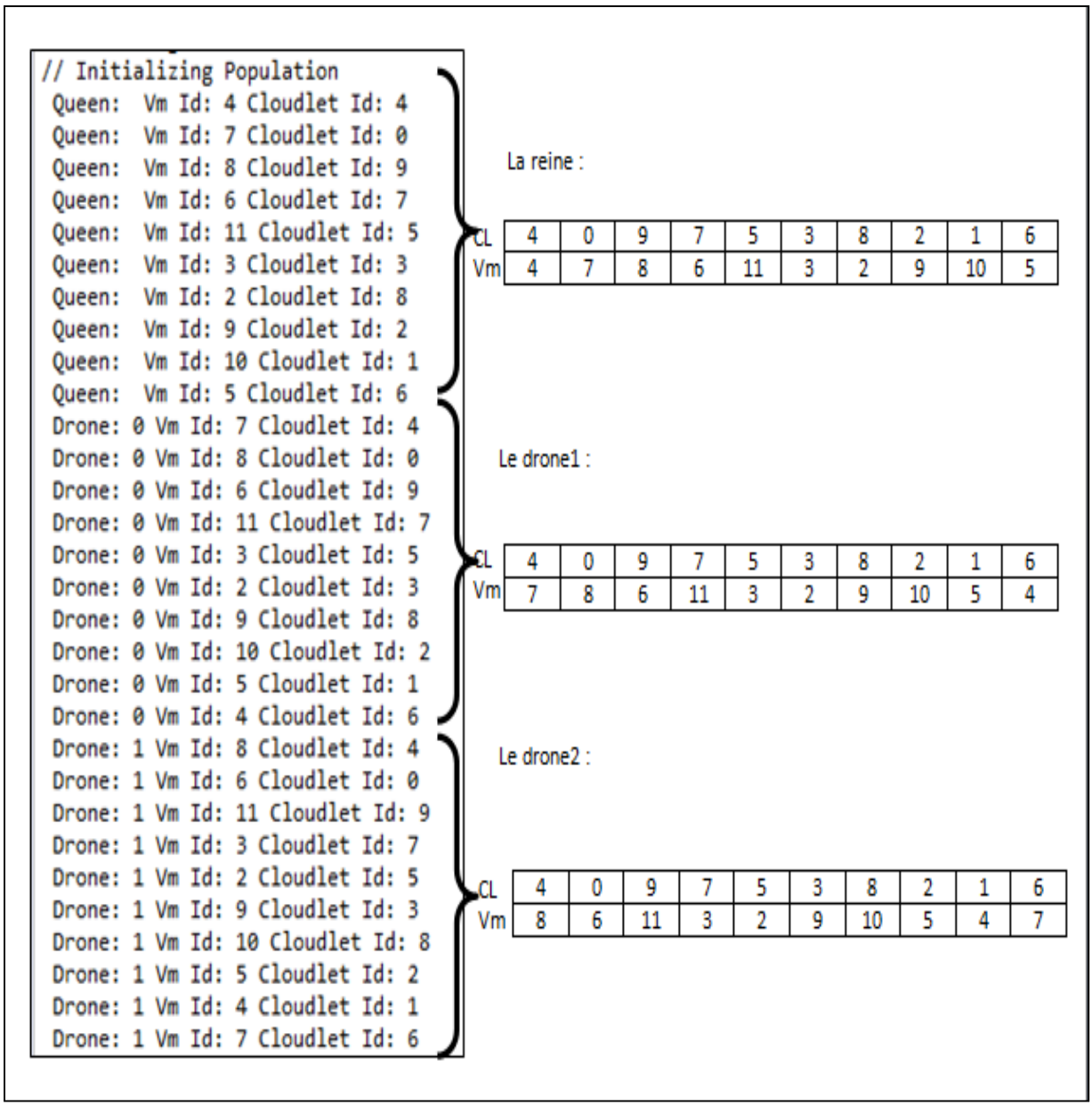



**Figure IV.2 : Exemple du classement de 3 individus**

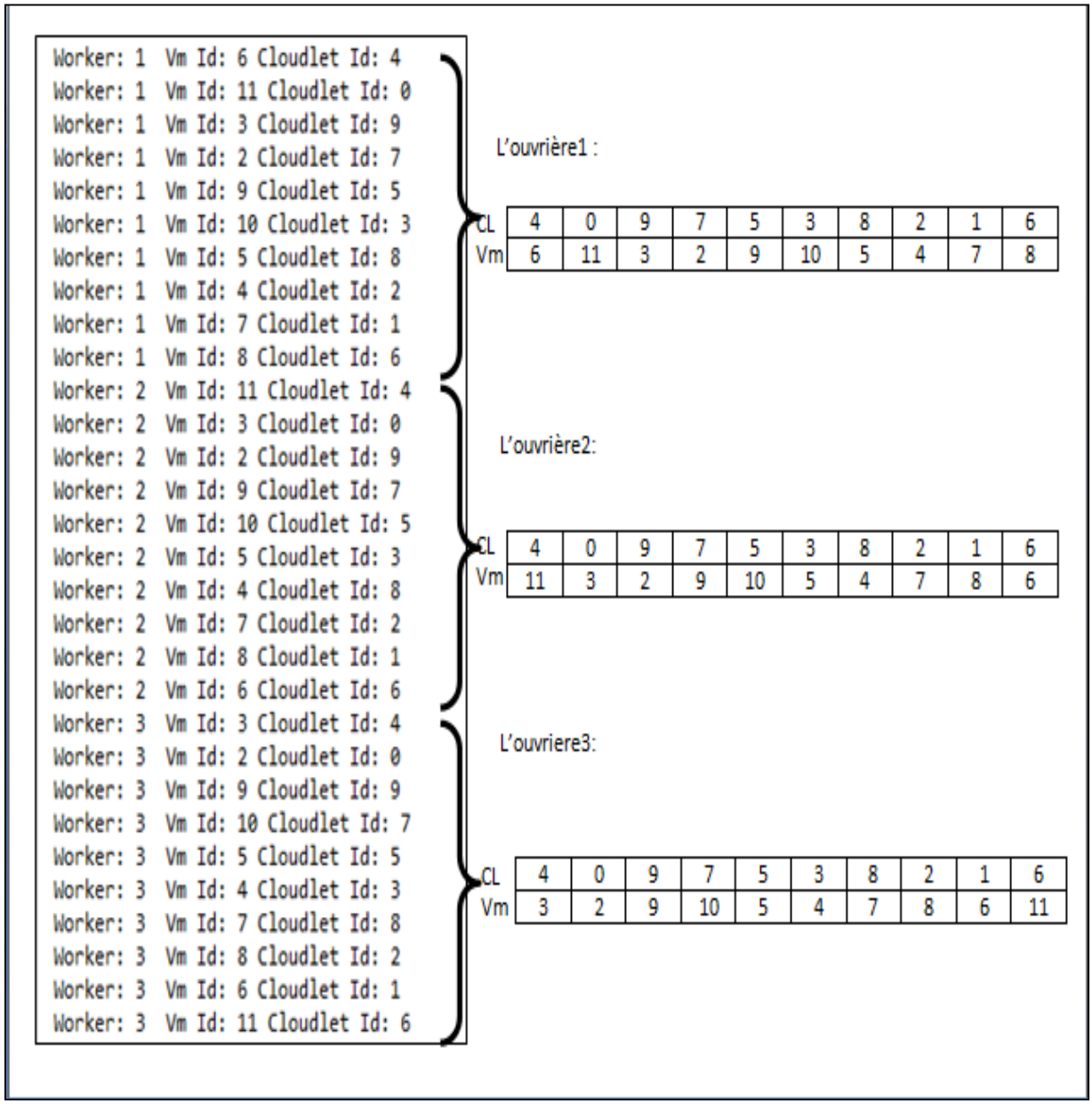



**Figure IV.3 : Exemple du classement de 3 ouvrières**

## 2. *La phase de croisement et mutation*

Apres l'évaluation et l'ordonnancement de la population initiale, la phase de croisement et de mutation débutera, en voici un exemple :

Dans la première itération (itération 0), le croisement s'applique sur la Reine et le Drone0, afin de produire deux progénitures nommées worker 8 et worker 9. La première figure

montre les points de croisement et de mutation que nous avons appliquée à l'accouplement des deux parents, la deuxième figure montre les progénitures résultant de ces parents.

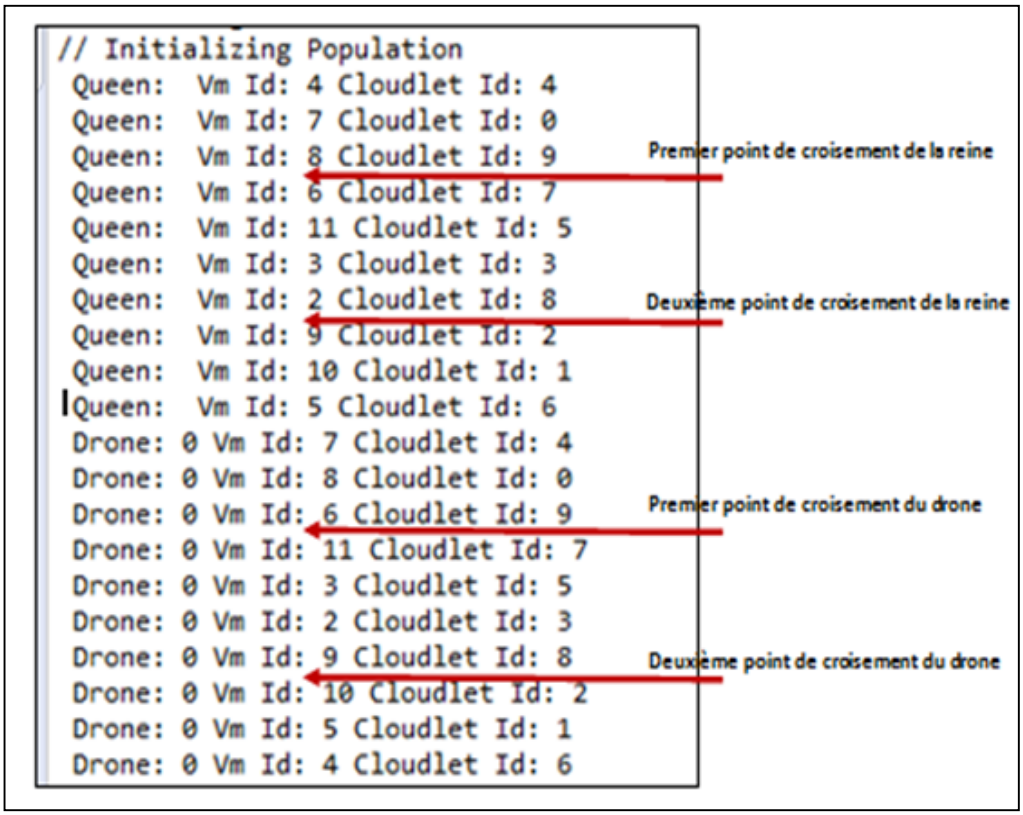



**Figure IV.4 : les points de croisement des deux parents**

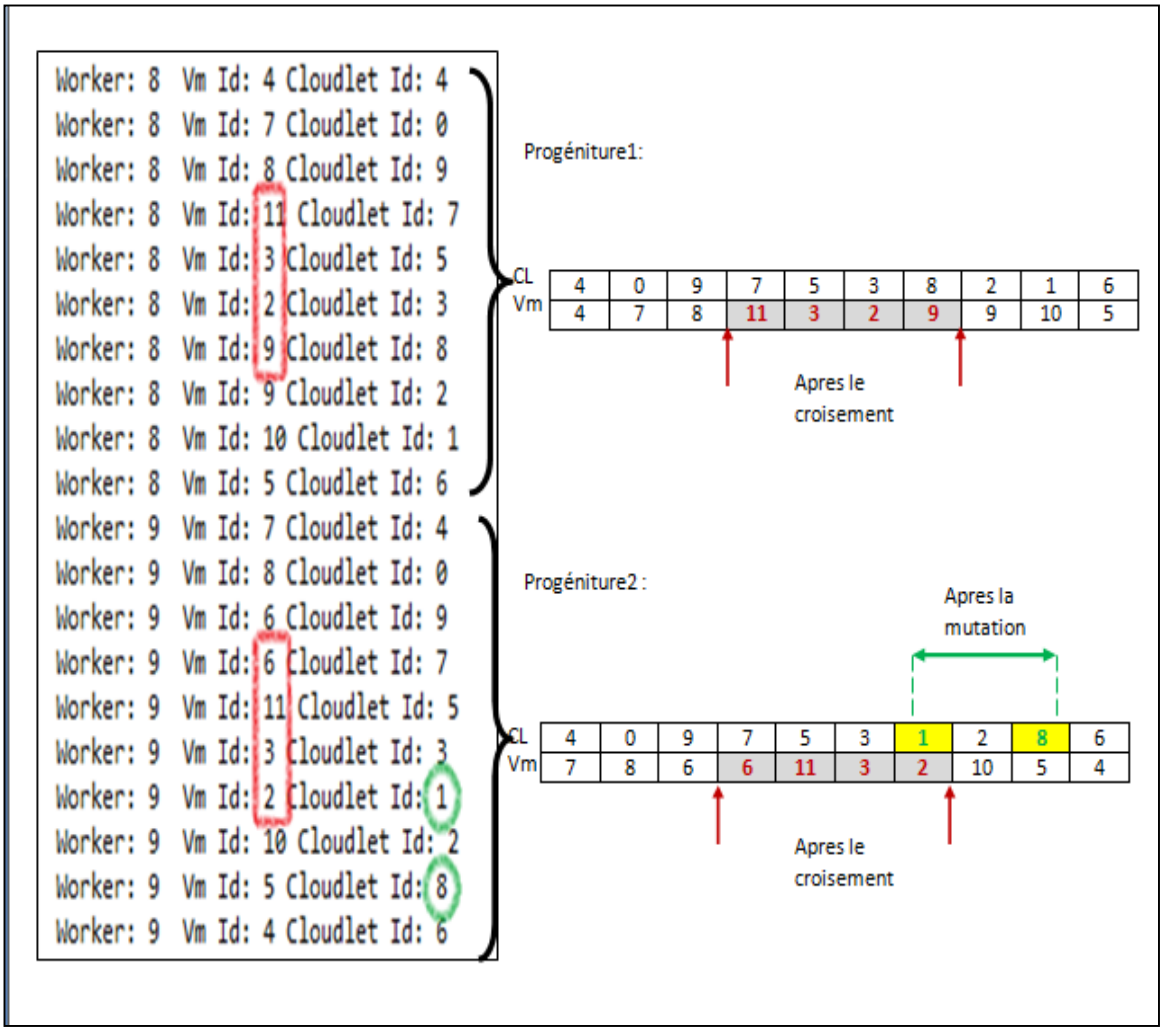



**Figure IV.5 : les progénitures résultant après le croisement**

### 3. *La phase de recherche de nourriture*

La phase de recherche de nourriture consiste à appliquer pour chaque ouvrière (source de nourriture), le calcul de voisinage afin de localiser l'ouvrière la plus proche de cette source (le voisin). Nous précisons que la localisation du voisinage se fait par une autre ouvrière (la spectatrice) choisit aléatoirement.

