REPUBLIQUE ALGERIÈNNE DÈMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :…….

Série :……

T H È S E

En vue d'obtenir le titre de

Docteur en Informatique 3éme cycle LMD

De l'université Mohamed Khider – Biskra

Option : Intelligence Artificielle

Une approche formelle pour la planification des tâches pour la QoS dans le cloud-computing

Présentée et soutenue par :

FEMMAM Manel

Devant le jury composé de :

Pr. SENOUCI Mohamed	Université d'Oran 1	Professeur	**Président**
Pr. KAZAR Okba	Université de Biskra	Professeur	**Rapporteur**
Pr. BOUBETRA Abdelhak	Université de bordj bou arreridj	Professeur	**Examinateur**
Dr. KAHLOUL Laid	Université de Biskra	MCA	**Examinateur**
Dr. BENNOUI Hammadi	Université de Biskra	MCA	**Examinateur**
Dr. BAARIR Soheyb	Université de Paris 6	MCA	**Invité**

Année universitaire **2017-2018**

REMERCIEMENTS

Je remercie, tout d'abord, Dieu qui m'a éclairé le chemin du savoir.

Ensuite, je tiens à remercier mon encadreur le Professeur ***KAZAR Okba*** *pour ses conseils, ses orientations et son aide.*

Je tiens à remercier particulièrement ***Dr. KAHLOUL Laid*** *pour toutes nos discussions, ses conseils et son aide qui m'ont accompagnées tout au long de cette thèse.*

Je remercie également le président du jury, ***Pr. SENOUCI Mohamed*** *de l'université d'Oran 1, et les autres membres,* ***Pr. BOUBETRA Abdelhak*** *de l'université de Bordj bouareridj,* ***Dr. BENNAOUI Hammadi*** *de l'université de Biskra d'avoir accepté d'évaluer ce travail.*

Je remercie ***Dr.BAARIR Soheyb*** *d'avoir accepté l'invitation d'assister à ma soutenance.*

Enfin, je tiens à exprimer mes sincères gratitudes aux personnes qui ont contribués de près ou de loin, à l'élaboration de la présente thèse de Doctorat et en particulier ma famille.

Dédicace

Je dédie ce travail

À mon père et à ma mère,

À ma grande et petite famille,

À tous mes proches.

ملخص

في وقتنا الحالي، تتوفر العديد من الخوارزميات القابلة للتطوير لجدولة سير العمل في الحوسبة السحابية،حيث تركز معظم هذه الخوارزميات على الكفاءة، وتتجاهل مشكلة المرونة. البحوث المتعلقة بشبكات بيتري تعالج هذه المشكلة.وقد اقترحت عدة ملحقات لتسهيل نمذجة النظم المعقدة وتتمثل في الإضافات النموذجية وهي "اللون"، "الوقت" و "التسلسل الهرمي".

من أجل رسم مشاكل جدولة في شبكات بيتري، يمكننا استخدام نظرية القياسية لشبكات بيتري.في هذه الحالة، يمكن التقليل من مشكلة الجدولة من خلال إيجاد تسلسل أمثل من التحولات بدءا من الوسم الأولي إلى الوسم النهائي.

للوصول لأحسن جدولة، اقترحنا نهجا جديدا على أساس الشكلية المقترحة مؤخرا، "شبكات بتري التطورية" (EPN) وهو امتداد لشبكات بيتري، مزودة بإثنين من العوامل الوراثية والتي هي التهجين والطفرة.

أهداف بحثنا تتمثل في التقليل من وقت التنفيذ لمختلف تطبيقات سير العمل (wotkflow)وكذلك التكاليف المتكبدة باستخدام موارد السحابة. وقد أجريت بعض التجارب لإثبات فائدة بحثنا.

الكلمات المفتاحية :جدولةسيرالعمل،الحوسبة السحابية،شبكات بيتري،الخوارزميةالجينية.

Abstract

Nowadays, many evolutionary algorithms for workflow scheduling in cloud computing are available. Most of those algorithms focus on the effectiveness, discarding the issue of flexibility. Research on Petri nets addresses the issue of flexibility; many extensions have been proposed to facilitate the modeling of complex systems. Typical extensions are the addition of "color", "time" and "hierarchy".

By mapping scheduling problems into Petri Nets, we are able to use standard Petri Net theory. In this case, the scheduling problem can be reduced to finding an optimal sequence of transitions leading from an initial marking to a final one. To find the optimal scheduling, we proposed a new approach based on a recently proposed formalism "Evolutionary Petri Net" (EPN), which are an extension of Petri Net, enriched with two genetic operators, crossover and mutation. The objectives of our research are to minimize the workflow application completion time (makespan) as well as the amount cost incurred by using cloud resources. Some numerical experiments are carried out to demonstrate the usefulness of our algorithm.

Keywords: workflow scheduling; cloud computing; Petri nets; genetic algorithm

Résumé

Aujourd'hui, de nombreux algorithmes évolutifs pour l'ordonnancement du *workflow* dans le *cloud computing* sont disponibles. La plupart de ces algorithmes sont axés sur l'efficacité, et ignorent le problème de la flexibilité. La recherche sur les réseaux de Petri traite ce dernier problème. Plusieurs extensions ont été proposées pour faciliter la modélisation des systèmes complexes. Les extensions typiques sont l'ajout de "couleur", "temps" et "hiérarchie".

Afin de mapper les problèmes d'ordonnancement dans les réseaux de Petri, nous pouvons utiliser la théorie standard des réseaux de Petri. Dans ce cas, le problème d'ordonnancement peut être réduit à trouver une séquence optimale de transitions allant d'un marquage initial vers un marquage final. Pour trouver un ordonnancement optimal, nous avons proposé une nouvelle approche basée sur un formalisme récemment proposé, les «Réseaux de Petri Evolutionnaire» (EPNs), qui est une extension des réseaux de Petri, enrichie de deux opérateurs génétiques, d'un croisement et d'une mutation. Les objectifs de notre recherche sont de minimiser le temps d'exécution des applications de *workflow (makespan)* ainsi que le coût encourus en utilisant les ressources du *cloud*. Certaines expériences sont réalisées pour démontrer l'efficacité et l'utilité de notre approche.

Mots-clés: Ordonnancement du *workflow*; *Cloud computing*; Réseaux de Petri; Algorithmes génétiques.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Plan

I. Contexte et motivation de travail

Récemment, l'informatique dans les nuages (appelé *cloud computing* en anglais dans la suite du manuscrit) est proposée pour fournir une infrastructure informatique à la demande pour un certain nombre d'applications. Les applications fournies en mode *cloudcomputing* ne se trouvent pas forcément sur un serveur hébergé par l'utilisateur mais dans un « nuage » formé de l'interconnexion de plusieurs serveurs localisés à des endroits distants.

Cette interconnexion est réalisée au niveau de fermes de serveurs appelées communément *"datacenters"*ou centres de traitement de données. Ceci est une conséquence directe du procédé de virtualisation. Ce dernier est le socle de ce nouveau paradigme, dont l'objectif est de faire fonctionner des applications sur différents systèmes d'exploitation sur un même serveur physique [31].

Le *cloud computing* est basé à la fois sur des aspects informatiques et économiques [6]. D'une part, l'aspect informatique concerne l'infrastructure utilisée pour fournir les ressources, et il provient principalement du modèle informatique de grille amélioré par une flexibilité offerte par différents *frameworks* logiciels. D'autre part le modèle économique du *cloud* fournit des services flexibles et évolutifs, à la demande des utilisateurs, via un modèle de paiement à l'usage. Ainsi, l'accès effectif et l'utilisation de ces services soulèvent plusieurs problèmes dont l'ordonnancement est le plus important.

Ce modèle économique est constitué de quatre types de services SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service) et HaaS (Hardware as a Service), et de trois types d'usage (public, privé et hybride). En effet, le concept économique ajoute des contraintes et des objectifs totalement nouveaux, affectant l'ordonnancement sur une nouvelle architecture distribuée telle que le *cloud*.En outre, la flexibilité du *cloud* apporte des problèmes d'ordonnancement aux différents niveaux du nuages : au niveau de service, au niveau de tâche et au niveau de machine virtuelle [60].

Les services offerts par le *cloud* diffèrent les uns des autres par leurs QoS (Qualité de Service) non-fonctionnelles, telles que le temps, le coût, la consommation d'énergie, etc.

Ces paramètres de QoS peuvent être définis et proposés par différents contrats de service appelés SLA (Service Level Agreements). Le SLA spécifie les exigences négociées en termes de ressources, les QoS, les attentes minimales et les obligations qui existent entre les utilisateurs et les fournisseurs de *cloud* [6].

En effet, la flexibilité et la dynamique offerte par le *cloud* et certains défis majeurs tels que la réduction du temps d'exécution, augmentent la complexité de l'ordonnancement et

rendent difficile à trouver des solutions efficaces à ces problèmes dans un délai raisonnable. Donc, l'application de méthodes d'optimisation exactes s'avère illusoire en raison du temps de calcul.

En tant que solution, les méta-heuristiques multi-objectifs constituent une bonne alternative. Cependant, ces méta-heuristiques ainsi que la modélisation du problème doivent être repensées pour prendre en compte à la fois des spécifications et des contraintes du problème de l'ordonnancement sur le *cloud*.

Les réseaux de Petri sont des outils formels pour la modélisation et l'analyse de systèmes qui présentent des comportements tels que la concurrence, les conflits et les dépendances causales entre les événements. Les approches basées sur les réseaux de Petri résolvent le problème de flexibilité des tâches et des ressources. La puissance de modélisation et d'analyse des réseaux de Petri permet d'explorer des aspects facilement différents du problème d'ordonnancement (propriétés comportementales et structurelles) et d'intégrer les facteurs ignorés dans d'autres approches.

Les systèmes de gestion de *workflows* peuvent être considérés comme un type de service facilitant l'automatisation des applications de e-business et e-science sur la grille et le *cloud*. Ils sont utilisés pour gérer ces applications de façon transparente en masquant l'orchestration et les détails d'intégration spécifiques lors de l'exécution des tâches sur les ressources distribuées de grille ou de *cloud*.

Les systèmes de gestion de *workflows* requièrent des stratégies d'ordonnancement plus élaborées pour répondre aux exigences de QoS (*makespan*, coût et autres) spécifiées dans le SLA, ainsi qu'aux relations de précédence entre les tâches du *workflow*. L'étude des stratégies d'ordonnancement de workflows devient un enjeu important dans le *cloud computing* [6].

II. Problème et objectifs

L'ordonnancement est un aspect fondamental dans le cadre du *cloud computing* car il permet la parallélisation des applications et l'amélioration de leurs performances. L'ordonnancement d'un processus consiste à respecter les contraintes (les contraintes de précédence des tâches) qui lient les différentes tâches le constituant et à déterminer les ressources à leur affecter. Autrement dit, un problème d'ordonnancement consiste à organiser l'exécution d'un ensemble de tâches liées par des contraintes de précédence, de données et de ressources [31].

Le problème d'ordonnancement de *workflows* est un problème d'optimisation combinatoire, où il est impossible de trouver une solution globale optimale en utilisant des algorithmes ou des règles simples. Il est bien connu comme un problème NP-complet [6].

Le problème d'ordonnancement de *workflows* dans le *cloud* est largement étudié dans de nombreux travaux [60] [6] [46] [47]. En revanche, la plupart de ces travaux :

1) Sont ne respectent pas la relation de précédence entre les différentes tâches.
2) soit transforment le problème multi-objectif à un problème mono-objectif utilisant des fonctions d'agrégation.

Dans ce travail, notre objectif est de proposer et d'évaluer une nouvelle approche d'optimisation multi-objectif modélisée sous forme des réseaux de Petri évolutionnaire pour l'ordonnancement de *workflow* scientifique. L'algorithme doit :

- Respecter la contrainte de précédence entre les différentes tâches qui composent le *workflow*,
- Prendre en compte deux critères conflictuels, à savoir le temps d'exécution et le coût engendré par l'utilisation d'un ensemble de ressources,
- Optimiser les deux critères de *QoS* (*makespan*, coût).

III. Contributions de la thèse

Nous proposons un nouveau modèle d'optimisation adressé le niveau d'ordonnancement de tâche. Le présent travail se concentre plus particulièrement sur la modélisation du problème avec une extension des réseaux de Petri évolutionnaire (LEPN) et l'optimisation des deux objectifs contradictoires liés (*makespan*, coût) avec les algorithmes génétiques. Notre étude fournit un nouvel algorithme basé sur l'approche de Pareto pour aborder simultanément tous les critères et trouver un meilleur compromis.

VI. Structure de la thèse

Notre travail s'organise en cinq chapitres :

Le chapitre 1 présente la littérature qui couvre les définitions de base concernant le *cloud computing* et de *workflow*. L'objectif est de fournir en détail au lecteur les deux domaines dans lesquels s'intègrent les travaux de cette thèse. Il se compose de deux sections : dans la première seront présenté la définition du *cloud computing* et ses différents concepts, dans la seconde seront introduit les notions des *workfows* et les systèmes de gestion de *workflow*.

Le chapitre 2 présente un état de l'art sur l'ordonnancement des *workflows* dans le *cloud computing*, ainsi qu'un tour d'horizon sur les travaux existants.

Le chapitre 3 présente un état de l'art sur l'ordonnancement par les réseaux de Petri ainsi qu'une présentation de certains travaux relatifs.

Le chapitre 4 est dédié à la conception de notre approche proposée. Dans un premier temps, nous modélisons notre problème par les réseaux de Petri évolutionnaire. En deuxième lieu nous présentons notre algorithme proposé avec leurs différentes étapes.

Le chapitre 5 sera consacré à l'implémentation de notre approche, les outils de développement, ainsi que les résultats obtenus avec un test statistique seront tous présentés dans ce chapitre.

Nous terminons la thèse par une conclusion générale qui synthétisera notre projet et présentera des perspectives de recherche futures.

Chapitre I

Concepts fondamentaux de cloud et de workflow

Plan

I.1 Introduction

Le *cloud computing*, traduit le plus souvent en français par « informatique dans les nuages», « informatique dématérialisée » ou encore « info nuagique », est un domaine qui regroupe un ensemble de techniques et de pratiques consistant à accéder, en libre-service, à du matériel ou à des logiciels informatiques, à travers une infrastructure réseau (Internet). Ce concept rend possible la distribution des ressources informatiques sous forme de services pour lesquels l'utilisateur paie uniquement pour ce qu'il utilise. Ces services peuvent être utilisés pour exécuter des applications scientifiques et commerciales, souvent modélisées sous forme de *workflows*.

Ce chapitre présente en détail les deux domaines dans lesquels s'intègrent les travaux de cette thèse. Tout d'abord, dans la deuxième section, nous définissons le *cloud computing* et ses différentes composantes. Par la suite, nous présenterons, dans la troisième section, le *workflow* et ses systèmes de gestion, puis les systèmes de gestion de *workflows* pour le cloud.

I.2 Définitions et Concepts du *cloud computing*

Dans cette section nous allons définir le *cloud computing* et ses différents concepts.

I.2.1 Définition du contexte

Afin de comprendre les fondements de *cloud computing*, que l'on pourrait traduire en français par « Informatique dans le nuage », commençons par définir cette infrastructure «le nuage», De façon métaphorique le nuage en question est assimilé à Internet, ou plutôt à l'infrastructure matérielle et logicielle qui permet son fonctionnement [1]. Il est, traditionnellement utilisé pour schématiser des réseaux de télécommunications, où il y représente la partie du réseau que l'on ne connaît pas, que l'on ne sait pas définir, mais qui permet d'interconnecter tous les réseaux déployés à travers le monde. L'exemple le plus courant est illustré par le schéma donné dans la figure (I.1) Ce schéma représente un réseau informatique local (LAN), par exemple celui d'une petite entreprise. L'infrastructure gérée par l'entreprise en question (c-à-d. ordinateurs, commutateurs et routeurs) y est détaillée, alors que celle associée à Internet est représentée par un nuage.

Figure I.1. Métaphore du nuage [1].

L'idée de cette représentation est d'identifier la partie du schéma ou de la solution à représenter qui est à la charge d'un tiers. Une question se pose alors : si cette partie du réseau est gérée par un tiers, alors pourquoi la détailler ? C'est l'idée soulevée par Velte et coll. Dans le livre «*cloud computing, a practical approach*» (une approche pratique de l'informatique dans le nuage), et c'est cette idée qui, probablement, correspond le mieux à la place de la métaphore du nuage au sein de *cloud computing* [1]. Aussi *cloud computing* généralise ce concept de « partie du réseau que l'on n'administre pas, que l'on ne gère pas », à l'ensemble des ressources informatiques qui peuvent être accessibles via Internet en tant que service. Les détails techniques d'implantation de ces services, de leur hébergement ou de leur maintenance concernent alors uniquement le fournisseur, qui en contrepartie facture leur utilisation. Les services informatiques sont alors distribués publiquement.

I.2.1.1 Chronologie

D'un aspect purement technique, le *cloud computing* est loin d'être en soi une technologie nouvelle, le *cloud computing* provient de l'aboutissement de plusieurs technologies existantes antérieurement. Ces premières traces remontent aux années soixante avec les mainframes où les utilisateurs accédaient depuis leurs terminaux à des applications fonctionnant sur des systèmes centraux, à cette époque l'expression *cloud* n'était pas encore née, c'est avec l'apparition d'Internet que les architectes de réseaux la schématisaient par un nuage dans leurs croquis, on parlait alors

de « the *cloud* ». Par la suite, au début des années 2000, sont apparus des hébergeurs et des fournisseurs d'applications en ligne ASP (Application Service Providers) qui s'appuyaient sur une forme d'externalisation «Software as a Service». Et ce n'est que peu après, qu'Amazon Web Services «AWS», oriente vers les entreprises, et Google, oriente vers le grand public, font émerger le marché du *cloud computing*, suivis après par les autres éditeurs comme Microsoft et Oracle [2].

I.2.1.2 Vers une définition du *cloud computing*

Beaucoup de chercheurs ont tenté de définir le *cloud computing* [3] [4]. La plupart des définitions attribuées à ce concept semblent se concentrer seulement sur certains aspects technologiques. L'absence d'une définition standard a généré non seulement des exagérations du marché, mais aussi des confusions. Pour cette raison, il y'a eu récemment des travaux sur la normalisation de la définition du *cloud computing* comme dans [5] ou il compare plus de 20 définitions différentes et ont proposé une définition globale. En guise de synthèse des différentes propositions données dans la littérature, Yassa.S dans sa thèse de doctorat en 2014 [6] a été proposé la définition suivante de *cloud computing*, qui l'on trouve la plus adéquate :

« Le *cloud* comme un modèle informatique qui permet d'accéder, d'une façon transparente et à la demande, à un pool de ressources hétérogènes physiques ou virtualisées (serveurs, stockage, applications et services) à travers le réseau. Ces ressources sont délivrées sous forme de services reconfigurables et élastiques, à base d'un modèle de paiement à l'usage, dont les garanties sont offertes par le fournisseur via des contrats de niveau de service (***SLA, Service Level Agreement***) ».

I.2.1.3 Caractéristiques principales du *cloud computing*

Le *cloud computing* possède les caractéristiques suivantes :

- **Accès en libre-service à la demande :** Le *cloud computing* offre des ressources et services aux utilisateurs à la demande. Les services sont fournis de façon automatique, sans nécessiter d'interaction humaine [3].

- **Accès réseau universel :** Les services de *cloud computing* sont facilement accessibles au travers du réseau, par le biais de mécanismes standard, qui permettent une utilisation depuis de multiples types de terminaux (par exemple, les ordinateurs portables, tablettes, smartphones) [3].

- **Mutualisation de ressources** (***Pooling***) **:** les ressources du *cloud* peuvent être regroupées pour servir des utilisateurs multiples, pour lesquels des ressources physiques et virtuelles sont automatiquement attribuées [3]. En général, les utilisateurs n'ont aucun contrôle ou connaissance

sur l'emplacement exact des ressources fournies. Toutefois, ils peuvent imposer de spécifier l'emplacement à un niveau d'abstraction plus haut [6].

- **Scalabilité et élasticité :** des ressources supplémentaires peuvent être automatiquement mises à disposition des utilisateurs en cas d'accroissement de la demande (en réponse à l'augmentation des charges des applications), et peuvent être libérées lorsqu'elles ne sont plus nécessaires. L'utilisateur a l'illusion d'avoir accès à des ressources illimitées à n'importe quel moment, bien que le fournisseur en définisse généralement un seuil (par exemple : 20 instances par zone est le maximum possible pour *Amazon EC2*) [6]**.**

–**Autonome :** le*cloud computing* est un système autonome et géré de façon transparente pour les utilisateurs. Le matériel, le logiciel et les données au sein du *cloud* peuvent être automatiquement reconfigurés, orchestrés et consolidés en une seule image qui sera fournie à l'utilisateur [7].

–**Paiement à l'usage :** la consommation des ressources dans le *cloud* s'adapte au plus près aux besoins de l'utilisateur. Le fournisseur est capable de mesurer de façon précise la consommation (en durée et en quantité) des différents services (CPU, stockage, bande passante,…) ; cela lui permettra de facturer l'utilisateur selon sa réelle consommation [8].

–**Fiabilité et tolérance aux pannes :** les environnements *cloud* tirent parti de la redondance intégrée du grand nombre de serveurs qui les composent en permettant des niveaux élevés de disponibilité et de fiabilité pour les applications qui peuvent en bénéficier **[9].**

–**Garantie QoS :** les environnements de *cloud* peuvent garantir la qualité de service pour les utilisateurs, par exemple, la performance du matériel, comme la bande passante du processeur et la taille de la mémoire [7].

–**Basé-SLA :** les *clouds* sont gérés dynamiquement en fonction des contrats d'accord de niveau de service (SLA) **[9]** entre le fournisseur et l'utilisateur. Le SLA définit des politiques, telles que les paramètres de livraison, les niveaux de disponibilité, la maintenabilité, la performance, l'exploitation, ou autres attributs du service, comme la facturation, et même des sanctions en cas de violation du contrat. Le SLA permet de rassurer les utilisateurs dans leur idée de déplacer leurs activités vers le *cloud*, en fournissant des garanties de *QoS* [6].

Après avoir présenté les caractéristiques essentielles d'un service *cloud*, nous présentons, brièvement, dans la section suivante, quelques technologies connexes aux *clouds*.

I.2.1.3 Technologies connexes

Le *cloud computing* utilise des technologies telles que la virtualisation, l'architecture orientée services et les services web. Le *cloud* est souvent confondu avec plusieurs paradigmes informatiques, tels que le *Grid computing*, l'Utility *computing* et l'Autonomic *computing*, dont chacun partage certains aspects avec le *cloud computing*. La figure (I.2) montre la convergence des technologies qui ont contribué à l'avènement du *cloud computing*.

***Grid computing* :** le *grid computing* est un paradigme de calcul réparti qui coordonne des ressources autonomes et géographiquement distribuées pour atteindre un objectif de calcul commun. Le *grid* est basé sur le partage dynamique des ressources entre des participants, des organisations et des entreprises dans le but de pouvoir les mutualiser, et faire ainsi exécuter des applications scientifiques, qui sont généralement des calculs intensifs ou des traitements de très gros volumes de données. Le *cloud computing* est similaire au *grid computing* : les deux adoptent le concept d'offrir les ressources sous forme de services. Toutefois, ils sont différents dans leur finalité : tandis que la technologie des grilles vise essentiellement à permettre à des groupes différents de partager l'accès en libre-service à leurs ressources, le *cloud* a pour objectif de fournir aux utilisateurs des services « à la demande », sur le principe du « paiement à l'usage ». De plus le *cloud* exploite les technologies de virtualisation à plusieurs niveaux (matériel et plateforme d'application) pour réaliser le partage et l'approvisionnement dynamique des ressources.

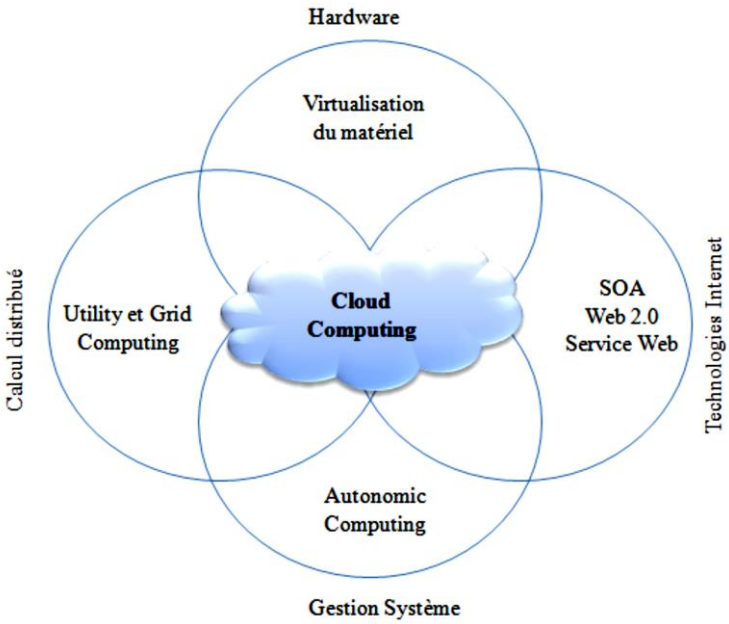


Figure I.2. Convergence des différentes avancées menant à l'avènement du *cloud computing*. [71]

- ***Utility computing*** **:**(informatique utilitaire). L'utility *computing* est simplement une délocalisation d'un système de calcul ou de stockage. Il présente un modèle d'affectation des ressources à la demande et facturation des utilisateurs en fonction de leur usage. Le *cloud computing* peut être perçu comme une réalisation de l'informatique utilitaire. Il adopte un système de tarification basé sur l'utilité, entièrement pour des raisons économiques. Avec l'approvisionnement des ressources à la demande et le paiement à l'usage, les fournisseurs de services peuvent maximiser l'utilisation des ressources et minimiser leurs coûts d'exploitation.

- ***L'autonomie computing*** **:** vise à construire des systèmes informatiques capables de s'auto-administrer en s'adaptant à des changements internes et externes sans intervention humaine. Le but de l'informatique autonome est de surmonter la complexité de la gestion des systèmes informatiques d'aujourd'hui. Bien que le *cloud computing* présente certaines caractéristiques autonomes, telles que l'approvisionnement automatique des ressources, son objectif est de diminuer le coût des ressources, plutôt que de réduire la complexité du système.

- ***Virtualisation*** **:**la virtualisation est une technologie qui fait abstraction des détails du matériel physique et fournit des ressources virtualités pour les applications de haut niveau. En effet, la virtualisation regroupe l'ensemble des techniques matérielles ou logicielles permettant de faire fonctionner, sur une seule machine physique, plusieurs configurations informatiques (systèmes d'exploitation,…), de sorte à former plusieurs machines virtuelles, qui reproduisent le comportement des machines physiques. La virtualisation constitue le socle du *cloud computing,* car elle offre la possibilité de mettre en commun des ressources informatiques à partir de *clusters* de serveurs, et ainsi d'affecter ou réaffecter dynamiquement des machines virtuelles aux applications à la demande. La figure (I.3) Montre l'architecture en couches de la technologie de virtualisation. Le moniteur de machine virtuelle (VMM), également appelé *hyperviseur,* partitionne la ressource physique du serveur physique sous-jacente en plusieurs machines virtuelles différentes, chacune fonctionnant sous un système d'exploitation distinct et une pile de logiciels utilisateur.

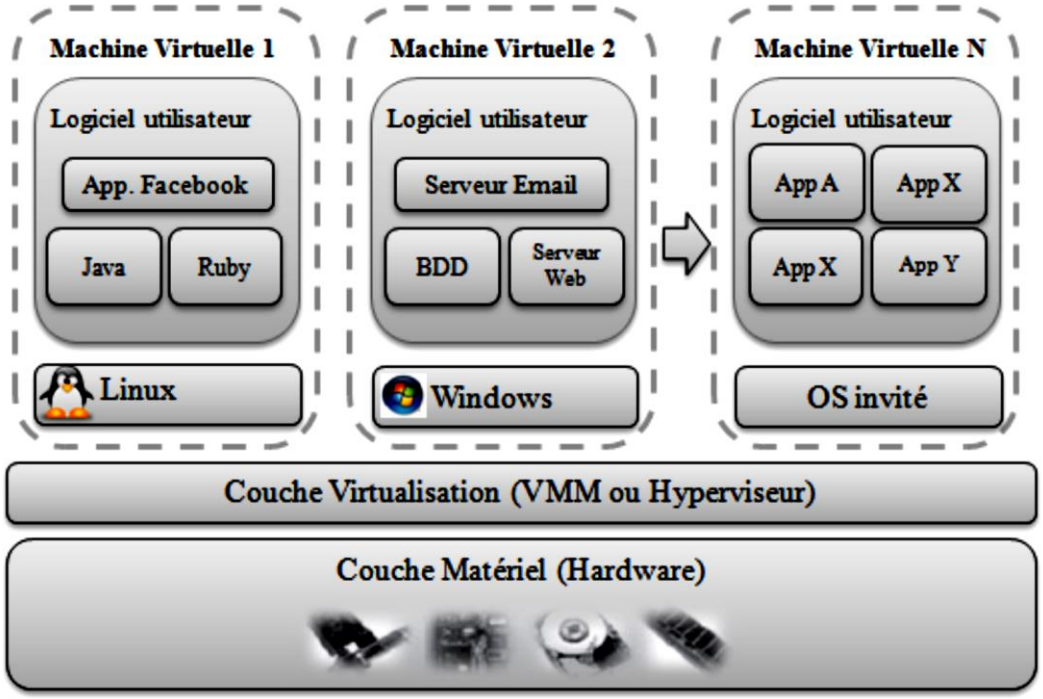


Figure I.3. Une architecture en couches de la technologie de virtualisation [6].

I.2.2 Modèles du *cloud computing*

Cette section présente les modèles du *cloud*. Nous commençons par détailler les modèles de service, puis nous donnons les modèles de déploiement et enfin nous introduisons les acteurs du *cloud*.

I.2.2.1 Modèles de services de *cloud*

XaaS (X as a Service) représente la base du paradigme du *cloud computing*, où X représente un service tel qu'un logiciel, une plateforme, une infrastructure, un Business Process, etc. Nous présentons, dans cette section, quatre modèles de services [10], à savoir: (1) Logiciel en tant que services SaaS (Software as a Service), où le matériel, l'hébergement, le *framework* d'application et le logiciel sont dématérialisés, (2) Plateforme en tant que service PaaS (Platform as a Service), où le matériel, l'hébergement et le *framework* d'application sont dématérialisés, (3) Infrastructure en tant que service IaaS (Infrastructure as a Service) et (4) Matériel en tant que service HaaS (Hardware as a Service), où seul le matériel (serveurs) est dématérialisé dans ces deux derniers cas. La figure (I.4) montre le modèle classique et les différents modèles de service de *cloud*.