Apres le calcul du voisinage, nous comparons sa valeur de la fonction objective avec celle de la source de nourriture. Si le résultat de la fonction objectif du voisin est inférieur à celui obtenu par la source de nourriture. Le voisin deviendra une nouvelle source de nourriture et un processus itératif sera appliquer afin de chercher une autre nouvelle source de nourriture jusqu'à atteindre l'optimum.

```
 food source :  Vm Id: 3 Cloudlet Id: 5
 food source :  Vm Id: 7 Cloudlet Id: 6
 food source :  Vm Id: 7 Cloudlet Id: 4
 food source :  Vm Id: 11 Cloudlet Id: 8
 food source :  Vm Id: 11 Cloudlet Id: 7
 food source :  Vm Id: 2 Cloudlet Id: 1
 food source :  Vm Id: 9 Cloudlet Id: 2
 food source :  Vm Id: 9 Cloudlet Id: 9
 food source :  Vm Id: 5 Cloudlet Id: 3
 food source :  Vm Id: 5 Cloudlet Id: 0

 onlooker :  Vm Id: 11 Cloudlet Id: 5
 onlooker :  Vm Id: 10 Cloudlet Id: 6
 onlooker :  Vm Id: 8 Cloudlet Id: 4
 onlooker :  Vm Id: 2 Cloudlet Id: 8
 onlooker :  Vm Id: 9 Cloudlet Id: 7
 onlooker :  Vm Id: 5 Cloudlet Id: 1
 onlooker :  Vm Id: 3 Cloudlet Id: 2
 onlooker :  Vm Id: 6 Cloudlet Id: 9
 onlooker :  Vm Id: 4 Cloudlet Id: 3
 onlooker :  Vm Id: 7 Cloudlet Id: 0
the random value is : -1
calculates new position (neighbour)
 the VM  N°  0  of neighbour is :  11
 the VM  N°  1  of neighbour is :  10
 the VM  N°  2  of neighbour is :  8
 the VM  N°  3  of neighbour is :  2
 the VM  N°  4  of neighbour is :  9
 the VM  N°  5  of neighbour is :  5
 the VM  N°  6  of neighbour is :  3
 the VM  N°  7  of neighbour is :  6
 the VM  N°  8  of neighbour is :  4
 the VM  N°  9  of neighbour is :  7
```

**Figure IV.6 : Exemple decalcul du voisinage**

```
 neighbour :  Vm Id: 11 Cloudlet Id: 5
 neighbour :  Vm Id: 10 Cloudlet Id: 6
 neighbour :  Vm Id: 8 Cloudlet Id: 4
 neighbour :  Vm Id: 2 Cloudlet Id: 8
 neighbour :  Vm Id: 9 Cloudlet Id: 7
 neighbour :  Vm Id: 5 Cloudlet Id: 1
 neighbour :  Vm Id: 3 Cloudlet Id: 2
 neighbour :  Vm Id: 6 Cloudlet Id: 9
 neighbour :  Vm Id: 4 Cloudlet Id: 3
 neighbour :  Vm Id: 7 Cloudlet Id: 0
***** cost function  of food source is :  30.44152386351173
***** cost function  of neighbour is :  26.35843870089392
***** the amount of nectar of the neighbour is :  0.036551793431374636
 the new food source :  Vm Id: 11 Cloudlet Id: 5
 the new food source :  Vm Id: 10 Cloudlet Id: 6
 the new food source :  Vm Id: 8 Cloudlet Id: 4
 the new food source :  Vm Id: 2 Cloudlet Id: 8
 the new food source :  Vm Id: 9 Cloudlet Id: 7
 the new food source :  Vm Id: 5 Cloudlet Id: 1
 the new food source :  Vm Id: 3 Cloudlet Id: 2
 the new food source :  Vm Id: 6 Cloudlet Id: 9
 the new food source :  Vm Id: 4 Cloudlet Id: 3
 the new food source :  Vm Id: 7 Cloudlet Id: 0
```

**Figure IV.7 : la nouvelle source de voisinage**

## 4. *Les derniers résultats de notre algorithme*

Cette figure montre le dernier résultat de notre simulation, à savoir les identifiants des taches, leurs début et fin d'exécution, 10 centres de données, 10 machines virtuelles, le makespan, et flow time.