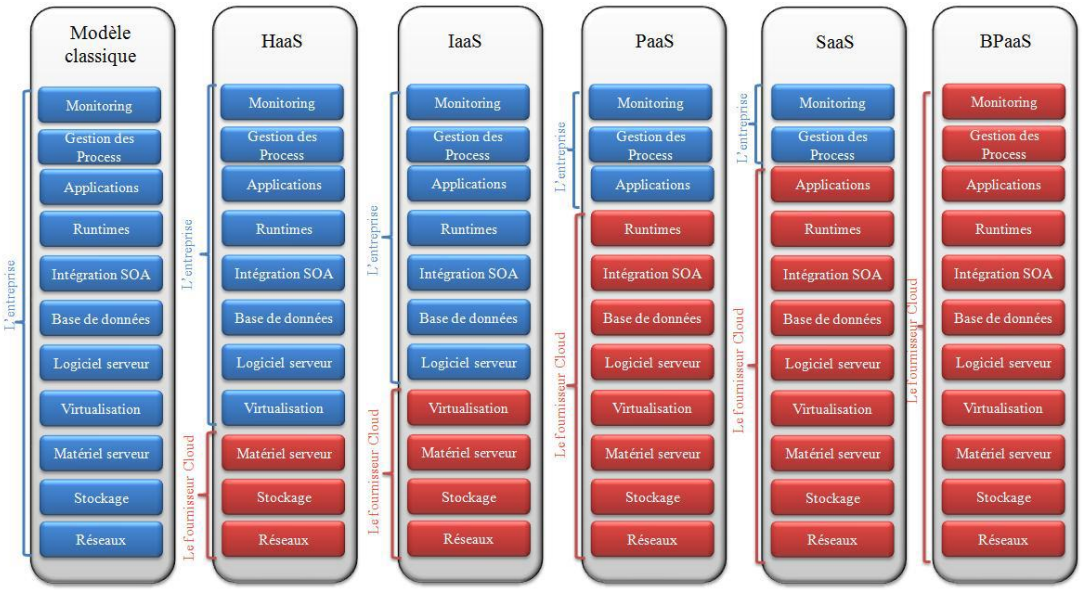


Figure I.4. Les services XaaS du *cloud computing* [71]

- **Software as a Service (SaaS)**

Le SaaS offre aux utilisateurs des applications sous forme de services en ligne déjà déployés dans le *cloud*. Cette couche est gérée par le fournisseur de SaaS de façon transparente pour les utilisateurs. Ces derniers ne savent ni où, ni comment ces applications sont déployées. Ils ne payent pas pour les posséder, mais plutôt pour les utiliser. Ils peuvent accéder à ces applications à tout moment via internet par le biais de n'importe quel dispositif connecté, tel que les ordinateur portables, tablettes, smartphones. Les applications sont utilisées soit directement via l'interface disponible, soit via des API fournies (souvent réalisées grâce aux Web Services ou à l'architecture REST (*RE presentational State Transfer*)). Les principales applications actuelles de ce modèle sont les applications de gestion de la relation client (CRM : *Customer Relationship Management*), la vidéo conférence, les communications unifiées, les outils collaboratifs, la messagerie. Les principaux fournisseurs de cette catégorie sont Salesforce.com (logiciels CRM) et Google (Gmail, Google Apps).

- **Business Process as a Service (BPaaS)**

Contrairement au SaaS, qui concerne les applications logicielles, le BPaaS cible les processus métiers et les ressources sont mutualisées entre les différents utilisateurs. Il s'intègre au-dessus des applications. Ce service est important pour les entreprises, car il permet l'automatisation et l'orchestration des flux d'actions et la supervision de ces flux. En tant que service de *cloud*, le modèle BPaaS est accessible via les technologies Internet. Des exemples de

BPaaS sont : la gestion des voyages, le paiement en ligne, la gestion financière, etc. Les fournisseurs de ce service sont : IBM, Atos, Wipro et autres.

- **Platform as a Service (PaaS)**

Ce type de *cloud* est plus orienté pour servir des développeurs d'applications. Il offre une plateforme entièrement configurée et gérée, sur laquelle l'utilisateur peut développer, tester et exécuter ses applications. Le déploiement des solutions PaaS est automatisé et évite à l'utilisateur d'avoir à acheter des logiciels ou d'avoir à réaliser des installations supplémentaires. Le PaaS offre une grande flexibilité, permettant notamment de tester rapidement un prototype ou encore d'assurer un service informatique sur une période de courte durée. Il favorise la mobilité des utilisateurs, puisque l'accès aux données et aux applications peut se faire à partir de n'importe quel périphérique connecté. Les principaux fournisseurs du PaaS sont SalesForce.com (Force.com), Google (Google App Engine), Microsoft (Windows Azure), Facebook (Facebook Platform).

- **Hardware as a Service (HaaS)**

Dans ce cas, le fournisseur de *cloud* loue essentiellement du matériel physique (par exemple, serveur, stockage). Les utilisateurs accèdent au HaaS via Internet pour installer et configurer les serveurs loués. Ils ont le contrôle sur le système d'exploitation et la pile logiciel/matériel. Les communautés de recherche louent du HaaS pour leurs applications et les configurent selon leurs besoins. Par conséquent, les *workflows* ou applications orientées calculs-intensifs et/ou traitement intensifs de données [10] [11], qui étaient traditionnellement exécutés sur des systèmes HPC, peuvent maintenant être exécutés dans le *cloud*. Parmi les principaux fournisseurs qui offrent du HaaS, on peut citer Baremetalcloud [12], Softlayer [13] et Grid5000 [14].

- **Infrastructure as a Service (IaaS)**

L'IaaS permet la mise à disposition des ressources d'infrastructure telles que des capacités de calcul, des moyens de stockage, du réseau sous forme de services publics. L'IaaS est similaire au HaaS, mais les ressources offertes sont virtualisées. Les utilisateurs d'IaaS sont libérés des charges causées par la possession, la gestion ou le contrôle du matériel sous-jacent. Par conséquent, ils n'ont pas accès directement aux machines physiques, mais indirectement à travers les machines virtuelles (MVs). Les utilisateurs ont la plupart du temps un contrôle presque complet sur les MVs qu'ils louent: ils peuvent choisir des images de systèmes d'exploitation préconfigurées par le fournisseur, ou bien des images de machines personnalisées contenant leurs propres applications,

bibliothèques et paramètres de configuration. Les utilisateurs peuvent également choisir les différents ports d'Internet à travers lesquels les machines virtuelles seront accessibles, etc. Les utilisateurs peuvent héberger et exécuter des logiciels, des applications quelconques, ou encore stocker des données sur l'infrastructure et ne paient que les ressources qu'ils consomment.

La particularité de l'IaaS est de fournir un approvisionnement en ressources informatiques, qui soit dynamique, flexible, extensible et qui possède de bonnes propriétés de passage à l'échelle pour répondre aux besoins spécifiques des utilisateurs. Le *cloud* est vu ainsi comme une source infinie de ressources de calcul, de stockage et de réseau accessibles en un modèle de paiement à l'usage. Internet devient une place de marché où l'infrastructure informatique est distribuée, et consommée en tant que marchandise, selon le modèle illustré sur la figure (I.5).

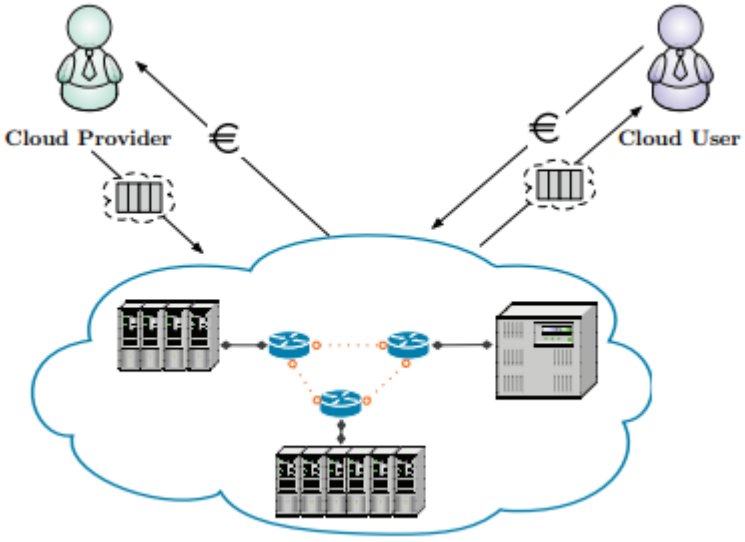


Figure I.5. Modèle de distribution de l'infrastructure en tant que service [1].

Il existe de nombreux fournisseurs commerciaux et open-source de l'IaaS. Parmi les plateformes commerciales, nous pouvons citer : Amazon EC2 [15], Rackspace [16], et d'autres. Dans les communautés open-sources, plusieurs plateformes ont été développées, à savoir : Eucalyptus [17], Nimbus [18],Open Nebula[19]et d'autres.

Notre travail se concentre sur l'Infrastructure as a Service (IaaS), car ils fournissent tous les services de base nécessaires pour les applications de *workflows*, y compris les machines virtuelles, les processeurs, le stockage et l'accès aux services réseau. En outre, l'IaaS est une technologie riche en fonctionnalité et mature actuellement.

Néanmoins, les services PaaS et SaaS peuvent être aussi utilisés pour les *workflows*. Le PaaS peut exposer un environnement de développement et d'exécution de haut niveau permettant de construire et de déployer les *workflows* sur l'infrastructure *cloud*. Les fournisseurs SaaS offrent aux utilisateurs finaux des solutions logicielles standardisées qui pourraient être intégrées dans leurs *workflows* existants [6].

I.2.3 Modèles de déploiement

On peut distinguer quatre types principaux de modèles de déploiement pour le *cloud computing* : le *cloud* privé, *le cloud* public et le *cloud* hybride, *le cloud* communautaire [87]**(**figure I.6).

I.2.3.1 Le cloud privé

L'infrastructure d'un *cloud* privé n'est utilisée que par un client unique. Elle peut être gérée par ce client ou par un prestataire de service et peut être située dans les locaux de l'entreprise cliente ou bien chez le prestataire, le cas échéant. L'utilisation d'un *cloud* privé permet de garantir, par exemple, que les ressources matérielles allouées ne seront jamais partagées par deux clients différents.

I.2.3.2 Le *cloud* public

L'infrastructure d'un *cloud* public est accessible publiquement ou pour un large groupe industriel. Son propriétaire est une entreprise qui vend de l'informatique en tant que service.

I.2.3.3 Le *cloud* hybride

L'infrastructure d'un *cloud* hybride est une composition de deux ou trois des types de *clouds* précédemment cités. Les différents *clouds* qui la composent restent des entités indépendantes à part entière, mais sont reliés par des standards ou par des technologies propriétaires qui permettent la portabilité des applications déployées sur les différents *clouds*. Une utilisation type de *cloud* hybride est la répartition de charge entre plusieurs *clouds* pendant les pics du taux d'utilisation.

En plus de ces trois modèles de déploiement de *cloud*, on trouve dans les littératures la notion de *cloud* communautaire.

I.2.3.4 Le *cloud* communautaire

L'infrastructure d'un *cloud* communautaire est partagée par plusieurs organisations indépendantes et est utilisée par une communauté qui est organisée autour des mêmes besoins,

vis-à-vis de son utilisation. Par exemple, dans le projet Open Cirrus, le *cloud* communautaire est partagé par plusieurs universités dans le cadre d'un projet scientifique commun. Son infrastructure peut être gérée par les organisations de la communauté qui l'utilise ou par un tiers et peut être située, soit au sein des dites organisations, soit chez un prestataire de service.

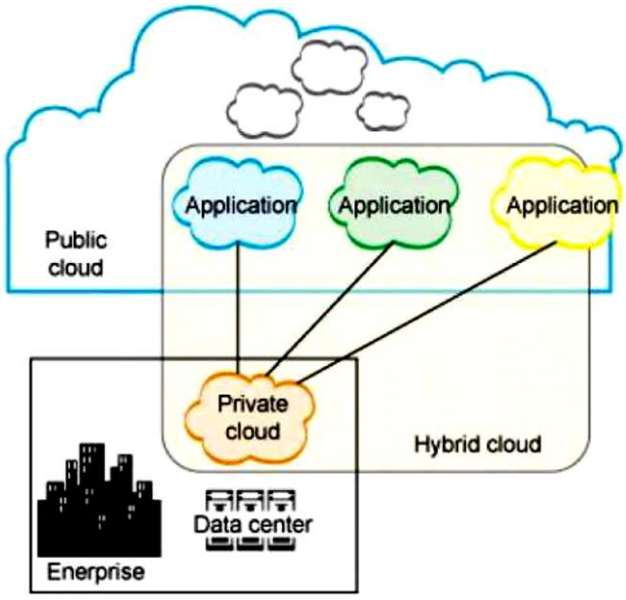


Figure I.6. Modèles de déploiement de *cloud computing* [20].

I.2.4 Acteurs du *cloud computing*

Selon l'architecture de référence du *cloud computing* décrite par le NIST [21], on distingue cinq acteurs principaux comme le montre la figure (I.7).

- ***Consommateur (cloud Consumer) :*** une personne ou une organisation qui utilise un service fourni par un fournisseur. Dans cette thèse, nous utilisons aussi le terme utilisateur pour désigner un consommateur.
- ***Fournisseur (cloud Provider) :*** une personne, une organisation ou une entité chargée de rendre un service à la disposition des parties intéressées.
- ***Courtier (cloud broker) :*** une entité qui gère l'usage, la performance et l'approvisionnement de services, et qui négocie les relations entre les fournisseurs et les consommateurs.

➢ ***Auditor (cloudauditor) :*** une entité qui peut procéder à une évaluation indépendante des services de cloud telle que la performance et la sécurité du cloud.

➢ ***Carrier (cloud carrier) :*** un intermédiaire qui offre la connectivité et le transport des services des fournisseurs aux consommateurs.

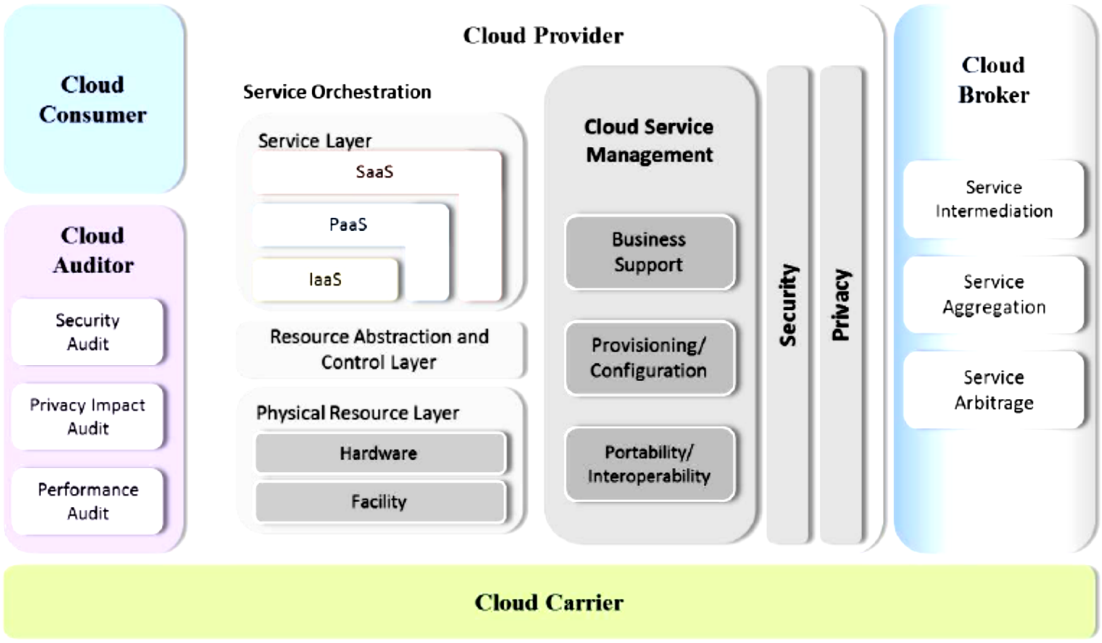


Figure I.7. Modèle de référence conceptuel du *cloud* NIST 2011.

I.2.5 Challenges de recherche en environnement de *cloud computing*

Même si certaines des caractéristiques essentielles du *cloud computing* ont été réalisées par des efforts commerciaux et universitaires, de nombreux problèmes existants n'ont pas été pleinement pris en compte, et d'autres nouveaux défis continuent d'émerger [6], Dans cette section, nous résumons quelques enjeux de recherche dans le *cloud computing*.

➢ **Migration des données entre les environnements non standards**

La plupart des fournisseurs de services de *cloud computing* utilisent des applications *cloud* propriétaires. Ces applications ne sont pas interopérables, ce qui rend très difficile pour les utilisateurs de faire passer leurs données à un autre fournisseur de *cloud* ou de revenir à leur machines.

- **Sécurité de données et confidentialité**

Dans le *cloud computing*, les données doivent être transférées entre les dispositifs de l'utilisateur et les Datacenter des fournisseurs de services de *cloud computing*, ce qui les rendra cible facile pour les pirates. La sécurité des données et la confidentialité doivent être garanties, que ce soit sur le réseau ou encore dans les Datacenter de *cloud* où elles seront stockées.