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID       Time    Start Time    Finish Time
    05         SUCCESS         07           07        01,22        00,1          01,32
    06         SUCCESS         08           08        01,62        00,1          01,72
    04         SUCCESS         06           06        01,85        00,1          01,95
    01         SUCCESS         03           03        01,95        00,1          02,05
    08         SUCCESS         10           10        02,24        00,1          02,34
    02         SUCCESS         04           04        02,82        00,1          02,92
    09         SUCCESS         11           11        03,02        00,1          03,12
    07         SUCCESS         09           09        03,16        00,1          03,26
    00         SUCCESS         02           02        04,56        00,1          04,66
    03         SUCCESS         05           05        05,17        00,1          05,27
Makspan_BAL  :5.267870036101082
flow time BLA : 26.448043268298818
BLA.BLA_Scheduler finished!
```

**Figure IV.8 : Le résultat de la simulation**

## III. Objectif à optimiser

***L'ordonnancement***, est un aspect important pour le fonctionnement du Cloud, nous avons traité dans ce travail le problème d'ordonnancement de tâches ainsi que sa modélisation avec la méta-heuristique BLA en vue de minimiser le makspan et le flow time.
Avant de détailler les résultats et les tests obtenu après l'implémentation, nous allons d'abord définir les objectifs a optimisé.

- ***Makespan***

$$\textbf{Makespan} = \max\{ ExeTime(Ti) \} \quad (3)$$

Nous précisons que dans ce travail, nous avons traité uniquement le cas **des tâches indépendantes.**

- **Somme des temps d'exécution (flow time)**

La somme des temps des exécutions (flow time) est définie comme étant la somme des temps de fin de toutes les tâches du système. Elle est définit comme suit :

$$\textbf{Flow time} = \sum_{T_k^i \in N_i Tasks} ExeTime(T_k^i) \quad (1)$$

Où $ExeTime(T_k^i)$ est le temps d'exécution de la tâche $T_k$ traité dans le nœud $N_i$, calculé par:

$$ExeTime(T_k^i) = \frac{lenght(T_k)}{CPUrate(N_i)} \quad (2)$$

Avec **lenght** ($T_k$) le nombre d'instructions de la tâche $T_k$ et **CPUrate** ($N_i$) le débit de la CPU du nœud $N_i$, qui est déterminé en fonction de la fréquence d'horloge, du nombre de noyaux, du parallélisme, du niveau d'instruction,… etc.

Dans la section suivante, nous allons présenter les paramètres de la simulation, quelques résultats expérimentaux obtenus après la simulation des différentes approches et cela en faisant varier le nombre de tâches.

## III.1 La fonction fitness utilisée :

La fitness appelée aussi fonction objective, mesure la qualité de l'individu exprimée sous forme d'un nombre ou d'un vecteur.

Cette fonction est une fonction de minimisation utilisée pour mesurer l'optimalité de ces deux objectifs: le makespan et le flow time.Elle est définit comme suit :

$$\textbf{Fitness} = W1 * \textbf{Makespan} + W2 * \textbf{flow time} \qquad (3)$$

Où **W1** et **W2** sont des facteurs de pondération fixés pour souligner l'importance de chacun des deux objectifs évalués (le makespan et le flow time).

**Les valeurs des facteurs de pondération choisies :**

**Cas 1** **W1 = W2 = 0,5 :**

$$0{,}5 * \textbf{Makespan} + 0{,}5 * \textbf{flow time} \quad (4)$$

**Cas 2** **W1 = 0 et W2 = 1:**

$$0 * \textbf{Makespan} + 1 * \textbf{flow time} \quad (5)$$

**Cas 3** **W1 = 1 et W2 = 0:**

$$1 * \textbf{Makespan} + 0 * \textbf{flow time} \quad (6)$$

Nous ferons une étude comparative de la fonction fitness décrite dans le tableau ci-dessous, en utilisant les formules (4), (5) et (6) pour différent nombre de taches : 10, 50, 100, 200, 300 ,400 pendant 50 itérations.

| Le nombre de tache | Le nombre de VM | Le nombre d'individu | W1 = W2 = 0,5 | W1 = 0 et W2 = 1 | | W1 = 1 et W2 = 0 | |
|---|---|---|---|---|---|---|---|
| | | | Le résultat de la fonction (4) | Makespan | Le résultat de la fonction (5) | Flow time | Le résultat de la fonction (6) |
| 10 | 3 | 210 | 22.714 | 14.273 | 29.155 | 39.750 | 16.838 |
| 50 | 10 | 250 | 85.211 | 19.065 | 141.358 | 147.869 | 18.961 |
| 100 | 10 | 300 | 161.763 | 36.999 | 282.527 | 288.670 | 39.804 |
| 200 | 20 | 400 | 297.411 | 36.584 | 556.239 | 583.88 | 47.431 |
| 300 | 20 | 500 | 439.366 | 35.589 | 833.144 | 779.333 | 67.238 |
| 400 | 20 | 600 | 614.311 | 79.701 | 1128.921 | 1134.627 | 77.773 |

**TabIV.1 Résultats d'exécution de BLA avec différentes valeurs de pondération de la fonction fitness**

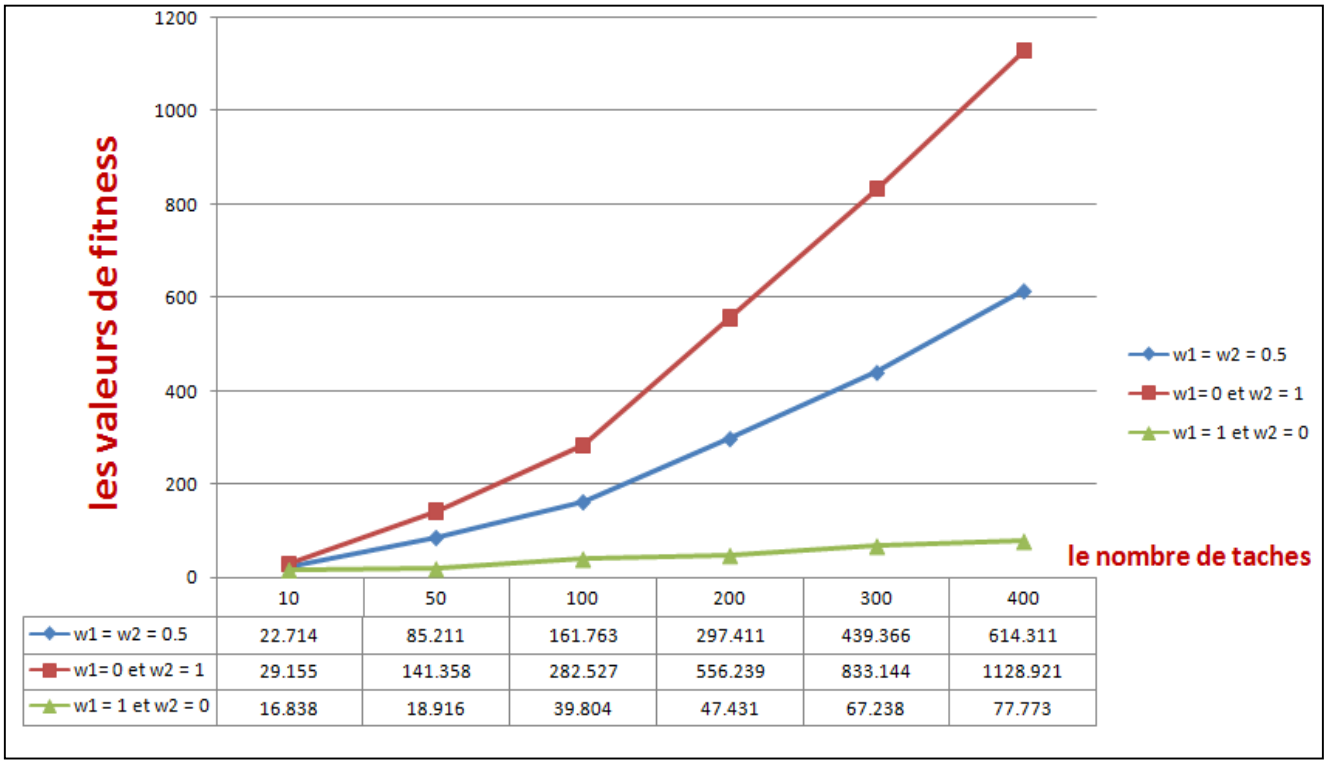



**Figure IV.9 : Courbe représentative des valeurs de la fonction fitness avec différentes valeurs de pondérationdonnées par BLA**

Comme c'est montré dans le tableau (**TabIV.1**) et la figure (**Figure IV.2**), on déduit que le makespan et le flow time augmentent avec l'augmentation du nombre de tâches.

## *III.2 Évaluation des performances*

Nous avons fait varier les valeurs des paramètres de BLA pour choisir les valeurs qui donnent les meilleurs résultats en termes de makespan et flow time afin de montrer le choix que nous avons retenu pour notre étude.

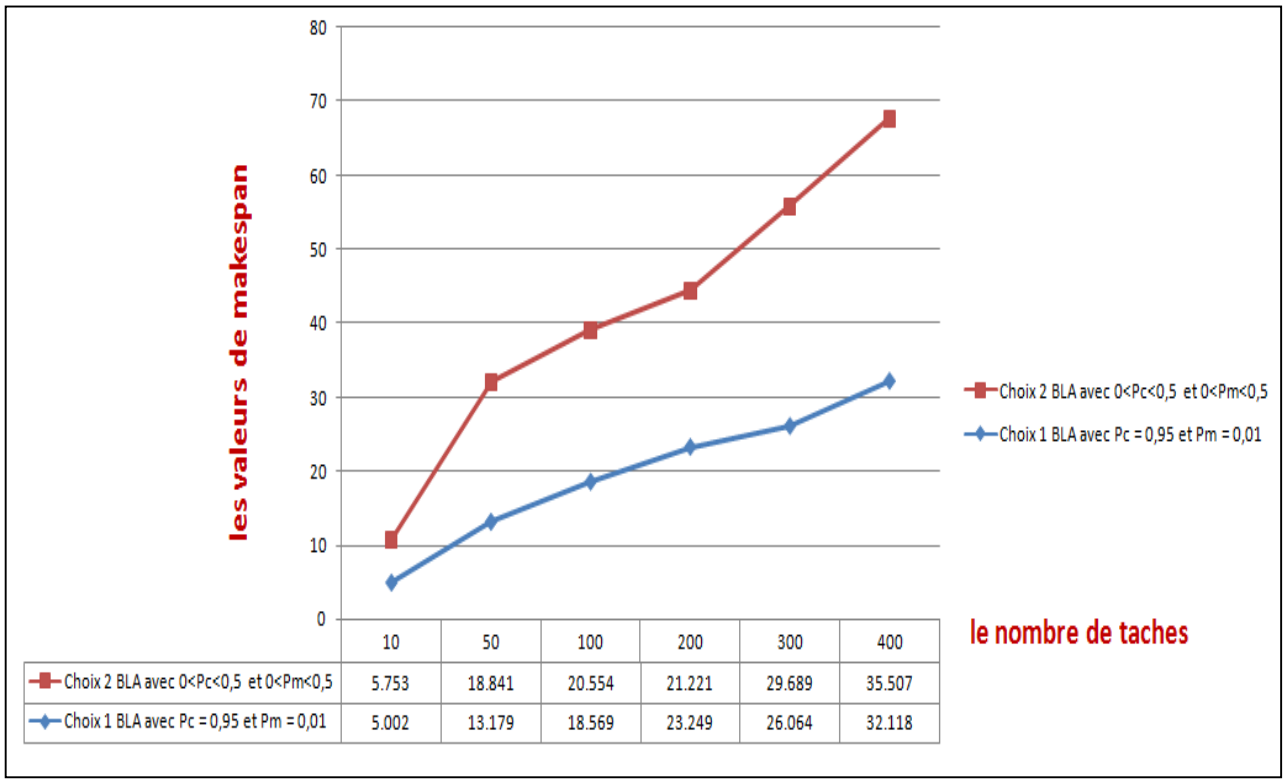
Dans cette étude, nous avons fixé les facteurs de pondération de la fonction fitness w1 et w2 à 1 et 10, respectivement, afin d'obtenir une représentation uniforme des solutions concernant lemakespan et le flow time.

### Réglage des paramètres de BLA

- **Choix 1 :** En tant que valeurs pratiques, les probabilités de croisement et mutation sont choisies 0,95 et 0,01 respectivement.
- **Choix 2 :** les probabilités de croisement et mutation sont des valeurs aléatoires supérieures à 0 et inférieures à 0,5.

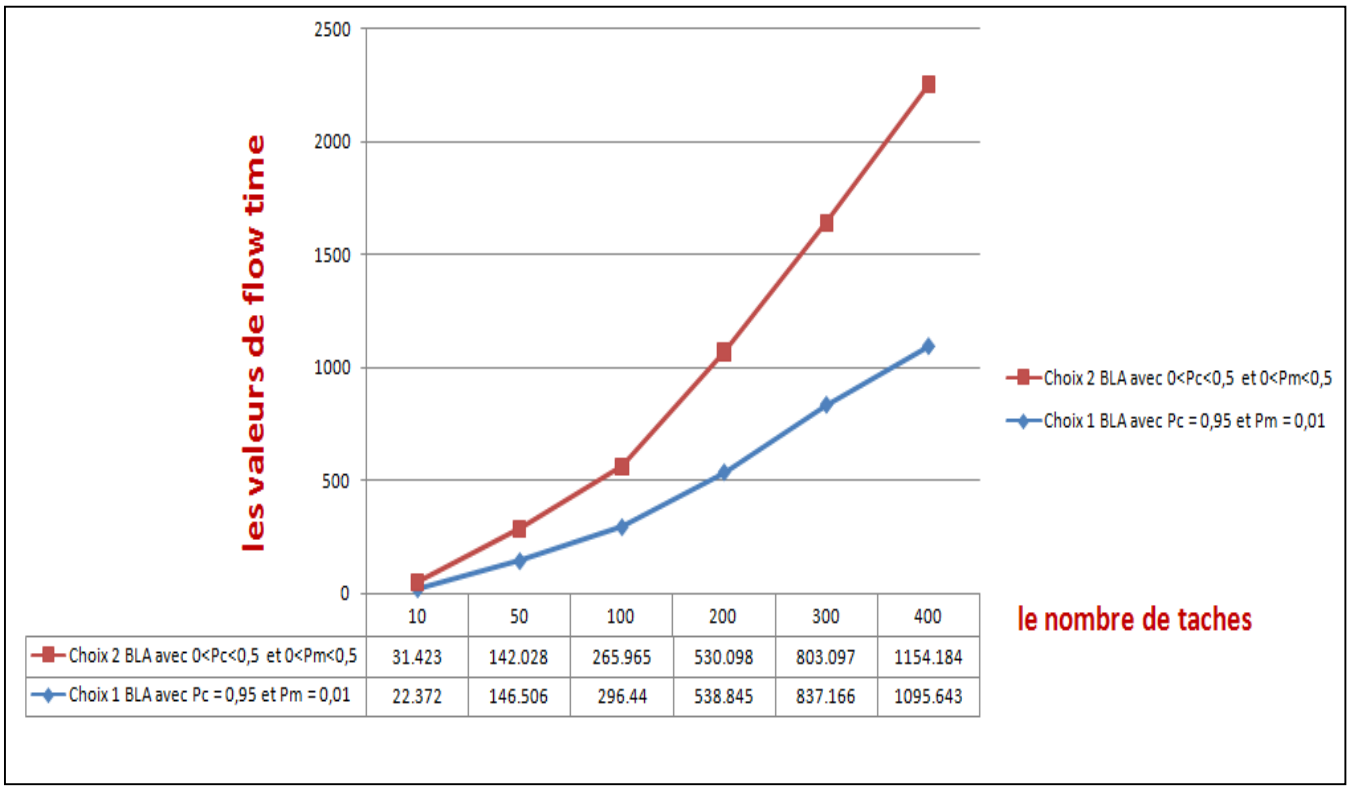
  Les courbes suivantes traduisent les résultats obtenus durant les simulations des deux configurations effectuées sur BLA durant 50 itérations et pour bien démontrer les différences entre les deux choix on a effectué une courbe spécifique pour chaque objectif de sorte que : la première représente les résultats du makespan, la deuxième représente les résultats du flow time.

  Nous réalisons ces tests avec un nombre différent de tâches et de machines virtuelles : 10, 100, 200, 300, 400, 500 tâches et 10, 20, 30, 40, 50, 50 machines virtuelles, respectivement.




| | 10 | 50 | 100 | 200 | 300 | 400 |
|---|---|---|---|---|---|---|
| Choix 2 BLA avec 0<Pc<0,5 et 0<Pm<0,5 | 5.753 | 18.841 | 20.554 | 21.221 | 29.689 | 35.507 |
| Choix 1 BLA avec Pc = 0,95 et Pm = 0,01 | 5.002 | 13.179 | 18.569 | 23.249 | 26.064 | 32.118 |

**Figure IV.10: Courbe représentative de Makespan donné par BLA en fonction de nombre de tâches avec deux choix de paramètres de BLA**.




| | 10 | 50 | 100 | 200 | 300 | 400 |
|---|---|---|---|---|---|---|
| Choix 2 BLA avec 0<Pc<0,5 et 0<Pm<0,5 | 31.423 | 142.028 | 265.965 | 530.098 | 803.097 | 1154.184 |
| Choix 1 BLA avec Pc = 0,95 et Pm = 0,01 | 22.372 | 146.506 | 296.44 | 538.845 | 837.166 | 1095.643 |

**Figure IV.11 : Courbe représentative du flow time donnée par BLA en fonction de nombre de tâches avec deux choix de paramètres de BLA.**

Nous constatons que les résultats du makespan, et du flow time par l'exécution de BLA avec les valeurs de **Pc = 0,95 et Pm = 0,01 (choix 1)** est toujours inferieurs par rapport à ceux obtenues par l'exécution de BLA avec les valeurs **0 <Pc < 0,5 et 0 <Pm < 0,5 (choix 2).**
D'où on peut dire que sur nos exemples, les résultats obtenus de BLA avec les probabilités de croisement et mutation choisies **0,95 et 0,01** respectivement sont meilleurs que les résultats obtenus de BLA avec les probabilités de croisement et mutation supérieur à 0 et inférieures à 0,5 choisies aléatoirement.

### *III.2.1 Etude comparative avecla métaheuristique GA*

Sur la base des mesures ci-dessus, nous comparons les résultats obtenus par l'algorithme BLA et ceux obtenus par l'algorithme génétique traditionnel (GA) présenté dans le chapitre II.

**Les paramètres utilisés pour cette étude :**

- **Le nombre de tâches :** Effectuer pour chaque résultat et un nombre différent de taches : 10, 50, 100, 200, 300, 400 et un nombre de machines virtuelles 3, 10, 10, 20, 20, 20 respectivement.
- **Le nombre d'itération :** Effectuer pour chaque résultat 100 itérations.
- **Les probabilités de croisement et de mutation :** Pc = 0,95 et Pm =0,01 utilisé pour les algorithmes BLA et GA.

❖ Le tableau suivant montre les résultats de la fitness avec les facteurs de pondérations W1 = 0,5 et W2 = 0,5.

| **Nombre de tache** | **10** | **50** | **100** | **200** | **300** | **400** |
|---|---|---|---|---|---|---|
| **Nombre d'individu** | **410** | **450** | **500** | **600** | **700** | **800** |
| **BLA** | 14.845 | 14.687 | 143.650 | 275.361 | 388.951 | 546.102 |
| **AG** | 22.586 | 83.451 | 177.822 | 182.941 | 397.840 | 884.321 |

**TabIV.2 les résultats de la fitness avec les facteurs de pondérations W1 = 0,5 et W2 = 0,5.**

**Analyse 1 :**Comme on peut le voir dans le tableau ci-dessus qu'en variant le nombre de taches et d'individu, les valeurs de la fitness avec les facteurs de pondérations W1 = 0,5 et W2 = 0,5 de BLA sont plus petits par rapport à ceux obtenus par GA.

- Le tableau suivant montre les résultats de la fitness avec les facteurs de pondérations W1 = 0 et W2 = 1.

| Nombre de tache | 10 | 50 | 100 | 200 | 300 | 400 |
|---|---|---|---|---|---|---|
| Nombre d'individu | 410 | 450 | 500 | 600 | 700 | 800 |
| BLA | 23.095 | 129.316 | 257.609 | 444.738 | 714.963 | 1006.052 |
| GA | 28.844 | 131.710 | 293.053 | 598.012 | 852.003 | 1503.662 |

**TabIV.3 les résultats de la fitness avec les facteurs de pondérations W1 = 0et W2 = 1.**

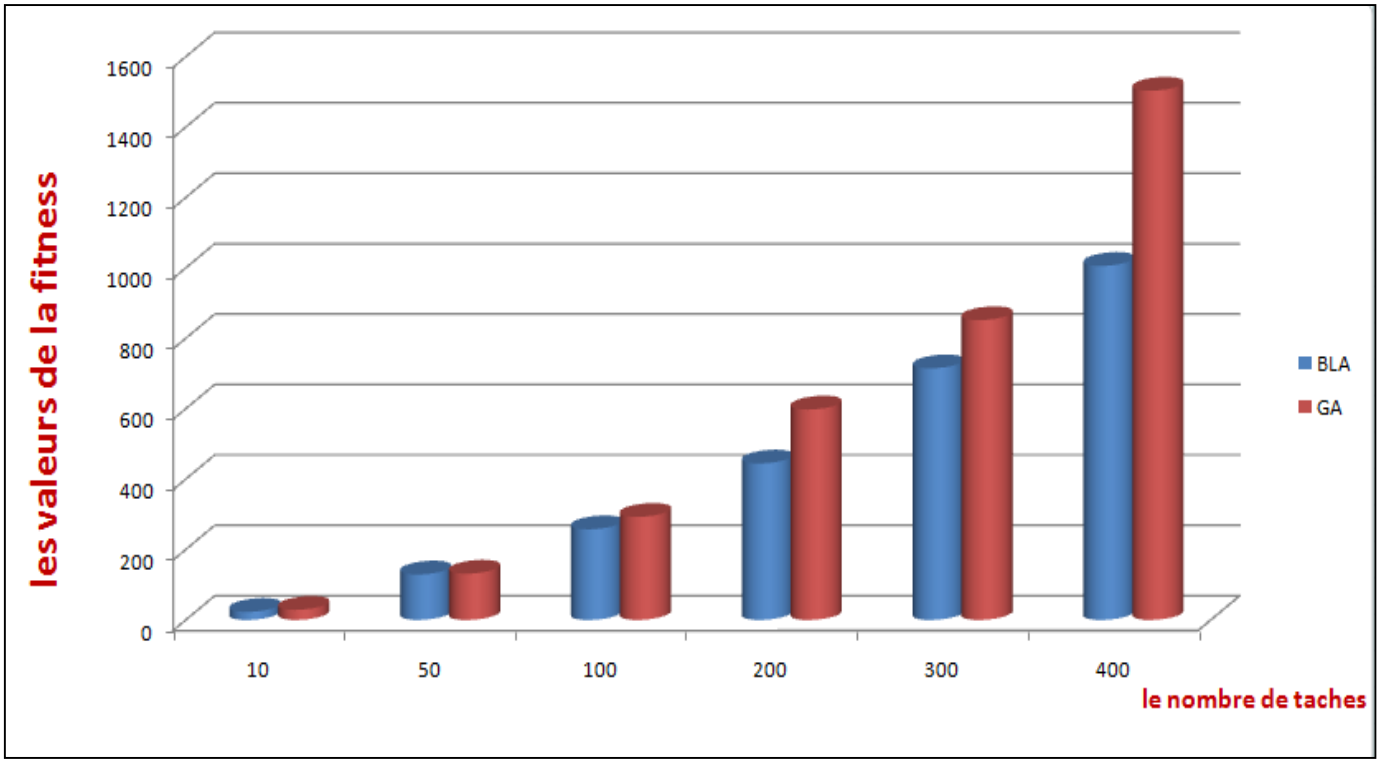



**Figure IV.12 Histogramme représentatif des résultats la fitness avec les facteurs de pondérations W1 = 0et W2 = 1 selon les algorithmes GA et BLA.**

**Analyse 2 :**A base des résultats obtenus on remarque bien que les valeurs de la fitness avec les facteurs de pondérations W1 = 0 et W2 = 1générées au niveau de chaque exécution pour chacun des deux algorithmes exécutés sont différentes d'une exécution à une autre et aussi

d'un algorithme à un autre.De plus on remarque que l'algorithme BLA a généré les minimums de ces résultats par rapport à l'algorithme GA.

- Le tableau suivant montre les résultats de la fitness avec les facteurs de pondérations W1 = 1 et W2 = 0.

| **Nombre de tache** | 10 | 50 | 100 | 200 | 300 | 400 |
|---|---|---|---|---|---|---|
| **Nombre d'individu** | 410 | 450 | 500 | 600 | 700 | 800 |
| **BLA** | 5.029 | 19.971 | 39.456 | 42.433 | 61.789 | 80.177 |
| **AG** | 12.328 | 23.172 | 50.591 | 74.870 | 116.438 | 132.732 |

**TabIV.4 les résultats de la fitness avec les facteurs de pondérations W1 = 1 et W2 = 0.**

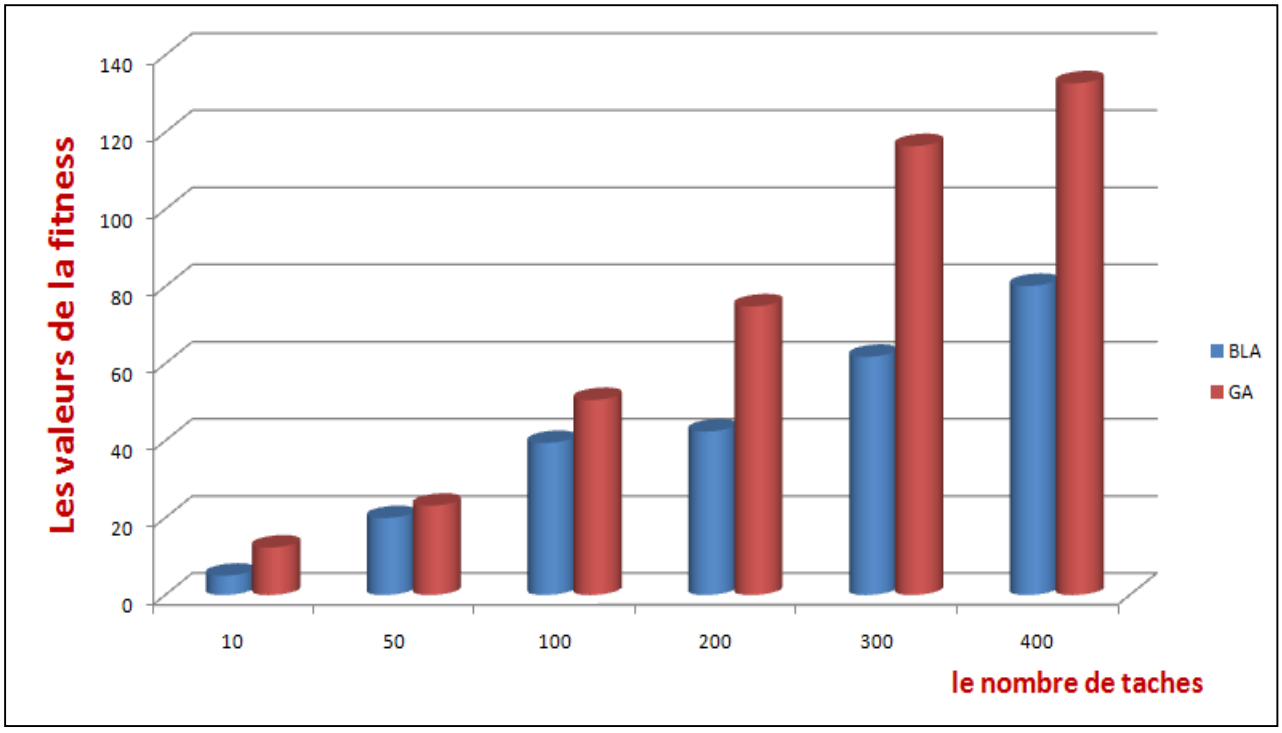



**Figure IV.13 Histogramme représentatif des résultats de la fitness avec les facteurs de pondérations W1 = 1 et W2 = 0selon les algorithmes GA et BLA.**

**Analyse 3 :** même chose pourla fitness avec les facteurs de pondérations W1 = 1 et W2 = 0générées. L'algorithme BLA àgénéré le minimum de ces résultats par rapport à l'algorithme GA.

**Synthèse :**

Afin de prouver la performance de l'algorithme BLA, nous avons analysé et comparé entre les différents résultats obtenu par l'algorithme BLA et l'algorithme AG en tant que algorithme évolutionnaire dans le contexte des algorithmes inspiré par la nature. Les résultats obtenus de BLA prouvent l'efficacité et la performance en termes de makespan et de la somme totale des temps d'exécution (flow time).

## *Conclusion*

Au cours de cette dernière étape de notre travail, nous avons présenté les étapes suivies pour la résolution du problème d'ordonnancement dans le Cloud, nous avons testé un algorithme bi-inspiré qui est l'algorithme BLA en prenant en considération les objectifs : makespan et flow time.

# *Conclusion générale*

# *Conclusion Générale* :

Le problème traité dans ce travail est le problème d'ordonnancement de tâches dans le Cloud Computing en vue de minimiser le makespan et le flow time.