- **Migration de machines virtuelles**

La virtualisation peut offrir des avantages importants dans le *cloud computing* en permettant la migration de machine virtuelle pour équilibrer la charge de travail entre les Datacenter. Elle permet un approvisionnement robuste et très réactif dans les Datacenter. Les principaux avantages de la migration de MV est d'éviter les points chauds (hot spots); Toutefois, cela n'est pas simple à réaliser. Actuellement, la détection de points chauds et l'initiation d'une migration manque de souplesse pour répondre aux changements brusques de charge de travail.

- **Consolidation de serveurs**

La consolidation de serveurs est une approche efficace pour maximiser l'utilisation des ressources, tout en minimisant la consommation d'énergie dans un environnement de *cloud computing*. La technologie de migration de MV est souvent utilisée pour consolider les machines virtuelles se trouvant sur plusieurs serveurs sous-utilisés sur un seul serveur, de sorte à mettre ces derniers en mode d'économie d'énergie. Le problème de la consolidation optimale des serveurs est un problème d'optimisation NP-complet. Cependant, les activités de consolidation de serveurs ne devraient pas affecter les performances de l'application. Il est connu que l'utilisation des ressources de machines virtuelles individuelles peut varier au cours du temps [22].

- **Gestion de l'énergie**

Améliorer l'efficacité énergétique est un autre enjeu majeur dans le *cloud computing*. En 2006, les Datacenter aux États-Unis ont consommé plus de 1,5% de l'énergie totale produite dans l'année, et le pourcentage devrait croître de 18% par an [24]. Ainsi, les fournisseurs d'infrastructure doivent prendre des mesures pour réduire la consommation d'énergie. L'objectif est non seulement de réduire le coût de l'énergie dans les Datacenter, mais aussi pour répondre aux réglementations gouvernementales et aux normes environnementales.

- **Ordonnancement**

L'ordonnancement est un enjeu important qui influence considérablement les performances de l'environnement de *cloud computing*. Le processus d'ordonnancement est appliqué sur les différents niveaux selon la nature de service de *cloud* : (1) orienté-marché (l'ordonnancement au niveau de service et au niveau de tâche); (2) non orienté-marché (l'ordonnancement au niveau de MVs) [23]. L'ordonnancement au niveau de service est statique et concerne une partie de la couche de gestion des ressources. Le principale critère à optimiser au ce niveau c'est le profil. L'ordonnancement au niveau de tâche d'ordonnancement est dynamique, adapté au changement de *cloud*. Son objectif est d'optimiser l'assignement selon les contraintes de *QoS* de chaque tâche et de chaque client, tout en minimisant le prix total d'exécution et le coût en temps. L'ordonnanceur au niveau de tâche est dédié à un seul data center (*cloud*), et il est incapable de gérer les ressource d'un autre *cloud* d'un autre fournisseur. Le processus de correspondance entre la tâche et la MV se fait par un courtier. L'ordonnancement au niveau machine virtuelle c'est la couche d'ordonnancement la plus basse, utilisée pour fournir l'ensemble des MVs demandés par la tâche dans l'ordonnancement au niveau de tâches. Le but de ce niveau est de trouver un meilleur ordonnancement des machines virtuelles sur les hôtes qui composent les data center (*cloud*) [23]. La figure (I.8) montre les différents niveaux d'ordonnancement.

Le problème d'ordonnancement a été formellement définit par [85] sur le *cloud* par les équations (I.1), (I.2) et (I.3) comme suit :

$$Critére\ \text{à}\ optimiser = \left(\sum_i^N \sum_j^J Metrique_{ij} \times x_{ij}\right) \times Temps \quad \text{(I.1)}$$

$$Critére\ de\ contrainte\ \left(\left(\sum_i^N \sum_j^J Metrique_{ij}\ \times\ x_{ij}\right)\ \times Temps\ \leq\ la\ valeur\ de\ contrainte\right) \quad \text{(I.2)}$$

$$\sum_j^J x_{ij} = 1\ pour\ chaque\ i, 1\ \leq i \leq N \quad \text{(I.3)}$$

Où :

- *Metrique i,j* : représente la valeur de métrique du critère à optimiser, où un membre *i* de *N* est assigné à un membre *j* de *J* .
- *Temps* : la durée de l'assignement.
- *Valeur de contrainte* : la borne limitée de critère à optimiser.
- $x_{i,j}$: représente les contraintes de placement, où chaque membre *j* de l'ensemble *J* peut être assigné à un seul membre de *i* de *N*.

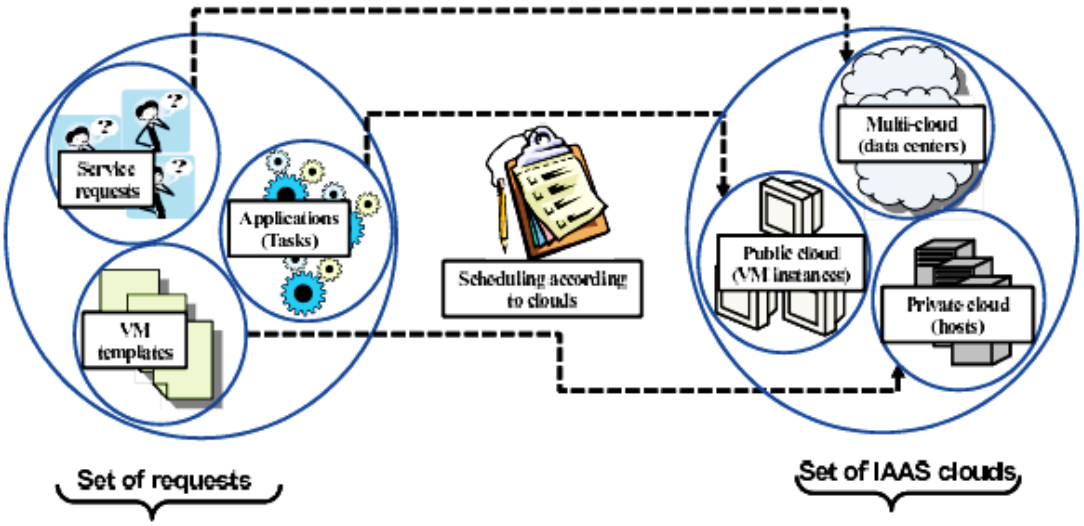


Figure I.8. Différents niveaux d'ordonnancement du cloud [85].

Dans cette thèse, notre intérêt porte sur le deuxième niveau d'ordonnancement, où les applications de l'utilisateur sont souvent exprimées sous forme de *workflows*.

I.3 Concepts de base des *workflows* et des systèmes de gestion de *workflows*

De nos jours, les systèmes de *workflow* se présentent comme une réponse technologique idéale pour répondre aux objectifs fixés par une activité de réingénierie. La qualité de la spécification, de la validation et de la mise en œuvre des systèmes *workflows* est donc un aspect essentiel de l'ingénierie des systèmes d'information pour garantir des systèmes fonctionnellement fiables, flexibles et maintenables, sûrs de fonctionnement même dans un environnement d'exécution ouvert.

Deux grandes catégories d'usages utilisent la notion de *workflow* : les protocoles expérimentaux, dans des domaines tels que la biologie, l'astronomie, la physique, la neuroscience, la chimie, etc. (*workflows* scientifiques) et les chaînes de traitement pratiquées dans des domaines commerciaux, financiers, pharmaceutiques (processus métiers). Elles donnent lieu à plusieurs pistes de recherche diverses, mais cependant connexes. Dans notre recherche, nous traitons plus particulièrement les *workflows* scientifiques [6].

I.3.1 Définitions de *workflow*

La définition générale de *workflow* (ou la gestion automatique du flux de travail) désigne un travail coopératif impliquant un nombre limité de personnes devant accomplir, en un temps limité, des tâches articulées autour d'une procédure définie et ayant un objectif global [25].

Cependant diverses définitions ont été proposées, selon les catégories d'usages:

Les définitions dans [6] [25] [26] concernent les *workflows* scientifiques et les définitions [27] [28] concernent les business *process*.

Un *workflow* est composé d'un ensemble de tâches (traitements) organisées selon un ordre logique, afin de réaliser un traitement global, complexe et pertinent sur un ensemble de données sources. [6].

Les *workflows* scientifiques de grande taille sont souvent de calcul intensif, il est souhaitable de répartir les tâches entre plusieurs ressources, afin d'optimiser les temps d'exécution. En tant que tel, les *workflows* impliquent souvent des calculs répartis sur des clusters, des grilles, et d'autres infrastructures informatiques. Récemment, les *clouds computing* sont évalués comme une plateforme d'exécution de *workflows* [29]. L'exécution d'un *workflow* est gérée par un Système de Gestion de *Workflow (SGWf).*

I.3.2 Définition d'un système de gestion de *workflows*

Un SGWf représente un système qui définit, implémente et gère l'exécution de *workflows* à l'aide d'un environnement logiciel fonctionnant avec un ou plusieurs moteurs de *workflows* et capable d'interpréter la définition d'un processus, de gérer la coordination des participants et d'invoquer des applications externes. L'objectif de la gestion de *workflow* est de s'assurer que des activités appropriées sont exécutées par les bons services, au bon moment [81].

I.3.3 Systèmes de gestion de workflows pour le *clouds*

Un système de *workflow cloud SwinDeW-C* (*Swinburne Decentralised Workflow for cloud*) a été proposé par [30], inclure quatre partie principal : (i) les services de *cloud* (ii) les agents d'exécution de *workflow cloud* (iii) catalogue des services (iiii) interface d'utilisateur. Un autre travail proposé par [31], qui explore l'exécution des *workflows* scientifiques sur les *clouds*, en utilisant une application d'astronomie qui s'appelle Montage.

Un des principaux modules de tous les systèmes de gestion de *workflow* est l'ordonnancement. Ce dernier est un processus qui "mappe" et gère l'exécution de tâches interdépendantes sur des ressources distribuées. Il alloue des ressources appropriées pour les tâches de *workflow* de sorte que l'exécution puisse être achevée, tout en satisfaisant les objectifs et contraintes imposés par l'utilisateur. Un ordonnancement adéquat peut avoir un impact significatif sur la performance du système. En général, le problème de *mapping* de tâches sur les services distribués appartient à une classe de problèmes connus comme NP-complets [6].

I.4 Conclusion

Dans ce chapitre, les aspects fondamentaux et les concepts qui sont présents, tout en faisant face aux domaines de *cloud computing*, du *workflow* et des intérêts du *cloud* pour les *workflows*. Nous avons exploré divers défis de recherche dans le *cloud*. Un de ces défis porte sur l'ordonnancement de *workflows*, qui présente un élément essentiel des systèmes de gestion de *workflows*. Dans le chapitre suivant nous allons présenter une synthèse sur les algorithmes méta-heuristiques pour l'ordonnancement des problèmes multi-objectifs et les principaux concepts, définitions et notations liées au domaine d'optimisation multi-objectif.

Chapitre II

L'ordonnancement des workflows dans le cloud :état de l'art

Plan

II.1 Introduction

Dans la littérature, les méthodes utilisées pour résoudre les problèmes multi-objectifs se divisent en deux classes : les méthodes exactes et les méthodes qui utilisent les méta-heuristiques. La différence entre ces deux méthodes est que les méthodes exactes trouvent dans la plupart du temps des solutions exactes pour le problème mais consomme un temps de calcul énorme et il ne peut se trouver qu'une solution pour un problème multi-objectif, ce qui est le contraire pour les méthodes de résolutions qui utilisent les méta-heuristiques, essentiellement les algorithmes évolutionnaires qui trouvent un ensemble approximatif de solutions Pareto avec un temps acceptable.

Dans ce chapitre, nous allons faire une synthèse des méta-heuristiques pour l'ordonnancement des problèmes multi-objectifs et les principaux concepts, définitions set notations liées au domaine d'optimisation multi-objectif, notamment, la relation de dominance, l'optimalité et le front de Pareto ainsi que les méthodes de résolution.

II.2 L'optimisation multi-objectif

Dans cette section, nous présentons les définitions de base des problèmes d'optimisations multi-objectives et les différentes stratégies de résolutions de ces problèmes.

II.2.1 Définitions de base

Un problème d'optimisation multi-objectif est un problème de la forme suivante :

$$(\text{PMO})\begin{cases} Optimiser \mathrm{F(x)} = \big(\mathrm{f1(x)}, \mathrm{f2(x)}, \ldots, \mathrm{fn(x)}\big) \\ sous x \in D \end{cases}$$

Où: n ($n \geq 2$) est le nombre d'objectifs , $x = (x1, x2, \ldots, xk)$ est le vecteur représentant les variables de décision, D représente l'ensemble des solutions réalisables et chacune des fonctions fi(x) est à optimiser, c'est-à-dire à minimiser ou à maximiser. Sans perte de généralité nous supposerons par la suite que nous considérons des problèmes de minimisation. La figure (II.1) présente la liaison entre l'espace décisionnel et l'espace objectif.

Contrairement à l'optimisation mono-objectif, la solution d'un problème multi-objectif n'est pas unique, mais est un ensemble de solutions non dominés, connu comme l'ensemble des solutions Pareto Optimales (PO), l'image de cet ensemble dans l'espace objectif représente le front de Pareto [32].

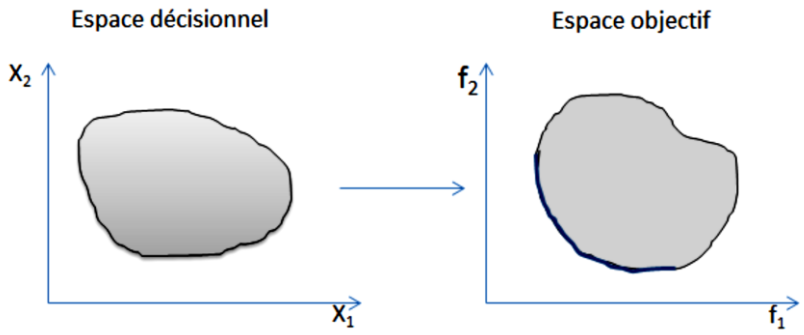


Figure II.1. Espace décisionnel et espace objectif d'un PMO (cas de deux variables de décision x_1 et x_2 et de deux fonctions objectif f_1 et f_2) [6].

II.2.2 Notions de dominances et d'optimalité

Pour les problèmes de minimisation, les concepts de Pareto sont définit comme suit (les définitions sent les même pour les problèmes de maximisation):

$$\forall\, i \in \{1, \dots, n\}, Z_i \leq Z'_i$$
$$\forall\, j \in \{1, \dots, n\}, Z_j \leq Z'_j$$

- **Dominance de Pareto :** Un vecteur objectif z ∈ Z domine un vecteur objectif z′∈ Z. si les deux conditions suivantes sont vérifiées :

$$\forall\, i \in \{1, \dots, n\}, Z_i \leq Z'_i$$
$$\forall\, j \in \{1, \dots, n\}, Z_j \leq Z'_j$$

Si z domine z′, cette relation sera notéez $\prec z'$. Par extension, une solution $x \in X$ domine une solution x′∈ X, notée$x \prec x'$, ssif(x)<f(x′) .Le concept généralement utilisé est l'optimalité de pareto.

- **Optimalité de Pareto**

  - **Difinition1 : Solution Pareto optimale :** Une solution Pareto optimale peut être considérée comme optimale du fait qu'il n'est pas possible de trouver une solution permettant d'améliorer la valeur d'un objectif sans dégrader celle d'au moins un autre objectif. Il constitue l'ensemble des solutions Pareto optimales (ensemble de solutions non-dominées).

  - **Définition 2 : Ensemble Pareto optimal :** Etant donné un PMO (f,x) l'ensemble Pareto optimal est défini comme suit : $P^* = \{x \in X / \nexists\, x' \in X, x' \prec x\}$.

  - **Définition 3 : Front Pareto :** Etant donné un PMO (f,x) et son ensemble Pareto optimal P^*, le front Pareto est défini comme suit:$PF^* = F(x), x \in P^*$.(voir figure II.2)

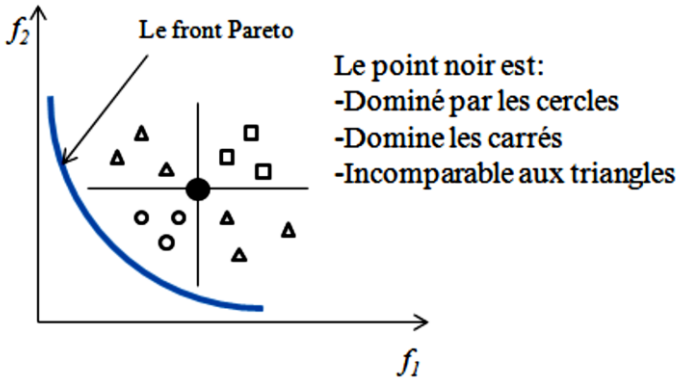


Figure II.2. Relation de dominance (cas de deux objectifs à minimiser) [6].