L'apparition de la métaheuristique appelée « BLA » a sollicité notre attention. C'est une nouvelle métaheuristique qui a enrichi le nombre de métaheuristiques basées sur la vie des abeilles. Nous avons implémenté cet algorithme pour résoudre le problème d'ordonnancement de tâches dans le Cloud Computing.

Afin de tester la performance de notre algorithme, nous avons effectué une série de tests de simulation, une étude comparative a été réalisée avec l'algorithme BLA et l'algorithme génétique GA pour une étude de performance.

Comme dans tous les travaux de recherche, il est certain que les solutions proposées ne sont pas les meilleures d'une manière absolues, plusieurs améliorations sont envisageables et plusieurs points peuvent être étudiés dans le cadre de travaux futurs, tels que :

- Prise en compte de nouveaux critères à optimiser ou de nouvelles contraintes pouvant rendre le problème étudié plus réaliste comme : la taille de la mémoire allouée qui est définie comme la quantité total de mémoire d'une ressource utilisé à l'exécution d'une tâche donnée.

- Se concentrer sur la phase scoute que nous n'avions pu réaliser dans la partie recherche de nourriture.

- Prise en compte de nombre de drones qui peut influencer la phase de croisement et réaliser des tests avec un nombre de drones différent pour pouvoir fixer un nombre optimal.

# RÉFÉRENCE BIBLIOGRAPHIQUE


[I.1] : https://www.frameip.com/cloud-computing-bonnes-pratiques/
[1.1] : https://lewebpedagogique.com/presencesenligne/tag/cloud-computing/
[I.2] : le Cloud computing une nouvelle filière fortement structurante, septembre 2012

[I.3] : Audrey Coutty, "Thierry Vonfelt L'évolution maitrisée vers le IaaS/PaaS". Livre blanc produit par EuroCloud France, Novembre 2011.

[1.3] : https://blog.blaisethirard.com/qu-est-ce-que-le-cloud-computing/
[1.4] : https://www.globalsp.com/private-cloud-public-cloud-et-hybrid-cloud/
[1.5] : http://itedgenews.ng/2019/05/23/strategising-successful-iaas-deployment-enterprise/
[1.6] : https://www.developpez.com/actu/249022/Pres-de-la-moitie-des-offres-de-PaaS-Platform-as-a-Service-sont-desormais-exclusivement-basees-sur-le-cloud-d-apres-un-rapport-de-Gartner/

[I.4] : https://www.lebigdata.fr/definition-saas
[I.5] : https://www.hebergeurcloud.com/composants-infrastructure-cloud/
[I.6] : https://fr.wikipedia.org/wiki/Virtualisation
[I.7] : https://www.ivision.fr/cloud-et-virtualisation-les-differences/
[I.8] : http://dspace.univ-tlemcen.dz/bitstream/112/13456/1/Etude-de-la-consommation-denergie-dans-le-Cloud-Computing..pdf
[I.9] : https://support.cloudwatt.com/kb/faq/lecloud/quels-sont-les-avantages-du-cloud-computing.html
[I.10] : https://hal.archives-ouvertes.fr/tel-01275626/document
[I.11] : https://www.renaudvenet.com/cloud-computing-avantages-et-inconvenients-2011-01-26.html
[I.12] : https://www.techopedia.com/definition/30255/fog-computing

[I.13] : [https://www.rfwireless-world.com/Terminology/Fog-Computing-Architecture-working.html]
[I.14] : https://fr.wikipedia.org/wiki/Fog_computing

[I.15] : https://www.forbes.com/sites/janakirammsv/2016/04/18/is-fog-computing-the-next-big-thing-in-internet-of-things/2/#1971ac3a34c9
[I.16] : https://www.ionos.fr/digitalguide/serveur/know-how/fog-computing/

[I.17] : https://pdfs.semanticscholar.org/2f7e/41e41764f97cff39503118d0c2f94fd75e77.pdf
[I.18] : https://gblogs.cisco.com/fr/datacenter/50-milliards-dobjets-connectes-en-2020/
[I.19] : https://www.ionos.fr/digitalguide/serveur/know-how/fog-computing/
[I.20] : https://pdfs.semanticscholar.org/2f7e/41e41764f97cff39503118d0c2f94fd75e77.pdf
[I.21] : https://www.sam-solutions.com/blog/fog-computing-vs-cloud-computing-for-iot-projects/
[I.22] : J. CARLIER et P. CHRÉTIENNE, Les Problèmes d'ordonnancement : un domaine très ouvert, RAIRO-Oper. Res., août 1982, p. 175-217.

[I.23] : Lyes Bouali, Belkadi H, « hybridation d'euristiques pour le problème d'ordonnancement dans les grilles de calcul, » Memoir master, tizi-Ouzou, 2014.

[I.24] : D. E. Insaf, optimisation d'ordonnancement et d'allocation de ressources dans le cloud computing, thèse de doctorat, 2016.

[I.25] : Chengbin Chu. Nouvelles approches analytiques et concept de mémoire artificielle pour divers problèmes d'ordonnancement. PhD thesis, Université de Metz, 1990.

[I.26] : https://pmb.univ-saida.dz/butecopac/doc_num.php?explnum_id=573

[I.27] : https://theses.univ-oran1.dz/document/15201208t.pdf

[I.28] : B. Dyhia. M. Kahina « ordonnancement de workflows dans le cloud » Memoir master, tizi-Ouzou, 2017

[I.29] : https://fr.wikipedia.org/wiki/Chemin_critique

[II.1]:Omran M. G. H. "Particle Swarm Optimization Methods for Pattern Recognition and Image Processing", PHD Thesis, University of Pretoria, Pretoria, South Africa, 2004.

[II.2]: Simon D., "Evolutionary optimization algorithms: Biologically-Inspired and Population-Based Approaches to Computer Intelligence", John Wiley &Sons, Inc., Hoboken, New Jersey, 2013

[II.3] : R. Meziane., «Optimisation de la structure d'un réseau de production d'énergie électrique et amélioration de sa performance », thèse de doctorat, USTO 2007.

[II.4] : O. Roux, «La mémoire dans les algorithmes à colonie de fourmis : applications à l'optimisation et à la programmation automatique », thèse de doctorat de l'Université du Littoral Cote d'Opale, 2001.

[II.23] :https://publications.polymtl.ca/2716/1/2017_MohamedMouradMamlouk.pdf

[II.5]:Darwin Charles. On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. Murray, London, 1859.

[II.6] : AnneSpalanzani. "Algorithmes évolutionnaires pour l'étude de la robustesse des systèmes de reconnaissance automatique de la parole". Thèse de Doctorat, Université Joseph Fourier, Grenoble I, 1999.

[II.24] :http://elearning.univjijel.dz/elearning/pluginfile.php/1271/mod_resource/content/1/Polycopie_ali_2014.pdf

[II.7] : VincentMagnin. "Contribution a l'étude et à l'optimisation de composants optoélectroniques". Thèse de Doctorat, Université Des Sciences et Technologies, Lille, 1998.

[II.8] :Sana Ben Hamida. "Algorithmes évolutionnaires, Prise en compte des contraintes et application réelle", Thèse de Doctorat, Université, 2001.

[II.9] :MarcSchoenauer. "Les algorithmes évolutionnaires : état de l'art et enjeux". AlgorithmsSeminair 2001-2002. Projet Fractales, INRIA Rocquencourt, 2001.