II.2.3 Choix de la méthode d'aide à la décision

La résolution d'un problème multi-objectif menant à la détermination d'un ensemble de solutions Pareto, il est nécessaire de faire intervenir l'humain à travers un décideur, pour le choix final de la solution à garder.

Ainsi, avant de se lancer dans la résolution d'un problème multi-objectif, il faut se poser la question du type de méthode d'optimisation à utiliser. En effet, on peut répartir les méthodes de résolution de problèmes multi-objectifs en trois familles, en fonction du moment où intervient le décideur. Ainsi nous pouvons trouver les familles suivantes [33] :

– **Les méthodes d'optimisation a priori (décideur → recherche):** dans ce cas, le compromis que l'on désire faire entre les objectifs a été défini avant l'exécution de la méthode. Ainsi une seule exécution permettra d'obtenir la solution recherchée. Cette approche est donc rapide, mais il faut cependant prendre en compte le temps de modélisation du compromis et la possibilité pour le décideur de ne pas être satisfait de la solution trouvée et de relancer la recherche avec un autre compromis.

– **Les méthodes d'optimisation a posteriori (recherche →décideur):** dans cette famille de méthodes, on cherche à fournir au décideur un ensemble de bonnes solutions bien réparties. Il peut ensuite, au regard de l'ensemble des solutions, sélectionner celle qui lui semble la plus appropriée. Ainsi, il n'est plus nécessaire de modéliser les préférences du décideur (ce qui peut s'avérer être très difficile), mais il faut en contre - partie fournir un ensemble de solutions bien réparties, ce qui peut également être difficile et requérir un temps de calcul important (mais ne nécessite pas la présence du décideur).

Dans le cadre de notre travail, nous nous plaçons dans le cas de l'approche *a posteriori*. Dans ce type d'approche, deux phases sont distinguées : la phase d'optimisation et la phase d'aide

à la décision. Seule la première phase sera traitée dans notre travail, et sera considérée comme la résolution de notre problème.

–**Les méthodes d'optimisation Interactive (décideur ↔ recherche):** ici, le décideur intervient dans le processus de recherche de solutions en répondant à différentes questions afin d'orienter la recherche. Cette approche permet donc de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche.

II.3 l'état de l'art des différents *frameworks* d'optimisation Multiobjectifs (MOPs)

Dans cette section, plusieurs algorithmes de l'état de l'art dans des différents *frameworks* d'optimisation multi-objectifs sont décrits [34].

II.3.1 Framework basé préférence

Les *frameworks* basés préférence sont les méthodes classiques de traitement des MOPs. L'idée de base de ce *framework* est d'agréger les multiples objectifs contradictoires d'un MOP en un problème d'optimisation mono-objectif ou d'utiliser les connaissances de préférence des problèmes afin que les optimiseurs puissent se concentrer sur l'optimisation de certains objectifs.

L'approche la plus fondamentale dans ce *framework* est d'agréger les objectifs contradictoires dans un seul objectif par le biais d'une méthode de somme pondérée. Cette méthode combine tous les objectifs contradictoires en multipliant chaque objectif avec une valeur de poids prédéfinie. La méthode de métrique pondérée ou de Tchebychev pondérée est une autre approche qui combine les objectifs dans un seul objectif. Dans cette dernière, l'objectif est de minimiser les métriques de distance pondérées, où la métrique distance mesure la distance d'une solution à une solution idéale. Le principal inconvénient de ce *framework* est qu'il ne parvient pas à atteindre le but commun d'optimisation multi-objectif qui est d'obtenir un ensemble de compromis et de solutions diverses en un seul passage des imulation. La recherche dans ce type de *framework* se concentre principalement sur l'étude de la façon d'utiliser au mieux l'information de préférence dans l'optimisation.

II.3.2 Framework basé dominance

Les algorithmes dans ce *framework* optimisent tous les objectifs contradictoires de MOPs simultanément par l'attribution d'un fitness à chaque solution. Cette idée a été suggérée par

Goldberg [35]. Notons que cette méthode n'a pas utilisé la simulation pour prouver sa pertinence dans la manipulation des MOPs. Parmi les premiers MOEAs remarquables dans ce *framework*, nous citons l'algorithme génétique multi-objectif (MOGA), a été proposé par Fonseca et Fleming [36]. Dans MOGA, deux étapes importantes ont été conçues pour déterminer le fitness d'une solution. En premier, le fitness d'une solution est calculée en fonction du nombre d'autres solutions qu'elle domine. Ce fitness est utilisé pour déterminer le rang des solutions. Deuxièmement, le fitness des solutions dans le même rang est partagé par un mécanisme de partage de fitness. L'utilisation de ces deux étapes rend MOGA capable de maintenir un ensemble de solutions non-dominées en une seule exécution de la simulation.

Un autre algorithme proposé par Srinivas et Deb dans [37], l'algorithme proposé un mécanisme de classement qui classe les solutions selon le niveau de domination. Le premier niveau comprend toutes les solutions non dominées. Ensuite, les solutions marquées comme premier niveau sont ignorées. La deuxième série de solutions non-dominées dans la population sont identifiées et marquées comme le deuxième niveau (front). Ceci continue jusqu'à ce que le classement n'a plus de solutions stockées dans la population. La diversité est maintenue grâce à une méthode de partage de fitness. Parmi les inconvénients de cet algorithme est la non-élitiste.

Parmi les GA adoptant une stratégie élitiste, nous citons l'algorithme SPEA (*Strength Pareto Evolutionary Algorithm*), proposé par [38], où le passage d'une génération à l'autre commence par la mise à jour des différentes sauvegardes. Tous les individus non dominés sont sauvegardés et les individus dominés, déjà présents, sont éliminés. Cet algorithme utilise la notion de compte de dominance pour calculer la fitness d'un individu. Une évolution de cet algorithme est disponible dans par [39]. Cette évolution a une plus grande complexité, mais présente de meilleures performances.

Dans [40], Deb et al, présentent l'algorithme *NSGA*-II qui conserve également un ensemble de solutions archivées depuis le début de l'évolution. Au lieu de stocker les solutions non dominées seulement comme *SPEA*, *NSGA*-II stocke tous les parents (N solutions) et les enfants (N solutions) des solutions. Par la suite, un tri non-dominé est appliqué aux solutions d'archivage entières (2N solutions).Les solutions sont classées selon le rang de dominance. Les meilleures solutions non-dominées sont marqués comme premier rang. Les meilleurs deuxièmes solutions non-dominées sont marquées comme second rang, et ainsi de suite. Les meilleures N solutions avec les rangs inférieurs sont choisis comme solutions parents et seront soumises à l'évolution de la prochaine génération. Comme il faut que N solutions soient sélectionnées comme des solutions parents, certaines des solutions à rang particulier ne convient pas d'être dans la population parent. Dans ce cas, une mesure de distance de *crowding* est utilisée pour déterminer les solutions qui ont moins

de distances pour devenir des solutions parents. Comparé à la *NSGA*, *NSGA*-II a une complexité de calcul inférieure, utilise une approche élitiste, et n'a pas besoin de spécifier un paramètre de partage. Actuellement, *NSGA*-II est l'un des MOEAs les plus célèbres qui a été mis en œuvre pour résoudre de nombreux problèmes du monde réel et de servir comme un algorithme de base pour MOEAs. Cependant, la performance de l'optimisation de *NSGA*-II est moins bonne dans les problèmes avec plus de trois objectifs. Ceci est parce que ses mécanismes de classement et de *crowding* sont inefficaces pour différencier la supériorité des solutions à des problèmes à multiples objectifs (many-objective).

Le *framework* à base de dominance de l'optimisation multi-objective est devenu l'un des principaux domaines de recherche de la dernière décennie. La possibilité d'obtenir un ensemble de solutions de compromis en un seul passage de simulation détermine la pertinence de cette approche pour l'optimisation multi-objectif. De plus, la diversité des solutions peut être préservée en tenant compte de la distribution des solutions dans une population maintenue. Cependant, un inconvénient majeur de cette approche est que la pression de sélection est affaiblie avec l'augmentation du nombre de fonctions objectifs. Par conséquent, cette approche ne convient que pour résoudre les problèmes avec deux ou trois fonctions objectifs ou le cas de notre problème. En outre, il est nécessaire de déterminer le réglage d'un paramètre de partage dans certains systèmes de préservation de diversité.

II.3.3 Framework basé sur la décomposition

Le *framework* basé sur la décomposition, décompose un MOP en plusieurs sous-problèmes, par la suite l'algorithme optimise tous les sous-problèmes simultanément. La dominance de Pareto est partiellement appliquée ou totalement éliminées dans ce cadre. Louis et Thomas dans [41] propose une recherche locale à deux phases pour trouver un bon ensemble approximatif de solutions non dominées. La première phase consiste a généré une solution initiale en optimisant un seul objectif, à partir de cette solution, une recherche de solutions non dominées exploitant une séquence de différentes formulations du problème en fonction des agrégations des objectifs. Les auteurs ont proposé dans [42] un algorithme évolutionnaire multi-objectifs (MOEA) basé sur la décomposition (MOEA/D), qui décompose le problème donné en un certain nombre de problèmes d'optimisation scalaires les optimise simultanément.

II.4 L'ordonnancement des tâches

La performance des applications est reliée à la bonne gestion de l'utilisation des ressources, donc il est nécessaire de proposés des algorithmes et des outils efficace pour gérer une utilisation efficace des ressources. Plusieurs algorithmes d'ordonnancements ont été proposés dans la plate-forme de type *cloud*. Le travail de Bessai.K dans [43] a été proposé une classification, des algorithmes existants, en trois catégories selon les types de tâches constituant l'application : tâches indépendantes, tâches dépendantes (DAG) et les DAGs s'exécutant en parallèle.

II.4.1 Ordonnancement de tâches indépendantes

La plupart des études ont été proposé dans leurs études des tâches indépendantes par rapport aux deux autres types d'applications, cependant très rare des travaux utilisent des plates-formes hétérogènes. Les heuristiques d'ordonnancement dynamique des tâches indépendantes et séquentielles proposés dans [44] sont classées en deux classes :

II.4.1.1 Des heuristiques d'ordonnancement en ligne

Les tâches qui arrivent dans le système sont ordonnancées et affectées à des ressources dès leurs arrivées.

II.4.1.2 Des heuristiques d'ordonnancement par lot (*batch scheduling*)

Les tâches ne sont pas exécutées à leurs arrivées mais regroupées dans des ensembles distincts et les dates de leurs ordonnancements sont déterminées ultérieurement.

Les tâches parallèles sont aussi étudier par des nombreuses heuristiques dynamiques, L'objectif de ces heuristiques est essentiellement l'augmentation du taux d'utilisation des ressources et la minimisation du temps d'attente des tâches [43].

II.4.2 Ordonnancement de tâches dépendantes

Les applications composées de tâches dépendantes visent à optimiser leurs temps d'exécution en se focalisant sur des applications définies par des tâches liées par des contraintes de précédence (DAGs).

L'ordonnancement de graphes de tâches composées détaches séquentielles a été largement étudié et plusieurs algorithmes heuristiques ont été proposés. Ces algorithmes peuvent être classés en trois catégories [43]:

II.4.2.1 Les algorithmes d'ordonnancement de listes

Les algorithmes de cette classe commencent par le calcul de la priorité des tâches constituant l'application, ensuite ces algorithmes consiste à choisir une ressource pour tâches dans l'ordre de priorité tout en minimisant une fonction de récompense préalablement définie (par exemple, la date de fin d'une tâche).Ces heuristiques ont l'avantage d'être peu complexes par rapport aux autres types d'heuristiques et produisent des résultats similaires.

II.4.2.2 Les algorithmes d'ordonnancement de *clustering*

La première étape de ces algorithmes est de commencer de regrouper les tâches dans des groupes (cluster). La deuxième étape consiste à affecter ces tâches aux ressources disponibles. La contrainte à respecter est que toutes les tâches appartenant à un même groupe soient exécutées par une même ressource.

II.4.2.3 Les algorithmes d'ordonnancement de duplication de tâches

Les algorithmes de duplication sont des algorithmes NP-complet, capable de réduire les fais de communication par l'exécution d'une même tâche sur plusieurs ressources. Il existe plusieurs algorithmes basés sur d'ordonnancement de duplication de tâches basés sur les heuristiques. Le travail de [45] classe les algorithmes d'ordonnancement de duplication de tâches en deux catégories : l'ordonnancement avec duplication partielle (SPD) et l'ordonnancement avec double-duplication (SFD).

II.4.2.4 Les algorithmes d'ordonnancement méta-heuristiques

Les méta-heuristiques pour l'ordonnancement des taches sont des méthodes plus efficaces et largement utilisées et pour l'ordonnancement des graphes de tâches séquentielles. Les heuristiques basé sur les algorithmes génétiques produit des bons résultats par rapport aux autres heuristiques (*Recherche Tabou* [53], *Recuit Simulé* [52]), mais restent plus couteux en terme de complexité et temps de calcul. On trouve dans cette classification les travaux de recherche de

Yassa.S [6] [46] [47] qui utilisent l'algorithme *NSGAII* et l'algorithme génétique pour la résolution du problème d'ordonnancement des *workflows* dans le *cloud computing*.

II.4.3 Ordonnancement d'applications concurrentes

Les applications qui exécutent en parallèle (applications concurrentes) peuvent tomber dans un problème de famine. Ce problème produit par l'attente d'un certain temps d'application et n'a jamais accès aux ressources disponibles (indéfiniment retardées). Le *makespan* et le débit sont les critères les plus considérés par les applications qui partagent les mêmes ressources.

II.5 Méta-heuristique pour l'optimisation multi-objectif

Dans cette section, nous présentons la définition des algorithmes méta-heuristiques et ses différents concepts.

II.5.1 Les méta-heuristiques

Ces algorithmes sont plus complets et complexes qu'une simple heuristique, et permettent généralement d'obtenir une solution de très bonne qualité pour des problèmes issus des domaines de la recherche opérationnelle ou de l'ingénierie dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage [48].

Plusieurs classifications des méta-heuristiques ont été proposées, la plupart distinguent globalement deux familles : les méta-heuristiques à base de solution unique (S-méta-heuristiques) et celles à base de population de solutions (P-méta-heuristiques), comme illustré dans la figure (II.3). [49].

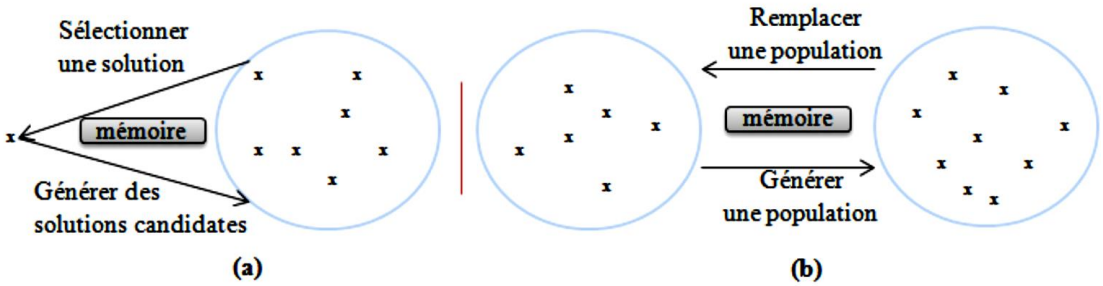


Figure II.3. Principes généraux d'une méta-heuristique à base de: (a) solution unique, (b) population [6].

II.5.1.1Concepts communs à tout type de méta-heuristique

Pour la conception de toute méta-heuristique, deux concepts fondamentaux doivent être considérés : la représentation des solutions manipulées par l'algorithme, et l'évaluation de ces solutions dans l'espace objectif [6].

- **Représentation de solution**

La conception de tout type de méta-heuristique nécessite une représentation (codage) d'une solution. La représentation joue un rôle majeur dans l'efficacité de toute méta-heuristique et constitue donc une étape de conception essentielle. La représentation doit être pertinente et adaptée au problème d'optimisation en question. En outre, le choix d'une représentation aura une influence considérable sur la façon dont les solutions seront manipulées par les opérateurs de recherche et lors de l'étape d'évaluation. On peut trouver plusieurs représentations pour un problème donné. Aussi, de nombreuses représentations peuvent être appliquées aux différents problèmes d'optimisation. Dans la littérature, quatre principales représentations (pour l'optimisation combinatoire) peuvent être distinguées : représentations basées sur des vecteurs de valeurs réelles, discrètes, binaires, ou des permutations. La figure(II.4) illustre un exemple de chaque représentation. Le vecteur de valeurs discrètes est couramment utilisé pour modéliser l'ordonnancement de *workflows* dans le *cloud computing.*

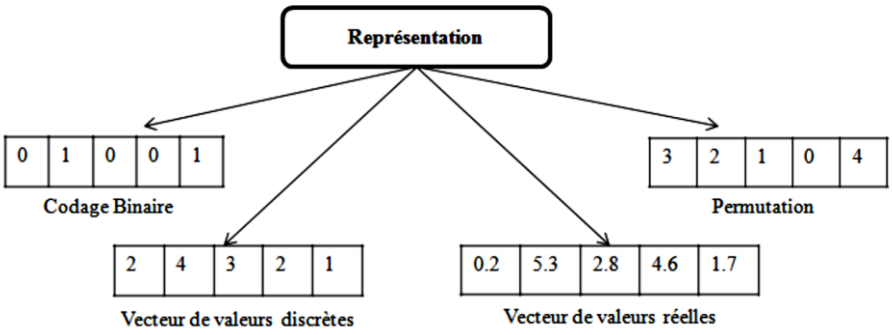


Figure II.4. Principaux codages pour des problèmes d'optimisation [6].

- **Evaluation**

L'étape d'évaluation correspond au calcul et l'affectation de valeurs objectives pour une solution donnée. Dans le cas de l'optimisation mono-objectif, une fonction objective unique formule le but à atteindre. Elle associe à chaque solution une valeur scalaire, qui donne la qualité de la solution permettant ainsi d'avoir un ordre total de toutes les solutions de l'espace de recherche. Pour un problème d'optimisation multi-objectif, l'évaluation consiste à associer, à

chaque solution, un vecteur de valeurs dont la taille correspond au nombre d'objectifs considérés et chaque élément du vecteur quantifie la qualité de la solution par rapport à la fonction objective correspondante. En conséquence, il n'existe plus d'ordre total entre les solutions, mais plutôt un ordre partiel, établi grâce à une relation de dominance. L'étape d'évaluation est un élément essentiel dans la conception d'une méta-heuristique. Elle permet d'orienter la méthode vers les bonnes solutions de l'espace de recherche. Notez qu'en pratique, l'évaluation est généralement l'étape la plus coûteuse, en termes de temps de calcul, de la méthode de résolution.

II.5.1.2 Les éléments clés d'un MOEA

Dans cette section, nous allons décrire en détail les éléments qui doivent être pris-en compte pour la conception d'un MOEA [6].

II.5.1.2.1 Attribution de Fitness

Dans un MOEA nous avons besoin d'un processus supplémentaire pour transformer un vecteur de fitness en une valeur scalaire. Principalement, il existe trois stratégies pour accomplir ce processus : à base de critère, à base d'agrégation et à base de préférence. À base de critère Cette approche choisit alternativement chacune des fonctions objectives lors de l'étape de sélection. Autrement dit, pour sélectionner un individu ou groupe d'individus seul un objectif est considéré. Par exemple, l'algorithme *Vector Evaluated Genetic Algorithm (VEGA)* [50] divise la population en sous-populations de même taille, et un objectif différent est utilisé pour attribuer le fitness de chaque sous-population.

- **À base d'agrégation**

Dans cette méthode, les fonctions objectifs sont agrégées ou combinées en une valeur scalaire unique. Pendant le processus d'optimisation, les paramètres sont modifiés systématiquement pour générer des différents éléments de l'ensemble Pareto optimal. Il faut noter que, même si une approche à base d'agrégation peut être formulée comme une relation de préférence, les solutions ne sont pas comparées dans l'espace objectif. En d'autres termes, les vecteurs sont mappés à partir de $\mathbb{R}^m$ vers $\mathbb{R}$ avant la comparaison.

- **À base de préférence**

Dans ce schéma, une relation de préférence est utilisée pour induire un ordre partiel de la population dans l'espace objectif. Ensuite, un score scalaire (rang) est attribué à chaque solution basée sur la façon avec laquelle la solution se compare par rapport aux autres solutions. Par exemple, les méthodes basées sur le rang de dominance comptent le nombre d'individus par lequel un individu donné est dominé. Dans les méthodes basées sur le nombre de dominances, le fitness d'un individu correspond au nombre d'individus qu'il domine. La dominance de Pareto est la relation de préférence la plus adoptée en MOEAs [34].

II.5.1.2.2 Élitisme

L'utilisation de l'élitisme doit permettre de préserver les meilleures solutions Pareto trouvées durant la recherche. Pour ce faire, une population de « bonnes » solutions est maintenue en parallèle de celles générées lors de la recherche. Cette population de solutions est appelée «archive». L'archive peut ne pas être utilisée dans le processus de recherche, on parle alors d'archive « passive », ou au contraire être régulièrement sollicitée pour l'obtention de nouvelles solutions (archive « active »). Comme toute utilisation d'élitisme, il est important, lors de l'utilisation d'archive active, de prévenir la convergence prématurée de l'algorithme. La taille de l'archive peut être fixe ou non. La mise à jour des archives est principalement basée sur la notion de dominance et il est possible d'inclure un mécanisme de diversification, comme c'est le cas pour les solutions générées lors de la recherche.

II.5.1.2.3 Estimateurs de densité

Un des buts en MOEAs est d'obtenir un ensemble de solutions non-dominées qui sont bien distribuées le long du front de Pareto. Dans ce qui suit, nous décrivons quelques techniques pour maintenir la diversité dans la population [34].

- **Partage de fitness**

L'objectif de partage du fitness est de former et de maintenir dessous-populations (niches) réparties sur l'espace objectif. L'idée est de considérer le fitness comme une ressource qui doit être partagée entre les individus dans la même niche.

Ainsi, plus il y a d'individus dans la niche, moins est le fitness attribué à chaque individu.

- **Hypergrilles**

Une hyper grille divise l'espace objectif dans des régions appelées hypercubes. Chaque solution non-dominée occupe un hyper cube comme il est montré dans la figure (II.5). L'idée est seulement d'accepter des solutions non-dominées appartenant à un hypercube peu peuplé. Bien que le nombre de divisions dans l'hypergrille dans chaque dimension soient constant, la position et l'extension de la grille peuvent être adaptées au cours du processus de recherche [34].

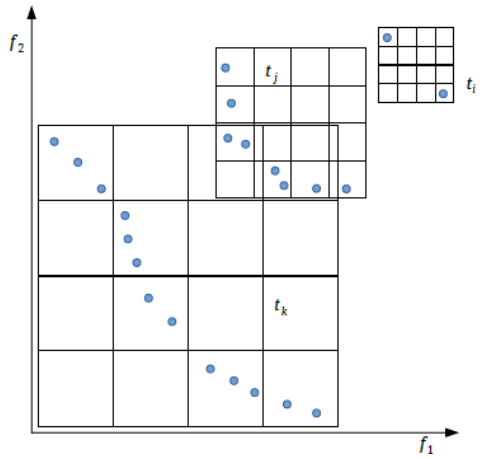


Figure II.5. L'hypergrille pour maintenir la diversité dans l'archive [34].

- ***Clustering***

L'objectif d'un algorithme de *clustering* est de partitionner un ensemble de points de telle sorte que : (1) chaque groupe contient des points très proches les uns des autres ; et (2) les points d'un groupe sont très différents des points des autres groupes. Dans un MOEA, nous utilisons le *clustering* pour préserver la diversité dans l'archive et réduire sa taille. Ce processus est composé de trois étapes figure (II.6) :

1. Partitionner l'archive en utilisant un algorithme de *clustering*.
2. Sélectionnez un individu représentatif de chaque groupe.
3. Enlever tous les autres individus dans le cluster [34].

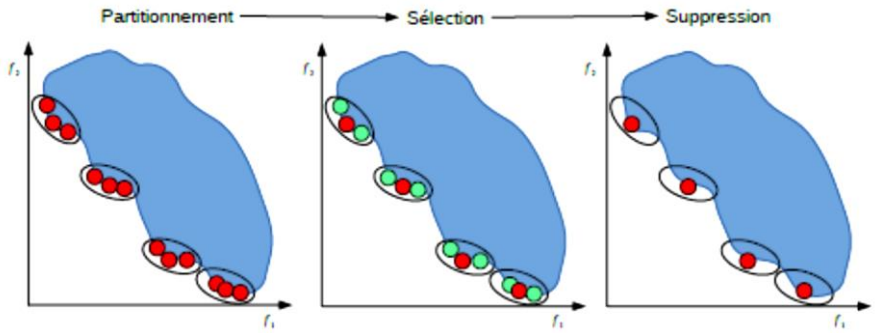


Figure II.6. Les étapes des méthodes de *clustering* dans le processus d'optimisation [34].

II.5.1.3 Méta-heuristiques à base de solution unique

Les S-méta-heuristiques commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, la méthode du recuit simulé, la recherche *tabou*, la méthode GRASP, la recherche à voisinage variable, la recherche locale itérée, et leurs variantes.

- **La recherche par descente**

La méthode de descente (DM : *Descentmethod*) est l'une des méthodes les plus simples de la littérature. Son principe consiste, à partir d'une solution initiale, à choisir à chaque itération un point dans le voisinage de la solution courante qui améliore strictement la fonction objectif. Il existe plusieurs moyens de choisir ce voisin : soit par le choix aléatoire d'un voisin parmi ceux qui améliorent la solution courante (first *improvement*) ; soit en choisissant le meilleur voisin qui améliore la solution courante (best *improvement*), ou encore en choisir une aléatoirement, parmi celles qui améliorent la solution courante (*random selection*). Le critère d'arrêt est atteint lorsque plus aucune solution voisine n'améliore la solution courante [51].

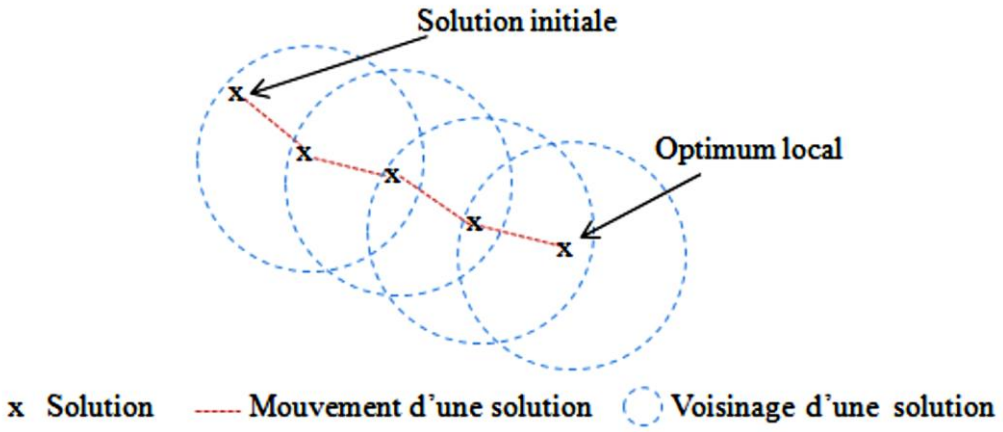


Figure II.7. Processus d'une recherche par descente.

Le principal inconvénient de la DM est qu'elle reste piégée dans le premier optimum local rencontré. Pour éviter de rester bloqué sur un optimum local des nouveaux algorithmes sont proposés (recuit simulé [52], recherche tabou [53]), ou à l'intégration de la recherche locale dans un processus itératif (ILS) [54] [55].

II.5.1.4 Méta-heuristiques à base de population de solutions

Les P-méta-heuristiques travaillent sur un ensemble de points de l'espace de recherche, en commençant avec une population de solutions initiales puis elles s'efforcent de l'améliorer au fur et à mesure des itérations. On distingue dans cette catégorie, les algorithmes évolutionnaires, qui sont une famille d'algorithmes issus de la théorie de l'évolution par la sélection naturelle, énoncée par Darwin et les algorithmes d'intelligence en essaim qui, de la même manière que les algorithmes évolutionnaires, proviennent d'analogies avec des phénomènes biologiques naturels [56].

Dans ce manuscrit, nous considérons principalement les algorithmes évolutionnaires, plus précisément les algorithmes pour résoudre le problème d'ordonnancement de *workflows* dans le *cloud*.

II.5.2 Les algorithmes évolutionnaires multi-objectifs

Les algorithmes évolutionnistes ou les algorithmes évolutionnaires (EAs) sont des méthodes de recherche stochastiques et d'optimisation qui simulent le processus de l'évolution naturelle. Comme d'autres stratégies de recherche stochastique (le recuit simulé, l'optimisation de colonie de fourmis, ou optimisation par essaim de particules), les EAs ne garantissent pas de trouver l'ensemble Pareto optimal mais essayent de trouver un ensemble non-dominé dont les vecteurs sont aussi proches que possible du front Pareto optimal. D'autre part, les EAs sont particulièrement bien adaptés pour résoudre les MOPs parce qu'ils travaillent en parallèle avec un ensemble de solutions possibles. Avec cette caractéristique, il est possible de trouver et avec une seule exécution plusieurs solutions de l'ensemble optimal de Pareto (ou une bonne approximation). En outre, les EAs sont moins sensibles à la forme ou à la continuité du front de Pareto [34].

Un algorithme évolutionnaire est typiquement composé de trois éléments fondamentaux :

- une population constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné.
- un mécanisme d'évaluation des individus permettant de mesurer l'adaptation de l'individu à son environnement,
- un mécanisme d'évolution de la population permettant, grâce à des opérateurs prédéfinis, d'éliminer certains individus et d'en créer de nouveaux.

Les algorithmes évolutionnaires pour l'optimisation mono-objective et les MOEAs partagent une structure similaire. La différence majeure est le mécanisme d'attribution de fitness puisque MOEA travaille avec des vecteurs de fitness de dimension m (m > 2) Comme il est

souligné par des différents travaux, trouver une approximation de front Pareto est en soi un problème bi-objectif dont les objectifs sont :

- Réduire la distance entre les vecteurs générés et le front Pareto optimal.
- Maximiser la diversité de l'approximation du front Pareto trouvé.

Par conséquent, l'attribution de fitness doit tenir compte de ces deux objectifs. L'algorithme1 décrit la structure de base d'un MOEA.

Algorithme 1 Pseudocode d'un MOEA

1: t = 0
2: Générer une population initiale P(t)
3:**tant que** le critère n'est pas satisfait faire
4: Évaluer le vecteur objectif F pour chaque individu de P(t)
5: Attribuer une Fitness pour chaque individu de P(t)
6: Choisir parmi P(t) un groupe de parents P'(t) préférant les meilleurs
7: Recombiner les individus de P'(t) pour obtenir les descendants P"(t)
8: Muter des individus en P(t)
9: Combiner P'(t) et P"(t) et sélectionner les meilleurs individus pour avoir P(t + 1)
10: t = t + 1
11:**fin tant que**

La population initiale est générée d'une manière aléatoire. L'attribution de fitness exige le classement des individus selon une relation de préférence, puis, en attribuant une valeur de fitness scalaire à chaque individu en utilisant ce rang. La sélection pour la reproduction (ligne 6) est réalisée comme dans le cas mono-objectif. En revanche, la sélection dans la ligne 9, destinée à mettre à jour les meilleures solutions (c.-à-d. élitisme), utilise une relation de préférence pour enlever des solutions et maintenir la taille de la population constante. Les algorithmes évolutionnaires les plus populaires sont les algorithmes génétiques.

L'algorithme génétique (*GA :Genetic Algorithms*) ont été développés par Holland en 1975. Ils sont basés sur les mécanismes de la sélection naturelle et de la génétique. Ils ont été efficacement utilisés pour résoudre plusieurs problèmes d'optimisation multi-objectifs.