[II.10] : David E.Goldberg. "Algorithmes génétiques Exploitation, optimisation et apprentissage automatique". Addison-Wesley, France, SA, 1994.

[II.11] : Blaise Madeline. "Algorithmes évolutionnaires et résolution de problèmes de satisfaction de contraintes en domaines finis". Thèse de Doctorat en sciences, Université de Nice-Sophia antipolis, 2002.

[II.12] : B.SafiaF.Hamid « Implémentation d'une méthode de Classification par les algorithmes génétiques » memoir Tizi-Ouzou 2008/2009

[II.25] :http://www.univ-bejaia.dz/dspace/bitstream/handle/123456789/8269/Etude%20comparative%20des%20algorithmes%20g%C3%A9n%C3%A9tiques%20multi-objectifs.pdf?sequence=1&isAllowed=y

[II.13]:Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesle: Reading, MA, 1989.

[II.14]:http://deptinfo.unice.fr/twiki/pub/Minfo04/IaDecision0405/Prsentationdesalgorithmesgntiquesetdeleursapplicationsenconomie_P.pdf

[II.26]: http://www.univ-constantine2.dz/files/Theses/Informatique/Doctorat/Amira-Gherboudj.pdf

[II.15] **:** JB. Gotteland, optimisation du trafic au sol sur les grands aéroports, Thèse de doctorat de l'institut national polytechnique de Toulouse, novembre 2004.

[II.16] : B.SafiaF.Hamid « Implémentation d'une méthode de Classification par les algorithmes génétiques » memoir Tizi-Ouzou 2008/2009

[II.17]**:** PAPA ALDEMBA FAYE, „Couplage algorithme génétique-code éléments finis pour le dimentionnement de structures en matériux composites", thèse doctorat,N° 1497,année 2004.

[II.18]**:** Bitam, S., A. Mellouk, and S. Zeadally. 2015. "Bio-Inspired Routing Algorithms Surveyfor Vehicular Ad Hoc Networks."IEEE Communications Surveys & Tutorials 17 (2): 843 867. doi:10.1109/COMST.2014.2371828.

[II.19]**:** H. YAHYA, *Book "The Miracle of the honeybee"*, G. M. D. Cd., Ed. OkmeydaniIstanbul-Turkey, March 2007.

[II.20]**:** harunyahya. http://jlpetitlaurent.free.fr/abeille/http://lerucherdulac.free.fr.

[II.21]:http://193.194.86.69/MMAGISTER/images/facultes/TEC/2012/Souhil%20MOUASSA.pdf

[II.22]: Salim Bitam, SheraliZeadally& Abdelhamid Mellouk(2018)Fogcomputing job scheduling optimization based on beesswarm, Enterprise Information Systems,12:4,373-397,DOI: 10.1080/17517575.2017.1304579

[III.1] : Salim Bitam, SheraliZeadally& Abdelhamid Mellouk(2018)Fogcomputing job scheduling optimization based on beesswarm, Enterprise Information Systems,12:4,373-397,DOI: 10.1080/17517575.2017.1304579

[III.2] : Philippe Feyel.Optimisation des correcteurs par les métaheuristiques. Applica-tion à la stabilisation inertielle de ligne de visée. PhD thesis, Supélec, 2015

[IV.1] : https://fr.wikipedia.org/wiki/Centre_de_donn%C3%A9es

[IV.2] : DadDjouhra. « Optimisation des performances des Datacenter des Cloud sous Contrainte d'énergie consomme », Thèse de Doctorat, 2016.

[IV.3] **:** https://www.datacenterknowledge.com/archives/2016/05/24/inside-facebook-data-center-sweden-video

[IV.4] : https://fr.wikipedia.org/wiki/Machine_virtuelle

[IV.5] : Professeur Jong-Moon Chung. « Emerging Technologies : From Smartphones to loT To Big Data-Cloud Computing &Vm introduction »,consulter le 17/06/2018

[IV.6] : https://en.wikipedia.org/wiki/Cloudlet

[IV.7] : https://fr.wikipedia.org/wiki/H%C3%B4te_(informatique)

[IV.8] : https://www.fimarkets.com/pages/brokers.php

[IV.9] : https://www.scalair.fr/blog/cloud-broker-definition

[A.1] : http://dictionnaire.sensagent.leparisien.fr/Java%20(langage)/fr-fr/

[A.2] : https://www.opensourcemacsoftware.org/developpement/eclipse-ide-java-mac.html

[A.3] : https://graal.ens-lyon.fr/~ecaron/m2rts/2015/blogbazm/

[A.4] : https://www.techno-science.net/definition/517.html

## 1. *langage de programmation Java :[A.1]*

Le langage **Java** est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au *SunWorld*.

- Peut-être décrit en quelques mots : Orienté Objet, Simple, Robuste, Dynamique et Sécurisé, Indépendant de la Plateforme (VM), Semi Compilé/Semi Interprété, Bibliothèque Importante (JDK API).

Parmi les différentescaractéristiquesquisontattribuéesàsonsuccès,nousavons[lan08] :

- L'indépendance de toute plate-forme : le code reste indépendant de la machine sur la Quelle ilsexécuté. Ilestpossibled'exécuterdesprogrammesJavasurtouslesenviron Nuementquipossèdentune JavaVirtualMachine.

- Java est également portable, permettantàlasimulationd'êtredistribuéefacilement Sans avoiràrecompilerlecodepourlesdifférentssystèmes.

- Le codeeststructurédansplusieursclassesdontchacunetraiteunepartiedifférente De lasimulation.

- Il assurelagestiondynamiquedelamémoire.

- Javaestmultitâches:ilpermetl'utilisationdeThreadsquisontdesunitésd'exécution Isolées.

Se trouve sous 3 environnements d'exécutions différents

- Java ME (Micro Edition) pour PDA, téléphone
- Java SE (Standard Edition) pour desktop
- Java EE (Entreprise Edition) pour serveur Servlet/JSP/JSTL, Portlet/JSF, JTA/JTS, JDO/EJB JavaMail, etc.
- Est OpenSource depuis novembre 2006 (2008 complètement)
- Toutes les sources sont disponibles : http://www.openjdk.org
- Il assure la portabilité entre différents environnements (machine/OS)

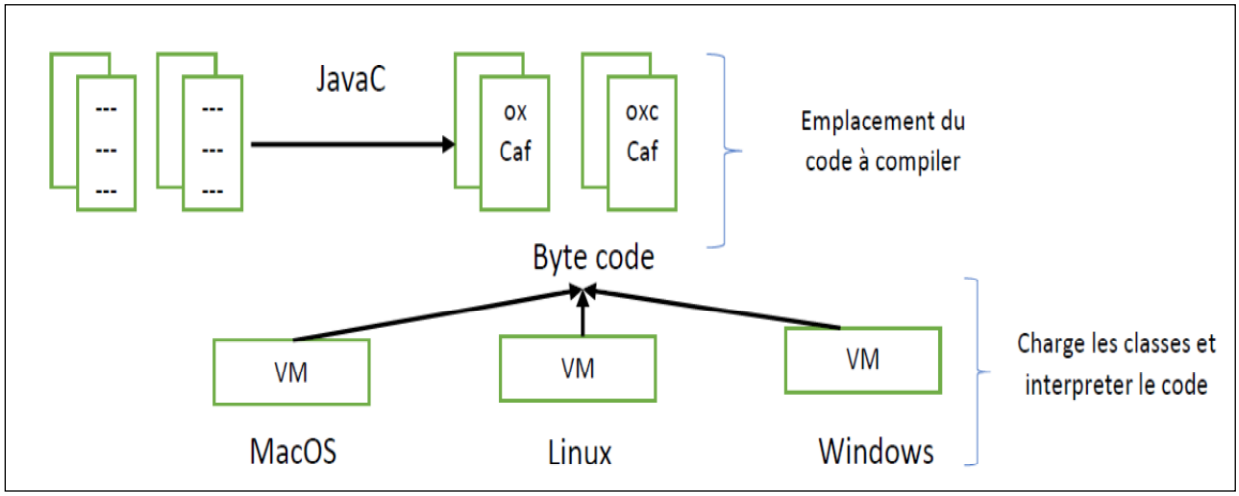



**Figure A.1 : La portabilité du langage java.** ***[A.1]***

## 2. *L'environnement de développement éclipse*

**Figure A.2 : Logo Eclipse** ***[A.2]***

**Eclipse IDE** est un environnement de développement intégré libre (le terme *Eclipse* désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plug-in (en conformité avec la normeOSGi) : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in.Plusieurs logiciels commerciaux sont basés sur ce logiciel libre, comme par exemple IBM Lotus Notes 8, IBM Symphony ou Websphere Studio Application Developer. ***[A.2]***

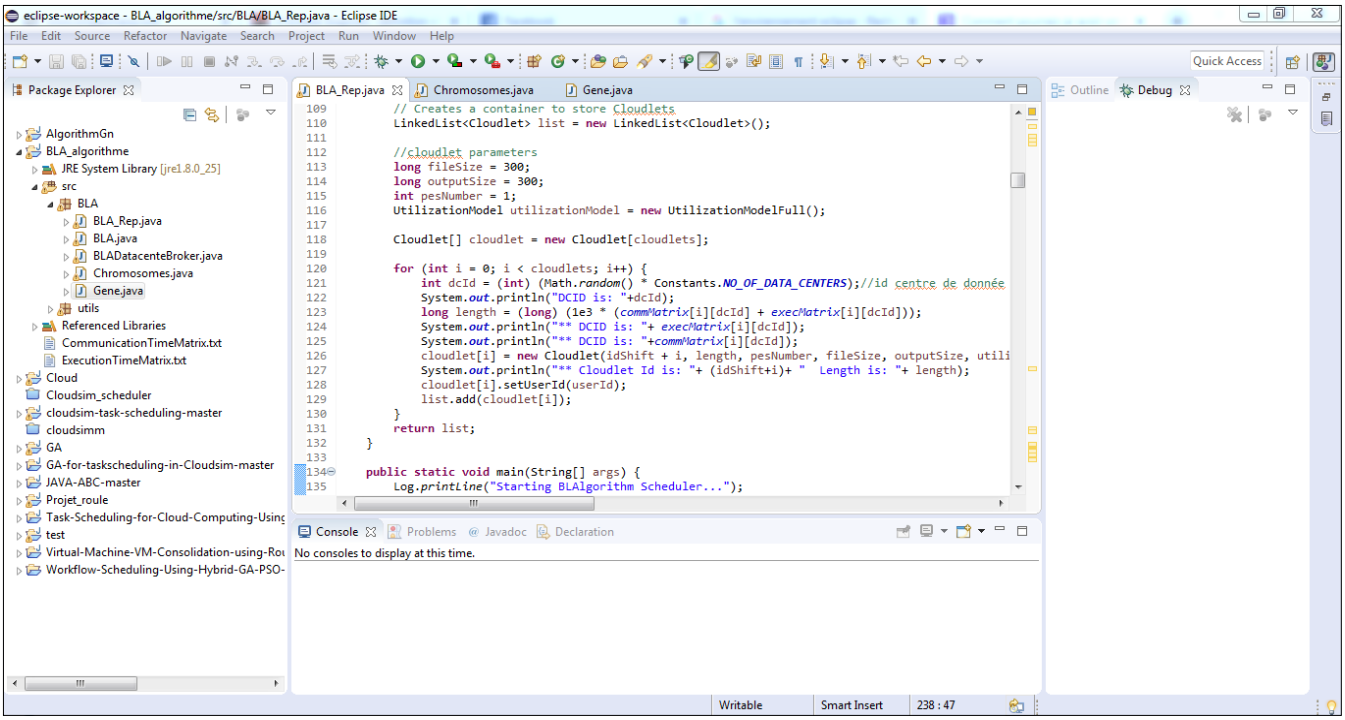



**Figure A.3 : environnement de développement éclipse**

## 3. *Le simulateur Cloudsim*

CloudSim est un Framework de simulation généralise et extensible qui permet la modélisation, la simulation et l'expérimentation des nouvelles infrastructures de Cloud Computing et des services d'application associes. Nous avons utilisé pour la réalisation de nos travaux de thèse la version du simulateur CloudSim 3.0.3. Le simulateur ClouSim est compose de plusieurs classes (Figure ci-dessus)