L'algorithme commence avec un ensemble de solutions possibles du problème (individus), constituant une population. Les individus sont formés par des variables, qui sont les paramètres à ajuster dans un dispositif. Cette population est conçue aléatoirement à l'intérieur de limites prédéfinies. Certaines solutions de la première population sont utilisées pour former une nouvelle

population, à partir d'opérateurs génétiques (croisement, mutation, etc.). Ceci est motivé par l'espoir que la nouvelle population soit meilleure que la précédente. Les solutions qui serviront à former de nouvelles solutions sont sélectionnées aléatoirement d'après leurs mérites (représentés par une fonction objective spécifique au problème posé, qui devra être minimisée ou maximisée) : meilleur est l'individu, plus grandes seront ses chances de se reproduire c'est-à-dire, plus grande sera sa probabilité d'être sélectionné pour subir les opérateurs génétiques. Ceci est répété jusqu'à ce qu'un critère de convergence soit satisfait (par exemple, le nombre de générations ou le mérite de la meilleure solution) [57]. Le fonctionnement général des algorithmes génétiques est illustré par l'organigramme de la figure (II.8).

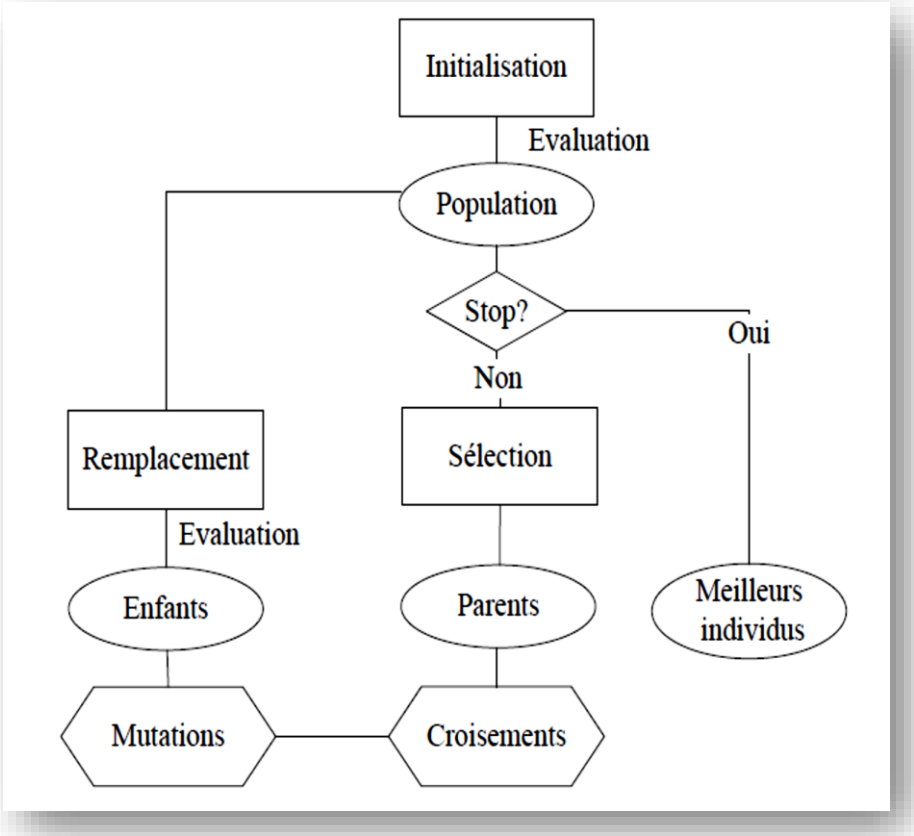


Figure II.8. Principe général des algorithmes génétiques [6].

II.5.3 Etat de l'art sur les Meta-heuristiques pour l'ordonnancement des *workflow* dans le *cloud*

L'ordonnancement des *workflows* est un problème NP-complet qui joue un rôle important dans le *cloud computing*. Ces algorithmes deviennent de plus en plus compliqués, car ils doivent gérer un grand nombre de paramètres contradictoires et qui concernent différents acteurs, dont les intérêts ne sont pas les mêmes. Le problème d'ordonnancement de *workflow* dans ces circonstances est caractérisé par le fait qu'il comporte un groupe d'objectifs contradictoires (*QoS*) et complexes. Par exemple (minimisation du coût, de temps réponse du service et la consommation d'énergie et autres). La motivation principale de ces algorithmes d'ordonnancement est de minimiser le temps et le coût d'exécution.

Un ordonnanceur trouve une allocation de ressource virtuelle parmi les différents fournisseurs (un plan de déploiement) qui optimise les paramètres de qualité de service (*QoS*). Ces paramètres de qualité de service peuvent être définis et proposés entre les fournisseurs de services *cloud* et leurs clients par un contrat de niveau de service SLA.

Plusieurs travaux ont été proposés pour résoudre le problème d'ordonnancements dans le *cloud computing*. Dans notre travail [23][88],on n'a concentré principalement sur les algorithmes basé sur les méta-heuristiques et récemment proposé dans l'environnement de *cloud*.

Dans [58], les auteurs ont proposé un algorithme génétique multi-objectif (MO-GA) pour l'ordonnancement des *jobs* dans le *cloud*. L'algorithme minimise la consommation d'énergie et maximise le profil des services fournis par le fournisseur sous la contrainte de délai. La première étape de MO-GA est la génération de la population initiale avec deux méthodes (aléatoire et glouton).

Après le calcul de fitness, les solutions ayant le meilleur rang (meilleure fitness) sont stockées dans les archives de Pareto qui contiennent des solutions non dominantes générées par des générations. L'opérateur de sélection est le tournoi, basé sur la stratégie d'élitisme et du *crowding*. Les opérateurs de reproduction sont des croisements et des mutations. Enfin, sélectionner la meilleure solution dans l'archive de Pareto est effectuée selon la préférence actuelle des clients entre les deux objectifs.

Dans [59], l'auteur a proposé un *Package* orienté vers le marché de l'ordonnancement hiérarchique dans un *cloud* distribué. Cette approche traite hiérarchiquement l'ordonnancement au niveau du service et l'ordonnancement au niveau de la tâche, chaque tâche de *wokflow* étant mappée sur le service *cloud*, L'objectif de cette approche est de minimiser le temps d'exécution, le coût et le temps de traitement. Pour l'ordonnancement au niveau de service un algorithme aléatoire

a été établi ; Alors au niveau de tâche trois méta- heuristiques sont utilisées qui sont les algorithmes génétiques et l'optimisation par la colonie de fourmis et l'optimisation par l'essaim de particule. Un autre algorithme été proposé par Kessaci.Y et al, dans [60], qui a pour but d'optimiser l'allocation des demandes des MVs dans une infrastructure de courtage dans trois niveaux de *cloud* : le client, le fournisseur de *cloud* et le courtier du *cloud*. Le rôle de ce dernier est de trouver une meilleure configuration des ressources proposées par le fournisseur pour répondre aux exigences des machines virtuelles. L'objectif de cette optimisation est de minimiser le temps de réponse et le coût des MVs en satisfaisant le client et pour maximiser le profit du courtier. Ce modèle définit deux types de services :(1) Le service d'infrastructure connecté au fournisseur et (2) un service commercial connecté au courtier. Le client expose ses demandes avec leurs contraintes *QoS* en deux types. Le premier type est indiqué parle client et le second est dérivé et utilisé par l'algorithme. Un algorithme génétique multi-objectif (MOGA) a été proposé pour fournir un ensemble d'allocations optimales de Pareto sur les meilleures machines virtuelles avec un coût et un temps de réponse minimaux.

Yassa.S et al. utilisent dans [6], l'algorithme *NSGA*-II basé sur l'approche Pareto. Cette méta-heuristique résoudre le problème d'ordonnancement multi-objectif de *workflow* dans l'infrastructure IaaS. Plusieurs paramètres de *QoS* ont été pris en considération (*makespan*, le coût, fiabilité et la disponibilité) et leur contraint tel que (temps, budget et autres) et des contrainte fortes (une tâche T*a* doit être exécutée sur des services comme MVa et MVb).

Deux autres travaux proposés par Yassa.S et al, dans le premier travail [46], où l'auteur développe un algorithme génétique basé sur l'agrégation de plusieurs objectifs (*makespan*, le coût, la disponibilité et fiabilité). Le but de cet algorithme et de résoudre l'ordonnancement des *workflows* dans le *cloud computing*. Dans le deuxième travail [47], les auteurs ont développé un algorithme génétique amélioré basé sur l'infection viral, où plusieurs métriques de qualité de service sont prises en considération (*makespan,* le coût, la fiabilité et la disponibilité) et des contraintes sur ces métriques (la durée et le budget) et aussi des fortes contraintes qui obligent l'exécution de certaines tâches sur certaines machines virtuelles. Les auteurs, dans [61], ont développé un système de contrôle à deux niveaux pour optimiser la gestion des ressources du data center. Les contrôleurs locaux au niveau de l'application déterminent la quantité des ressources nécessaires à la demande pour assurer sa performance. Un contrôleur global au niveau du data center détermine le placement des MVs et l'allocation des ressources. Il alloue toutes les machines virtuelles en même temps, tout en essayant de trouver l'allocation optimale en fonction des objectifs et des contraintes. Une politique de placement de MV a été proposée en utilisant un algorithme génétique amélioré avec une évaluation multi-objective floue pour minimiser

simultanément le gaspillage total des ressources, la consommation d'énergie et le coût de la dissipation de chaleur pour combiner différents objectifs. Dans cet algorithme, la meilleure solution est celle qui appartient plus à chaque ensemble flou avec un but.

Tableau II.1. Comparaison des travaux Méta-heuristiques pour l'ordonnancement des *workflows* dans le cloud.

Niveau	Titre	Algorithme\technique	Paramètres d'ordonnancement	Résultat\avantage	Limites
Ordonnancement au niveau de service	Job scheduling model for cloud computing based on multi-Objective genetic algorithm [58]	Algorithme génétique multi-objectifs (MO-GA)	– La consommation d'énergie – Le profit	– Optimise la consommation d'énergie et le profit de fournisseur	– Ne tient pas en compte les contraintes de précédence.
	A market-oriented hierarchical scheduling strategy in cloud workflow system [59]	Algorithme aléatoire	– Le temps de CPU	– Un temps d'exécution minimal. – Une stratégie hiérarchique efficace de commande.	– Il n'y a pas une relation entre le prix facturé et les performances du fournisseur.
Ordonnancement au niveau de tâche		Trois méta-heuristiques (AG,ACO,PSO)	– Makespan – Le cout		
	A Pareto-based genetic algorithm for optimized assignment of VM requests on a cloud brokering environment [60]	Multi-Objective Genetic Algorithm for Cloud Brokering (MOGA-CB)	– Le temps de réponse. – Le cout des instances MVs	– Une grande relation entre le prix facturé et la performance.	– Seulement deux critères de QoS pris en considération – Ne respecte pas la relation de précédence entre les tâches.
	Optimal multi-constraints allocation of workflows in resources of cloud computing environment [6]	NSGA-II (non dominated sorting genetic algorithm)	- Makespan - cout - fiabilité - disponibilité	– Prend en compte des contraintes sur les métriques QoS, les contraintes fortes et les contraintes de précédence entre les tâches.	– Un temps de calcule important. – Une difficulté de choisir la solution optimale.
	genetic algorithm approach to a cloud workflow scheduling problem with multi-QoS requirements [46]	Algorithme génétique (AG)	- Makespan - cout - fiabilité - disponibilité	– Plusieurs métriques de QoS à optimiser, tiennent compte des contraintes de précédence entre les tâches.	– L'agrégation des fonctions objectifs. – Ne tienne pas compte des contraintes sur les paramètres et les contraintes fortes.
	A genetic algorithm for multi-objective optimization in workflow scheduling with hard constraints [47]	Algorithme génétique basé sur l'infection virale (GA*)	– Makespan – Cout – Fiabilité – Disponibilité – Budge – Date limite.	– Prend en compte les contraintes de QoS, les contraintes fortes et les contraintes de précédence entre les tâches.	– Fournir une seul solution aux utilisateurs (Agrégation des objectives).
Ordonnancement au niveau de MV	Multi-objective virtual machine placement in virtualized data-center environnements [61]	Algorithme génétique flou multi-objectifs	– Gaspillage total des ressources. – Consommation d'énergie. – Coût de dissipation thermique	– Une grande performance. – Une combinaison des objectives contradictoires. .	– Manque de diversité des solutions. – Le placement des MVs n'est pas dynamique.

II.5.4 Synthèse

Les algorithmes existants pour l'ordonnancement des tâches ne sont pas adaptés dans le cas de l'utilisation d'une plateforme *cloud computing*. En effet, ces algorithmes ne prennent pas en compte les spécificités de ce dernier tel que « l'illusion de ressources infinies » et son modèle économique basé sur la consommation réelle des ressources.

Plusieurs heuristiques et méta-heuristiques sont étudiées pour traiter le problème d'ordonnancement ; Ce dernier est considéré comme un problème d'optimisation combinatoire, où il est impossible de trouver une solution optimale globale en utilisant des algorithmes ou des règles simples.

Pour améliorer la satisfaction du client et en même temps améliorer l'utilisation de différentes ressources des paramètres sont pris en compte, tels que : temps de réponse, coût, profit, la fiabilité, la disponibilité et la consommation d'énergie etc. Le tableau (II.2) présente les paramètres considérés dans les algorithmes présentés ci-dessus).

La plupart de ces algorithmes sont concentrées sur l'agrégation des objectifs (mono-objectif), et sur l'optimisation de deux paramètres de *QoS*, souvent le temps d'exécution (*makespan*) et le coût d'exécution. Ils ne prennent pas en compte d'autres paramètres ainsi que les contraintes de précédences entre les tâches.

Dans notre travail, nous avons pris en compte cet aspect (contraintes de précédences), en développant une approche d'ordonnancement basée Pareto à la façon dont chacune des métriques de *QoS (makespan*, coût*)* ne peut pas affecter l'autre.

Tableau II.2. Les métriques utilisées dans les différents algorithmes.

Référence	Algorithme/ Technique	makespan	coût	fiabilité	disponibilité	énergie	contrainte des QoS	contrainte de précédence	méta-heuristique	Pareto
[58]	MO-GA	non	non	non	non	oui	oui	non	oui	oui
[59]	– Un algorithme aléatoire – Trois méta-heuristiques	oui	oui	non	non	non	oui	oui	oui	non
[60]	MOGA-CB	oui	oui	non	non	non	non	non	oui	oui
[6]	NSGA-II	oui	oui	oui	oui	non	oui	oui	oui	oui
[46]	GA	oui	oui	oui	oui	non	non	oui	oui	non
[47]	GA*	oui	oui	oui	oui	non	oui	oui	oui	non
[61]	Algorithme génétique flou multi-objectifs	non	non	non	non	oui	non	non	oui	oui

II.6 Conclusion

Dans ce chapitre, nous avons présenté les principaux concepts, les définitions et notations liées au domaine d'optimisation multi-objectif, puis, nous avons donné un état de l'art des méthodes d'ordonnancement de *workflows* dans le *cloud* avec une comparaison entre les différents algorithmes. Nous consacrons le chapitre suivant pour présenter les réseaux de Petri et ses différentes propriétés, ensuite nous allons définir les réseaux de Petri redimensionnables, évolutionnaires, et l'ordonnancement basé sur ce formalisme et enfin les travaux liés dans ce contexte.

Chapitre III

L’ordonnancement par les réseaux de Petri

Plan

III.1 Introduction

Plusieurs algorithmes et approches ont été proposés pour résoudre les problèmes d'ordonnancement traditionnel par exemple les problèmes d'ordonnancement des ateliers à cheminement multiples (JSSP), les problèmes d'ordonnancement des projets à des ressources limitées (RCPSPs), et les problèmes d'ordonnancement des systèmes robotisés. La plupart de ces algorithmes se concentrent sur l'efficacité de résolution. Actuellement avec l'évolution de la technologie les méthodes classiques sont inadéquats parce qu'elles dégradent les performances du système (inflexibilité, ressources, … etc).