Caractéristiques principales :

- prise en charge de la modélisation et de la simulation de centres de données d'informatique en nuage à grande échelle
- prise en charge de la modélisation et de la simulation des hôtes de serveur virtualisés, avec stratégies personnalisables pour le provisionnement des ressources de l'hôte sur des machines virtuelles
- support pour la modélisation et la simulation de conteneurs d'applications
- aide à la modélisation et à la simulation de ressources informatiques sensibles à l'énergie

- prise en charge de la modélisation et de la simulation de topologies de réseau de centres de données et d'applications de transmission de messages
- support pour la modélisation et la simulation de nuages fédérés
- support pour l'insertion dynamique d'éléments de simulation, arrêt et reprise de la simulation
- prise en charge des stratégies définies par l'utilisateur pour l'allocation des hôtes aux machines virtuelles et stratégies pour l'allocation des ressources hôtes aux machines virtuelles

## *3.1Architecture du cloudsim*

Ce framework modélise et simule l'environnement du Cloud Computing et ses services, a été réaliser en java, afin que nous puissions simuler ces environnement, il faut connaitre l'architecture de ce framework.
La structure logicielle de CloudSim et ses composants est représentée par une architecture en couche comme s'est illustré sur la figure suivante**.**

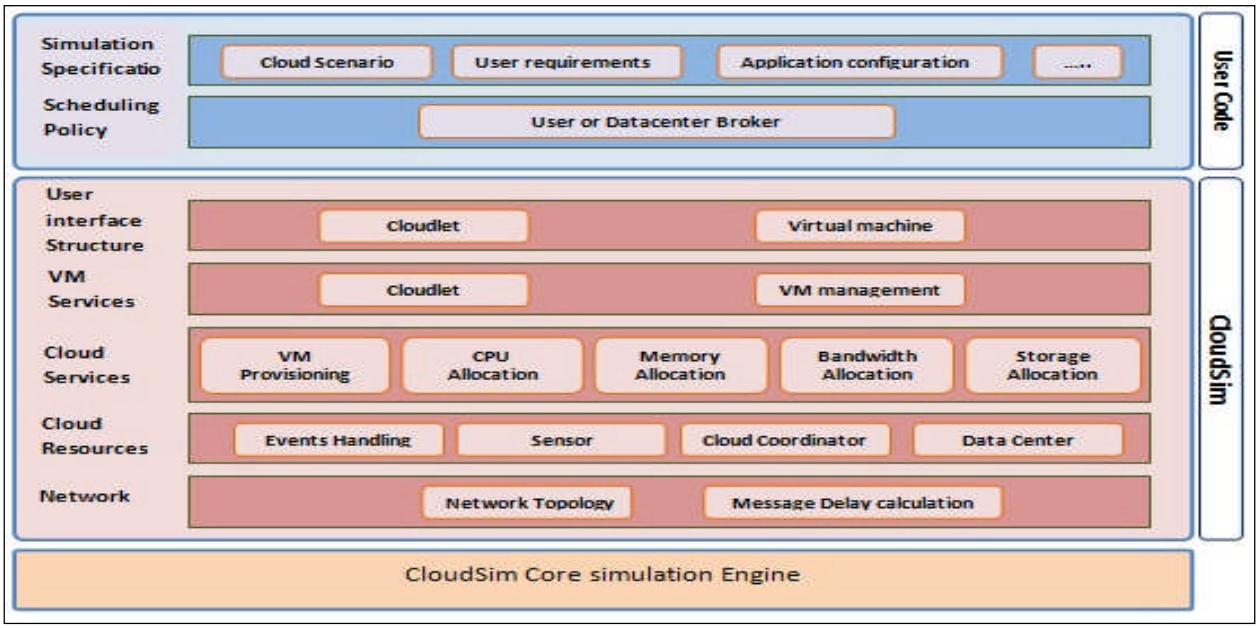



**Figure A.4 : architecture du Cloudsim*[A.3]***

La figure illustre les différentes couches de la structure du Cloud et ses éléments architecturaux. SimJava, qui implémente les fonctionnalités de base requises pour les cadres de simulation au niveau supérieur, les fichiers en attente, le traitement des événements, la création des composants du système (services, hôte, centre de données, courtier, les machines virtuelles), la communication entre les composants et la gestion de l'horloge de simulation.

CloudSim prend en charge la modélisation et la simulation de l'environnement de base de données basé sur Cloud, tel que les interfaces de gestion dédiées aux ordinateurs virtuels, la mémoire, le stockage et la bande passante. La couche Cloud Gestion des bases de données

(VM, hôtes, centres de données, applications) au cours de la période de simulation. Dans la couche plus haute de la pile de simulation, on trouve le code d'utilisateur qui expose la configuration de l'authenticité des hôtes (ex: nombre de machines, leurs spécifications), les politiques d'ordonnancement de Broker, applications (ex : nombre de tâches et leurs besoins), VM, nombre d'utilisateurs. Les fichiers du Cloud sont accessibles ici. ***[A.3]***

Pour utiliser ce cadre, il faut un outil de programmation. C'est pour cela, sur utiliser Eclipse pour faire la programmation en Java.

## *3.2Les différentes classes du Cloudsim [A.4]*

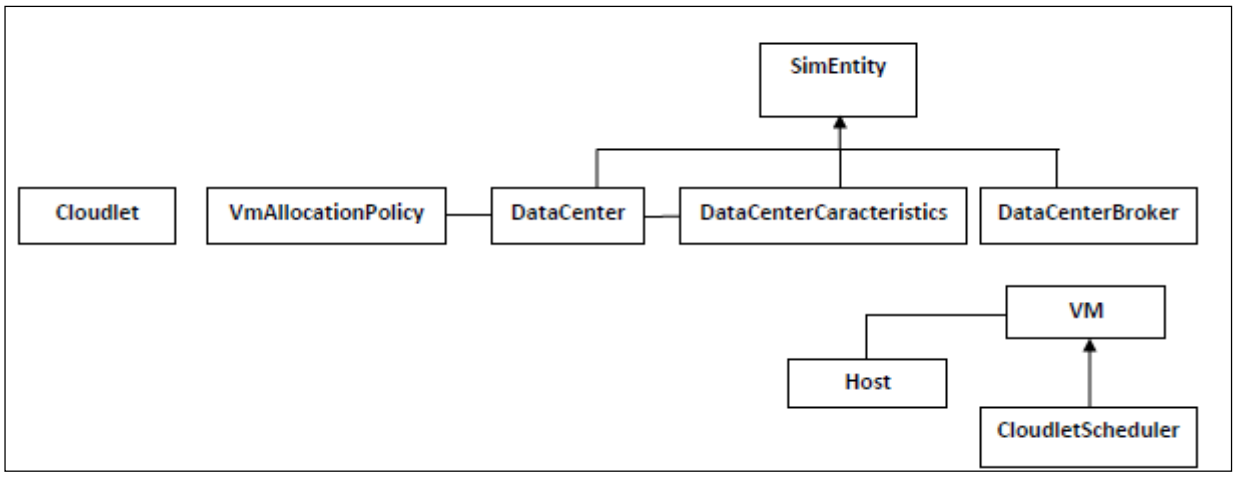



**Figure A.5 : les différentes classes de CloudSim**

- ***SimEntity***

Il s'agit d'une classe abstraite, elle représente l'entité de simulation qui est capable d'envoyer des messages à d'autres entités et de gérer les messages reçus ainsi les évènements.
Toutes les entités doivent étendre cette classe et redéfinir ses trois méthodes : **StartEntity**(), **processeEvent**() et **shutdownEntity().** Ces méthodes définissent les actions pour l'initialisation, le traitement des événements et la destruction de l'entité.

- ***Cloudlet***

C'est une classe qui représente une tâche. Elle modélise les services d'application du Cloud (comme la livraison, réseaux sociaux et les sites d'affaires). Cette classe peut aussi être étendue pour supporter la modélisation des performances et d'autres paramètres de composition pour les applications telles que les transactions dans les applications orientées bases de données (Oracle, SQL).

- ***Data center***

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

- ***DatacenterBrocker***

Cette classe modélise le courtier, qui est responsable de la médiation entre les utilisateurs et les prestataires de service,selon les conditions de QoS des utilisateurs il déploie les tâches de service à travers les Clouds.le Broker agisse au nom des utilisateurs, identifie les prestataires de service appropriés du Cloud par le service d'information du Cloud CIS (cloud information services), en négociant avec eux pour une allocation des ressources qui répond aux besoins des utilisateurs par le respect des QOS demandées.

- ***DatacenterCaracteristic***

C'est la classe qui contient les informations sur la configuration des ressources des Datacenters. Parmi eux le système d'exploitation, la liste des hôtes, le coût par seconde d'utilisation des ressources, etc.

- ***Hôst***

Cette classe représente un serveur informatique physique dans un Cloud. L'hôst exécute des actions liées à la gestion des machines virtuelles et a une politique définie pour l'approvisionnement mémoire et bande passante, ainsi que d'une politique de répartition des PE (Processeur Élément) à des machines virtuelles. Un hôte est associé à un Datacenter. Il peut héberger un ou plusieurs VMs.

- ***VM***

Cette classe modélise une instance de machine virtuelle VM, qui est gérée et hébergée par une machine pendant son cycle de vie par le composant Cloud Hôst. Chaque composant de VM a accès à un composant qui stocke les caractéristiques liées à elle telles que : l'accès mémoire, le processeur, la capacité de stockage et les politiques de provisionnement interne de la machine virtuelle.

- ***VMAllocationPolicy***

C'est une classe abstraire qui représente la politique de provisionnement des hôtes aux machines virtuelles dans un Datacenter.

L'objectif de VmAllocationPolicy est de sélectionner un centre de données répondre à l'exigence de mémoire, de stockage et de disponibilité d'un ordinateur virtuel.

## *3.3 Installation et implémentation du CloudSim sous Eclipse*

### *3.3.1 Procédure d'installation*

Pour l'installation du Cloudsim au niveau de l'IDE éclipse on a qu'à suivre les étapes suivantes :

***Etape1***: Télécharger les fichiers suivants au préalable :

- CloudSim 3.0.3 toolkit

https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3

- CloudSim

Mathshttp://redrockdigimark.com/apachemirror//commons/math/binaries/commons-math3-3.6.1-bin.zip

***Etape 2***: Décompresser les deux fichiers avant de commencer l'installation, et on aura les fichiers illustrés dans la figure qui suit :

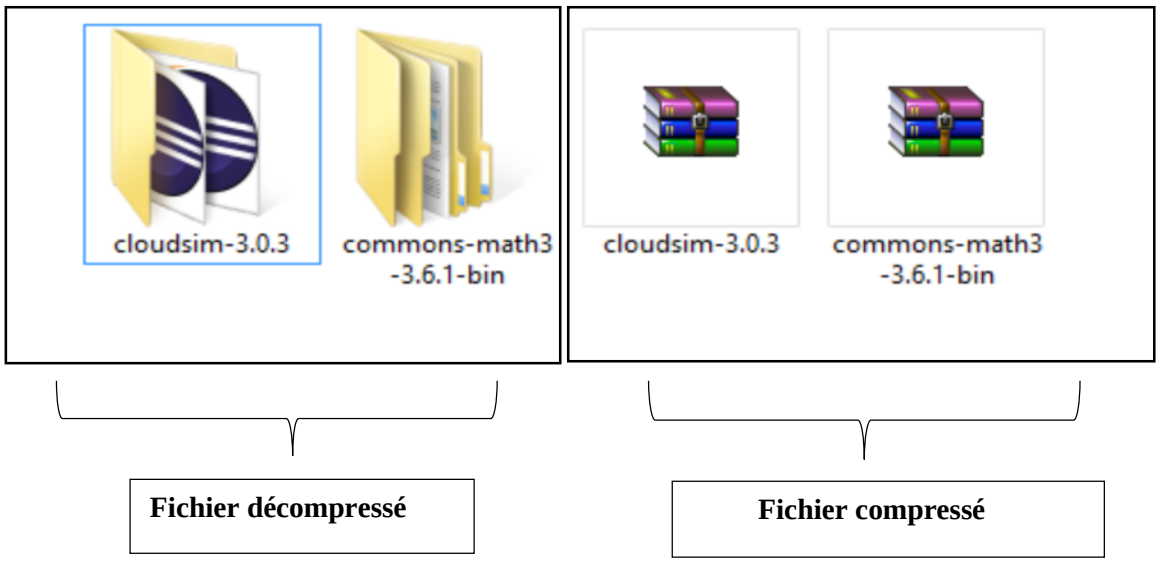



**Figure A.6 : les fichiers nécessaires pour l'installation de CloudSim**

***Étape 3* :** Ouvrez Eclipse Java IDE et cliquez sur **File** >**New**> , une boîte de dialogue sera ouverte choisissez « **Java Projet** »

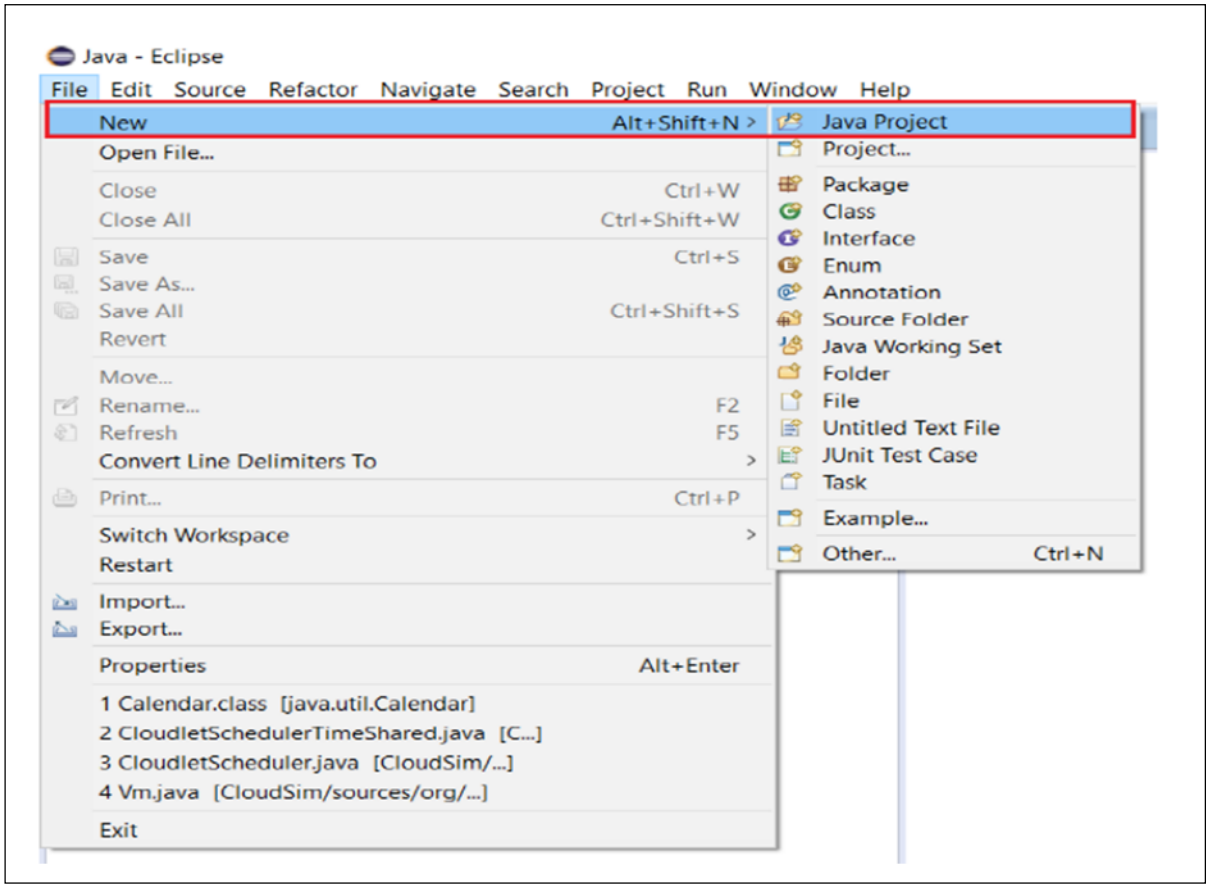



**Figure A.7 : Création d'un nouveau projet java**

***Etape 4***: Dans la boîte de dialogue "**New Java Project** ", entrez " **projet Name**" désélectionnez "**Use default Location**" puis cliquez sur " **Browse**" et parcourez le chemin d'accès auquel vous avez extrait le dossier "**Cloudsim-3.0.3**" puis cliquez sur **NEXT**

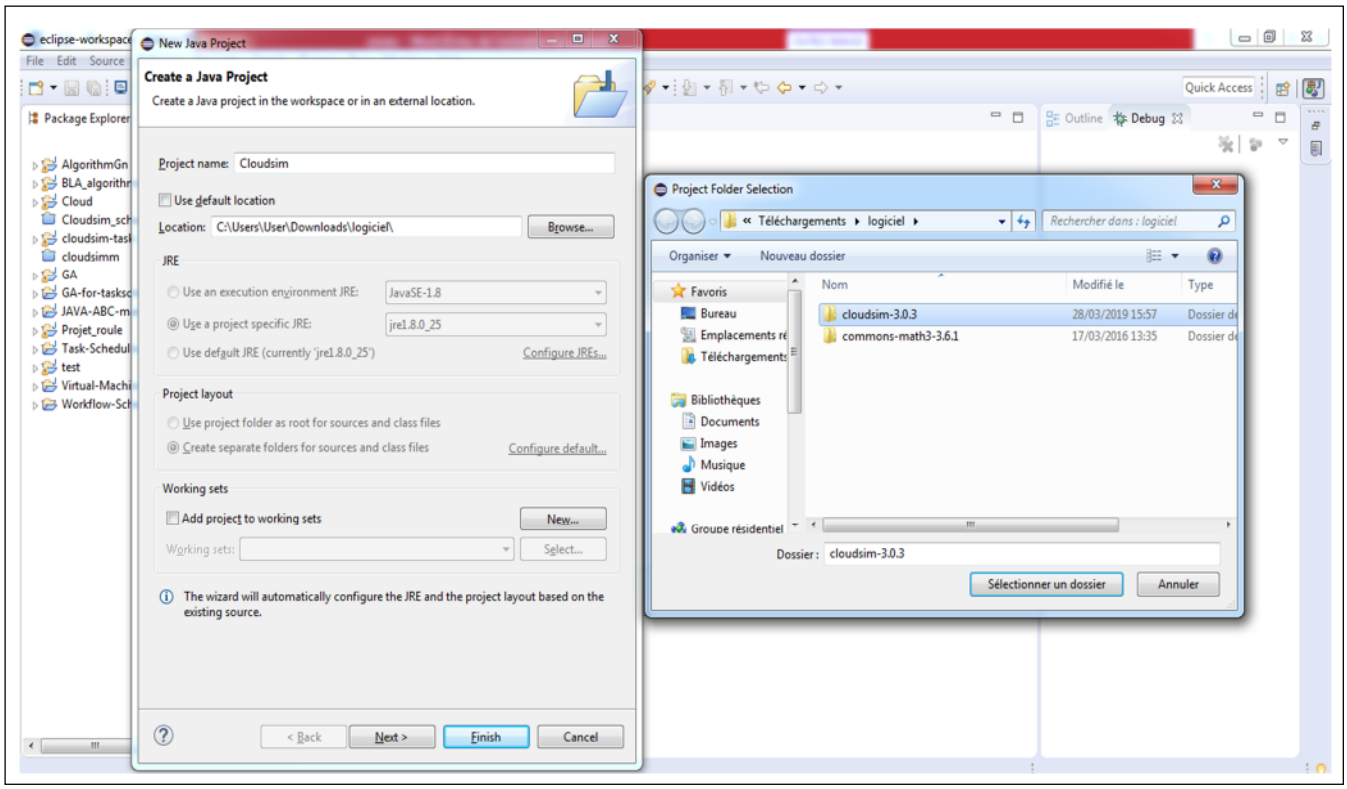



**Figure A.8 : Importation du fichier d'installation du CloudSim**

***Étape 5*** **:** Une fois que vous cliquez sur **Next**, allez dans l'onglet "**Librairies**" pour ajouter les librairies « **commons-math3-3.6.1.jar** », nous devons cliquer sur "**AddexternalJARs**", puis parcourir le chemin sur lequel vous avez téléchargé et décompressé le fichier « **commun-math3-3.6.1.jar** ». Ajoutez-les à la liste en cliquant sur Ouvrir.

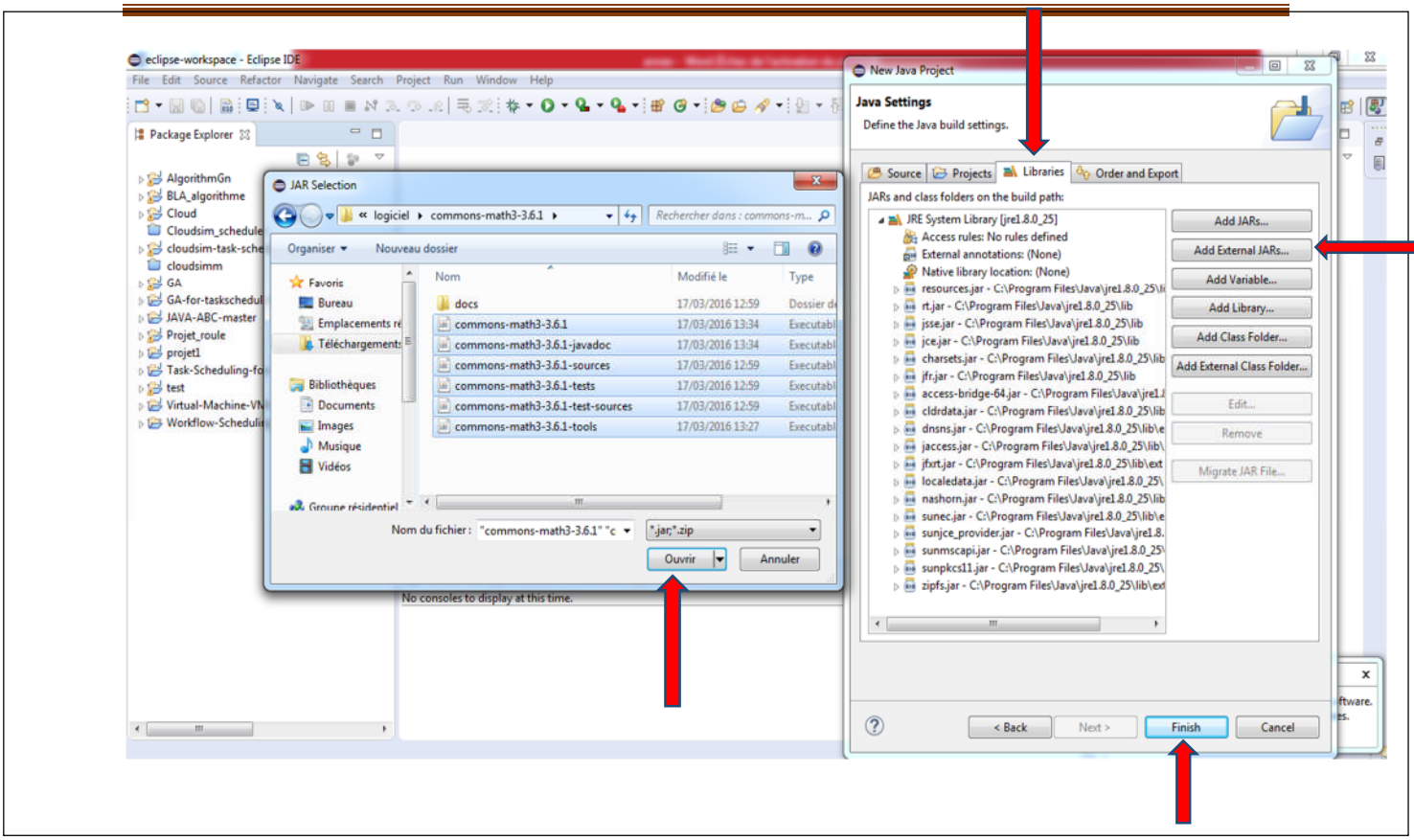



**Figure A.9 : importation des Librairies propres à CloudSim**

Enfin achever l'installation en cliquant sur le bouton « **Finish** » et clôturer l'installation

Le projet commencera à construire (compilation) automatiquement, ce qui peut prendre de 2 à 10 minutes selon la configuration de votre machine. Une fois le processus de construction est terminé, CloudSim est maintenant installé sous eclipse et vous pouvez commencer à travailler avec n'importe quel exemple CloudSim fourni dans le dossier '/ examples', ou bien implémenter d'autre algorithme comme pour d'autre simulation comme on l'a fait dans cas et qui sera l'étape suivante à expliquer.

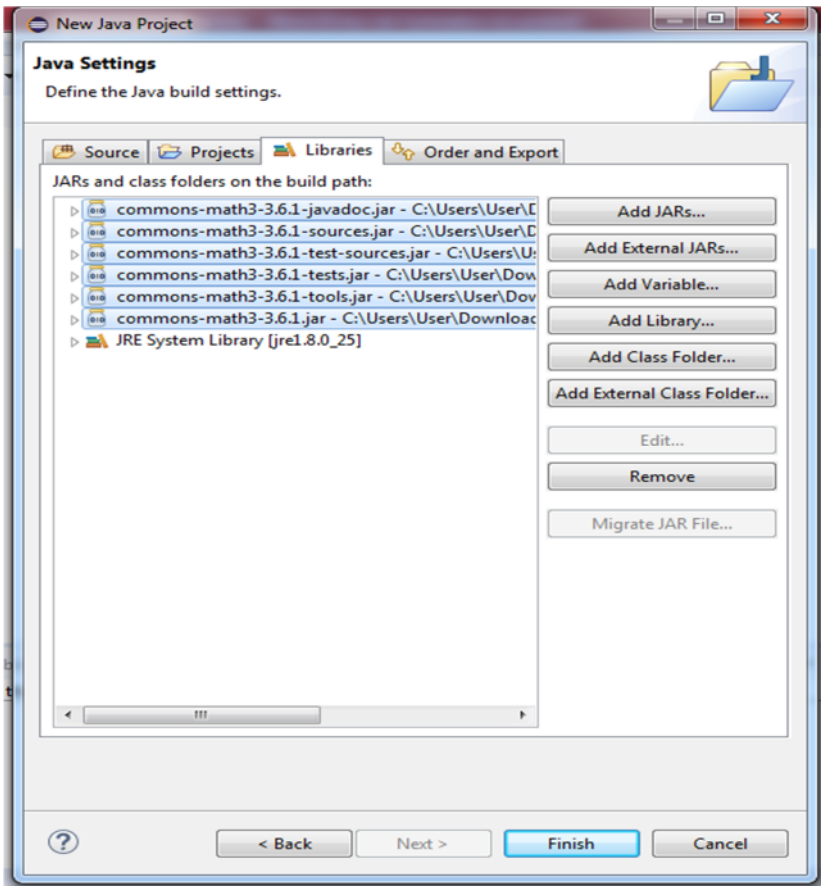
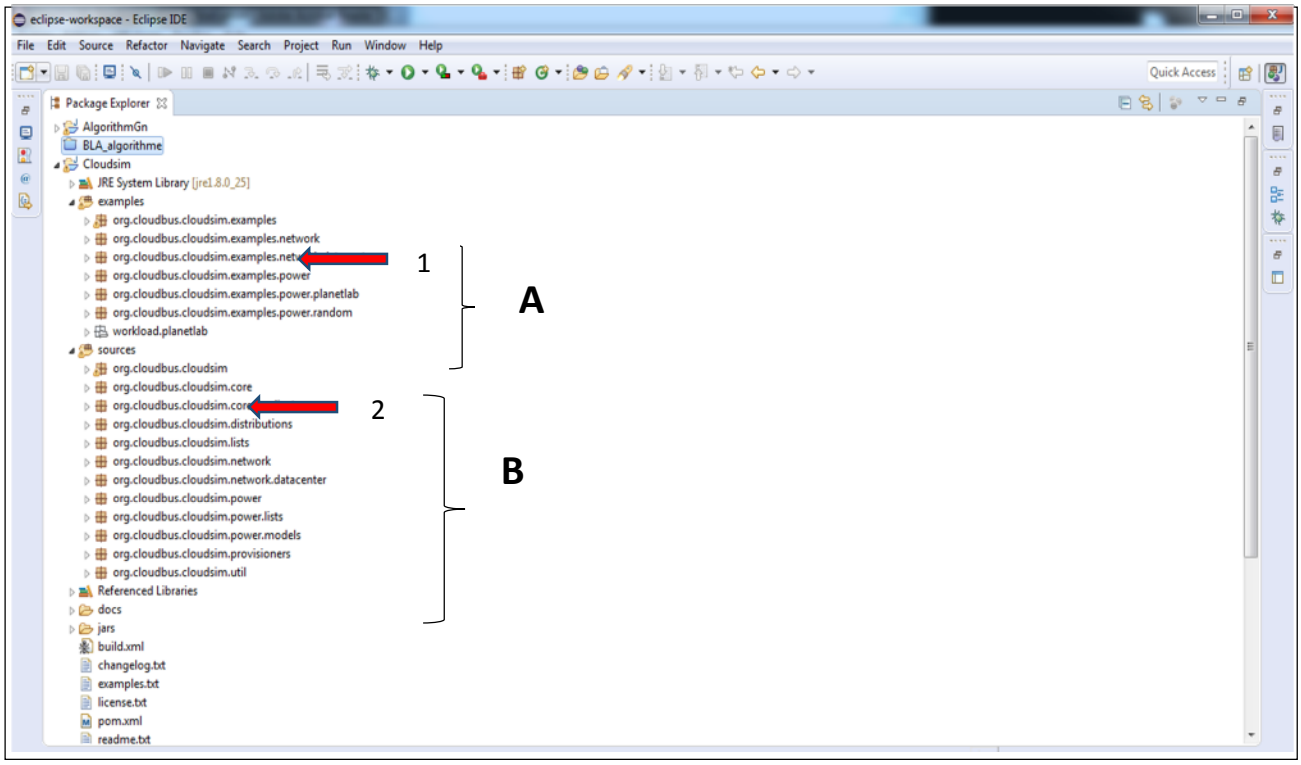



**Figure A.10 : fin de l'installation du CloudSim**

Le projet commencera à construire (compilation) automatiquement, ce qui peut prendre de 2 à 10 minutes selon la configuration de votre machine. Une fois le processus de construction est terminé, CloudSim est maintenant installé sous eclipse et vous pouvez commencer à travailler avec n'importe quel exemple CloudSim fourni dans le dossier '/ examples', ou bien implémenter d'autre algorithme pour d'autre simulation comme on l'a fait dans cas et qui sera l'étape suivante à expliquer.

### *3.3.1 Présentation des différents packages du CloudSim*

CloudSim est composé de deux packages principale : examples et le package sources et chaque packages contient à son tour des sous packages comme le montre la figure suivante




- 1 : le package CloudSim-3.0.3/Examples
- 2 : le package CloudSim-3.0.3/sources.
- A : l'ensemble des sous packages du package 1
- B : l'ensemble des sous packages du package2

**Figure A.11 : les différents packages du CloudSim**

Le package **Exemples** représente un ensemble d'exemple implémentés sous Cloudsim pour réaliser des tests et effectuer différentes simulations.
Pour le deuxième package sources qui est le plus important contient différents sous packages de base qui seront utilisés et appelés à chaque implémentation au niveau du CloudSim et nous citons quelques sous package :

- Le package **org.cloudbus.cludsim** est un package qui contient

L'ensemble des classes de base du Cloud sui sont par exemple les classes : coudlet, DataCenter, Host CloudletScheduler…etc comme le montre la figure suivante

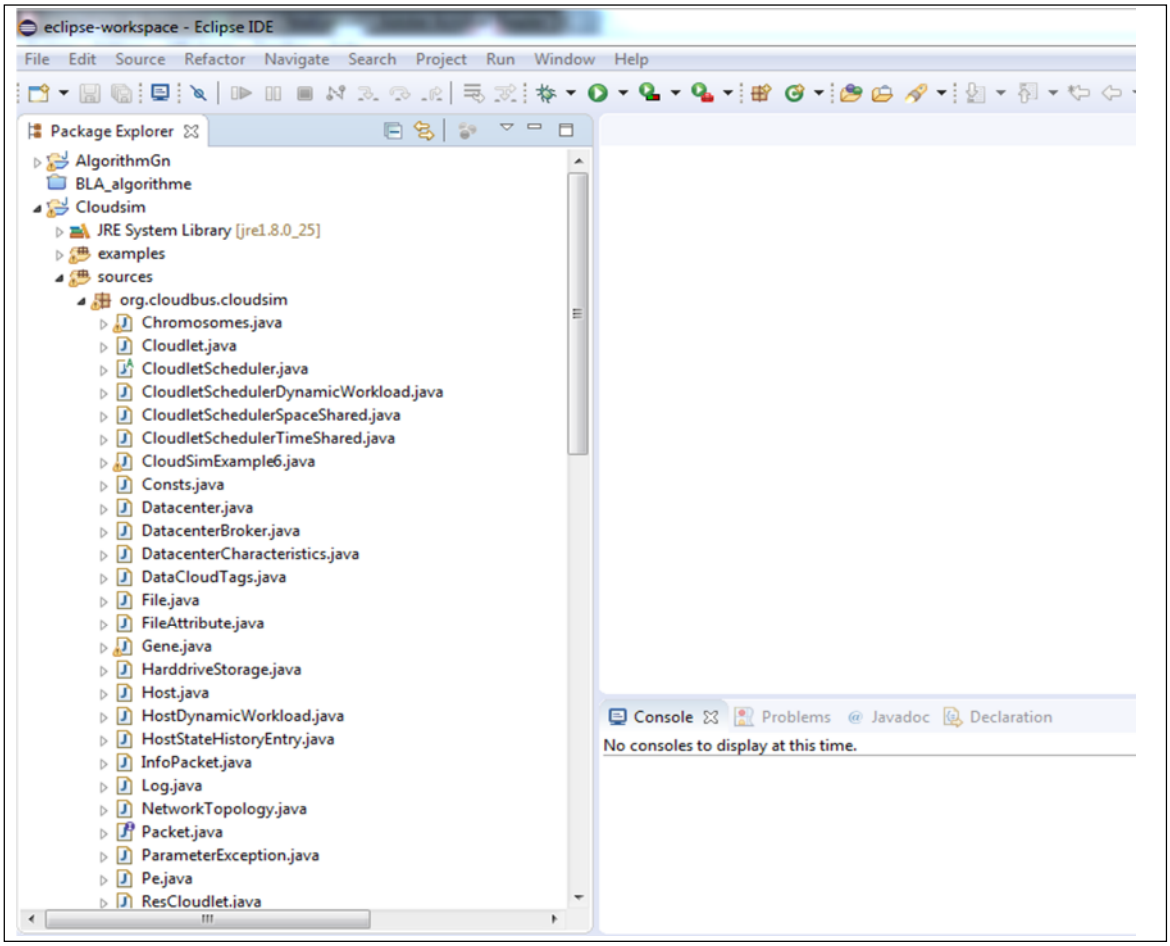



**Figure A.12 : Les classes du package org.cloudbus.cloudsim**

- Le package **org.cloudbus.cloudsim.lists** : contient les différentes classes des listes des éléments du cloud qui sont la liste des cloudlets, la liste des Hosts, la liste des Vms…etc

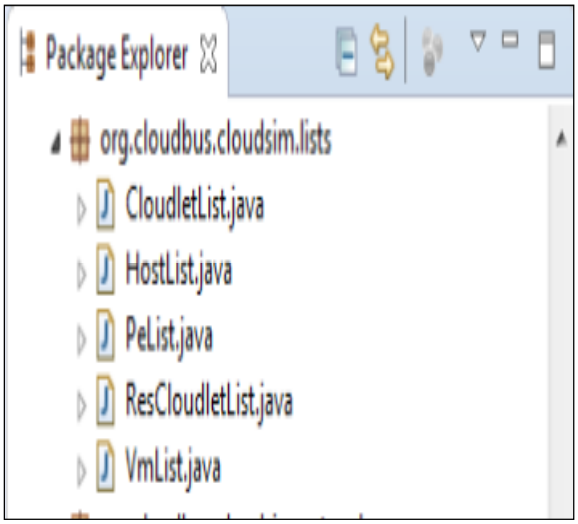



**Figure A.13 : Les classes du package org.cloudbus.cloudsim.lists**

- Le package **org.cloudbus.cloudsim.provisioners :**

Contient les différentes classes nécessaires pour la réservation de la bande passante, la Ram et les Périphériques edges(PE)

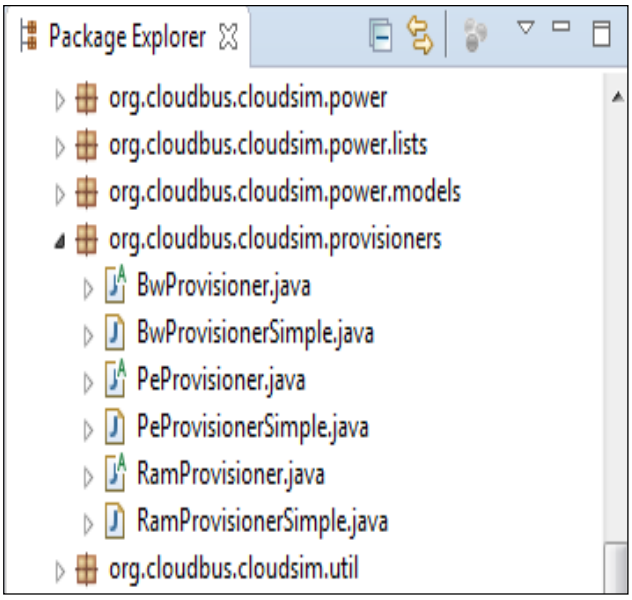



**Figure A.14 : Les classes du package org.cloudbus.cloudsim.provisioners**

# Résume mémoire

Le problème traité dans ce travail est le problème d'ordonnancement de tâches dans le Cloud Computing en vue de minimiser le makespan et le flow time.

L'apparition de la métaheuristique appelée « BLA » a sollicité notre attention. C'est une nouvelle métaheuristique qui a enrichi le nombre de métaheuristiques basées sur la vie des abeilles.

Dans ce nouveau paradigme, les fonctions de gestion et d'exploitation, telles que la planification des travaux, visent à fournir des services performants et économiques demandés par les utilisateurs mobiles et exécutés par des nœuds de brouillard.

Au cours de notre travail, nous avons présenté les étapes suivies pour la résolution du problème d'ordonnancement dans le Cloud, nous avons testé un algorithme bio-inspiré qui est l'algorithme BLA (Bees Life Algorithm) en prenant en considération les objectifs : makespan et flow time. Afin de tester la performance de notre algorithme, nous avons effectué une série de tests de simulation, une étude comparative a été réalisée avec l'algorithme BLA et l'algorithme génétique GA pour une étude de performance.

Nous pouvons rajouter aussi que grâce à CloudSim, notre étude et recherche a été plus pratique et plus facile, car ses simulations ont été pertinentes et importantes.

## Mots clés

BLA – GA- Cloudsim- Optimisation - Ordonnancement – Metaheuristique - abeilles - Bees Life Algorithme - croisement et mutation – makespan- flow time