Le problème d'ordonnancement dans les systèmes distribués est un problème complexe (NP-complet. L'objectif principal d'un tel ordonnancement est d'ordonnancer les tâches et de les mapper sur les machines appropriées pour obtenir la performance globale optimale du système (minimiser le *makespan* et minimiser le coût). L'approche basée sur les réseaux des Petri traitent le problème de flexibilité des tâches et des ressources. La modélisation et la puissance d'analyse par les réseaux de Petri explorent facilement les différents aspects de problème d'ordonnancement (comportement et les propriétés structurelles) et intègrent des facteurs ignorés dans d'autres approches. Tous ces avantages nous ont motivé de proposer une approche basée sur les réseaux de Petri pour résoudre le problème d'ordonnancement des *workflows* dans le *cloud computing*.

III.2 Les réseaux de Petri

Les réseaux de Petri [62] constituent une famille de formalismes mathématiquement fondés, adaptés à la modélisation de systèmes parallèles et asynchrones. Ils permettent l'analyse de différents aspects comportementaux d'un système (discret, stochastique, temporisé), soit sur des questions relatives à l'expressivité, soit sous l'angle de la décidabilité pour résoudre des problèmes de vérification [63].

Les réseaux de Petri est un outil formel pour la modélisation l'analyse des systèmes qui comportent des comportements tels que la concurrence, conflit et la dépendance causal entre les évènements [64]

III.2.1 Représentation graphique

Informellement un réseau de Petri (RdP) est un graphe biparti alterné qui possède deux types de nœuds : les places (cercles) et les transitions (rectangles). Des arcs (flèches) relient les places aux transitions figure (III.1). L'état du système, nommé marquage, est défini par la répartition des jetons dans les places. Une transition est franchissable sous certaines conditions, notamment lorsqu'il y a suffisamment de jetons dans ses places d'entrée. Le franchissement d'une transition se traduit par une modification du marquage consistant en la consommation des jetons indispensables au franchissement de la transition et la création éventuelle de nouveaux jetons dans les places en sortie de la transition et la création éventuelle de nouveaux jetons dans les places en sortie de la transition [64]. Quelques interprétations typiques des transitions et leurs places d'entrées et de sorties sont présentées dans le tableau (III.1).

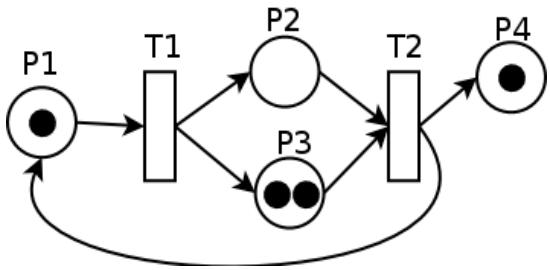


Figure III.1. Représentation graphique des éléments de RdP

Tableau III.1. Quelques interprétations typiques de transitions et de places [65]

Places d'entrées	Transition	Places de sortie
Pré-conditions	Evènement	Post-conditions
Données d'entrée	Traitement	Données de sortie
Signaux d'entrée	Processeur	Signaux de sorties
Ressources demandées	Tâche	Ressources libérées
Conditions	Clauses en logique	Conclusions
Buffers	Processeur	Buffers
Place	Transition	Arc	Jeton

III.2.2 Notation

Pour chaque nœud x de P∪T, nous notons par $\bullet x$ l'ensemble de ses nœuds entrantset par $x\bullet$ l'ensemble de ses nœuds sortants. Formellement : $\forall x \in P \cup T$:

$$\bullet x = \{y \in P \cup T | (y,x) \in F\}$$

$$x\bullet = \{y \in P \cup T | (x,y) \in F\}$$

- **Définition 1**

Réseaux de Petri (RDP un quintuplet N= (P, T, F, W, M_0) tel que :

- P = {p_1, p_2,…, p_m} un ensemble des places ;
- T = {t_1, t_2,…, t_n} un ensemble finis des transitions, tel que : P ∩ T = ∅ and P ∪ T ≠∅;
- F ⊆ ((P × T) ∪ (T × P)) un ensemble des arcs ;
- W : F → {1, 2, 3 …} est la fonction des poids;
- M_0 : P → {0, 1, 2, 3 …} est le marquage initial.

La structure de RdP est donnée par le quadruplet *Q= (P, T, F, W), Q* représente le RdP sans aucune spécification de marquage inilialM_0. Un RdP marqué est noté par *R= (Q, M0).*

III.2.3 Marquage

On appelle marquage une distribution de jetons sur les places. Le marquage initial noté *M0* est la distribution initiale de jetons dans le réseau à l'instant initial. Un marquage définit l'état du système.

Le marquage d'un RdP est une application $M : P \longrightarrow \mathbb{N}$ donnant pour chaque place le nombre de jetons qu'elle contient. Le marquage de la place *pi* est noté par *M (pi)* qui est un nombre entier [65].

III.2.4 Evolution d'un réseau de Petri

Un RdP est un graphe muni d'une sémantique opérationnelle, c'est -à- dire qu'un comportement est associé au graphe, ce qui permet de décrire la dynamique du système représenté. L'évolution d'un RdP correspond à l'évolution de son marquage au cours du temps, cette évolution simule le comportement dynamique du système modélisé. Le marquage du réseau change selon les règles suivantes :

1. Une transition t est validée (ou sensibilisée, franchissable ou tirable) si toutes les places p en amont de celle-ci (p ∈°t) possèdent au moins W (p,t) jetons.

$$t\ est\ valide \Leftrightarrow \forall\, p \in {}^{\circ}t : M(p) \geqslant W(p,t)$$

2. Un franchissement d'une transition validée t consiste à supprimer W (p, t) jetons de chaque place d'entrée, et ajouter W(t,p) jetons à chacune de places de sortie de t. Le franchissement d'une transition dans un marquage M donne un nouveau marquage Ḿ défini par :

$$\forall\, p \in P : \acute{M}(p) = M(p) - W(p,t) + W(t,\acute{p})$$

3. Lorsqu' une transition est validée, cela n'implique pas qu'elle sera immédiatement franchie ; cela ne représente qu'une possibilité de franchissement, dans un RdP, même si plusieurs transitions sont validées par un même marquage une et seulement une transition peut être franchie [65].

III.2.5 Propriétés comportementales des réseaux de Petri [65]

La construction de la structure du RdP qui modélise le système réel est une tâche très importante. La tâche suivante qui est de même importance est l'étude ou l'analyse des propriétés de ce réseau. Il existe deux classes de propriétés. Celles qui sont dépendantes d'un marquage initial M0 dites comportementales, si l'on change ce marquage, rien ne garantit que ces propriétés tiennent encore. D'autres qui sont indépendantes du marquage initial dites structurelles. Dans ce qui suit nous présentons les propriétés comportementales d'un RdP.

III.2.5.1 L'atteignabilité

Un marquage *M* est atteignable ou accessible depuis le marquage initial M_0s'il existe une séquence des transitions franchissables *S* menant de M_0à *M*.

$$M \in^{*} M_0 \Leftrightarrow \exists\, S \in T*: M_0 \xrightarrow{S} M$$

Où : *T** est l'ensemble de séquences de transitions

III.2.5.2 La bornitude

Un RdP marqué est *k- borné* si toutes ses places sont *k-bornées*, c'est-à-dire quel que soit le marquage accessible *M* à partir du marquage initial M_0, et quel que soit la place *p* considérée le nombre de jetons contenus dans cette places est inférieur ou égale à une borne *k* :

$$R(Q, M_0)\ est\ k-borné \Leftrightarrow \forall M \in * M_0\ , \forall p \in P : M(p) \preccurlyeq k$$

Un RdP marqué est *sauf* ou *binaire* pour un marquage initial M0 s'il est *1-borné*.

Cette propriété de bornitude est nécessaire lorsqu'une place représente par exemple une ressource dont le nombre est limité, ou un stock pour lequel la capacité est limitée

III.2.5.3 Vivacité et blocage

Une transition test *vivante* pour un marquage initial M_0si pour tout marquage *M* accessible à partir de ce marquage initial, il existe une séquence de franchissements à partir de *M* contenant *t*. Autrement dit, quelle que soit l'évolution, il existera toujours une possibilité de franchir *t*:

$$t\ est\ vivant \Leftrightarrow \forall\ M \in * M_0, \exists \acute{M}\ \in * M, \exists S\ \in T^*: M \xrightarrow{S} \acute{M}\ tel\ que\ t \in S$$

Un RdP marqué est vivant pour un marquage initial M_0 si toutes ses transitions sont vivantes pour ce marquage initial. Dans un tel réseau, on garantit que chaque transition soit étirable ventuellement peu importe l'état du système. Un RdP est dit conforme s'il est sauf et vivant.

$$R(Q, M) est\ vivant \Leftrightarrow \forall\ M \in *\ M_0\ , \forall t \in T, \exists\ \acute{M} \in * M, \exists S\ \in T *: M \xrightarrow{S} \acute{M} \wedge t \in S$$

Un RdP marqué est dit sans blocage pour un marquage initial M0si aucun marquage accessible n'est un blocage autrement dit s'il peut continuellement évoluer. Un marquage *M* est un *blocage* si aucune transition n'est validée

III.2.5.4 États d'accueil

Un RdP possède un état d'accueil *Ma* pour un marquage initial *Mo* si pour tout marquage accessible *Mi* il existe une séquence de franchissement permettant d'atteindre le marquage *Ma* :

$$Ma\ est\ un\ état\ d'accueil \Leftrightarrow \forall M_i\ \in * M_0, \exists S\ \in T *: M_i \xrightarrow{S} M_a$$

Un RdP est réinitialisable (ou réversible) pour un marquage initial M_o si M_o est un état d'accueil.

III.2.6 Réseau Petri Redimensionnable (RPR)

- **Définition 2**

Un réseau Petri redimensionnable (RPR) [66] (extension des RdPs) est défini comme :

$\xi = (P, P^h, T, T^h, F, W, M_0, O_{pre}, O_{post})$:

- $P = \{p_1, p_2, \ldots, p_m\}$ est un ensemble fini des places ;
- P^h est un ensemble des places cachées, pour le quel $P \cap P^h = \varnothing$;
- $T = \{t_1, t_2, \ldots, t_n\}$ est un ensemble finis des transitions, pour le quel : $P \cap T = \varnothing$ et $P \cup T \neq \varnothing$;
- T^h est un ensemble des transitions caché, tel que $T \cap T^h = \varnothing$ and $(P \cup P^h) \cap (T \cup T^h) = \varnothing$;
- $F \subseteq ((P \cup P^h) \times (T \cup T^h)) \cup ((T \cup T^h) \times (P \cup P^h))$ est un ensemble des arcs;
- $W : F \rightarrow \mathbb{N}$ est une fonction des poids qui associer des valeurs réelles non nulles de chaque arc ;
- $M_0 : P \rightarrow \mathbb{N}$ est le marquage initial des places non cachées (toutes les places cachées sont initialisées par zéro jeton)
- $O_{pre} \in \mathbb{N}$ est le pré-order maximal permis dans RPR (équation(III. 1)), pour chaque $t \in (T \cup T^h)$:

$$\sum_{\substack{p \in {}^\bullet t \\ p \in (p \cup ph)}} W(p,t) \leq O_{pre} \qquad \text{(III. 1)}$$

- $O_{post} \in \mathbb{N}$ est le post-order maximal permis dans RPR (équation(III. 2)), pour chaque $t \in (T \cup T^h)$:

$$\sum_{\substack{p \in t^\bullet \\ p \in (P \cup Ph)}} W(t,p) \leq O_{post} \qquad \text{(III. 2)}$$

Nobile et al, dans [66], ont proposé un réseau de Petri évolutionnaire (RPE) comme nouveau extension pour (RPR). Le but de cette proposition est de développer une méthodologie évolutive dont les solutions candidates sont basées sur les RdPs. Les réseaux de Petri évolutionnaires (RPE) sont basés sur réseaux de Petri redimensionnable (RPR), avec deux opérateurs génétiques : croisement et mutation qui donnent tout à fait le fondement du développement d'un algorithme évolutif robuste pour l'optimisation. Les RPEs fournissent un *framework* conceptuel pour la

représentation des solutions candidates. Ci-dessous, nous décrivons d'abord la définition formelle des RPEs.

III.2.7 Réseaux de Petri Evolutionnaires (RPE)

Nobile et al [66] ont proposé les réseaux de Petri évolutionnaires comme une nouvelle extension des RPRs. le but de cette proposition est de développer une méthodologie pour lequel les solutions candidat sont basés sur les RdPs. Les réseaux de Petri évolutionnaires sont basés sur réseaux de Petri redimensionnables(RPR), avec deux opérateurs génétiques : croisement et la mutation. RPEs fourni un *framework* conceptuel pour la représentation des solutions candidats.

- **Définition 3**

Un Réseaux de Petri Evolutionnaire (RPE) est un triple $E = (\xi, X, \mu)$, tel que :

- $\xi \in \Xi$ est un RPR; tel que Ξ est un espace de tous les topologies possibles des RPRs;
- $X: (\Xi \times \Xi) \rightarrow (\Xi \times \Xi)$ est l'opérateur de croisement, les inputs de cet opérateur sont deux RPRs ξ et $\bar{\xi}$ (telque $P_{\xi} = P_{\bar{\xi}}$) et qui donne deux nouveaux RPRs;
- $\mu : \Xi \cup \{P_{in}, t, P_{out}\} \rightarrow \Xi$ est l'opérateur de mutation, tel que $\{P_{in}, t, P_{out}\}$ est un triple de deux places P_{in} et P_{out} et une transition t, à savoir $P_{in}, P_{out} \in (P \cup P^h) \cup P^{\infty}$ et $t \in (T \cup T^h) \cup T^{\infty}$, tel que P^{∞} et T^{∞} sont des ensembles infinis des places et transitions, tel que :

$P^{\infty} \cap (P \cup P^h) = \emptyset$ and $T^{\infty} \cap (T \cup T^h) = \emptyset$.

III.2.8 L'ordonnancement basé sur les réseaux de Petri

Les réseaux de Petri composent un formalisme de modélisation adapté à la description des systèmes composé des activités hautement parallèles et coopérants. Ils sont de plus en plus utilisés pour la modélisation et l'analyse dans le domaine des systèmes de fabrication.

Parmi les avantages importants des réseaux de Petri est la spécification des propriétés des systèmes de production par exemple : les conflits, les blocages et les ressources limitées. Toute sa propriété peut être facilement représentée dans un seul modèle formel.

La simplicité de la construction, et la possibilité du capturer les contraintes fonctionnelles, temporelles et ressources motivent les chercheur à résoudre les problèmes d'ordonnancements à base des réseaux de Petri [83].

III.3 Travaux connexes

Plusieurs travaux ont été proposés pour modéliser le problème d'ordonnancement traditionnel avec les réseaux de Petri (RdP). Parmi ces travaux,le travail de Van der Aalst W. M. P. [67], utilise les réseaux de Petri temporelle (TPNs) afin de mapper les problèmes d'ordonnancement acyclique dans les TPNs pour but de minimiser le *makespan* et analyser le problème. Le travail de Kim [68] modélise les problèmes d'ordonnancements des projets à des ressources limités (RCPSPs), par TPNs avec l'utilisation de la stratégie de branchement dynamique et ressource basé sur les limites inferieures. Pas beaucoup des approches qui combinent les réseaux de Petri avec les algorithmes génétiques pour résoudre le problème d'ordonnancement. Sachant que le travail de [69] où les auteurs modélisent le problème d'ordonnancement des ateliers à cheminement multiples (JSSP) avec le RdP et optimise le *makespan* et retard total avec les algorithmes génétiques(AG).

Dans [70] les auteurs modélisent le problème des systèmes de fabrication flexibles avec les réseaux de Petri et les optimiser avec les AGs.

L'approche GAPN (algorithmes génétiques et réseaux de Petri), a été proposée dans [82] qui combinent la modélisation par les réseaux de Petri et la capacité d'optimisation avec des algorithmes génétiques (AG) pour ordonnancement des systèmes de fabrication, cette approche utilise les réseaux de Petri pour formuler le problème et les AG pour le processus d'ordonnancement. L'avantage de cette approche est la modélisation d'une grande variété des systèmes de fabrication sans faire une modification ni dans la structure du réseau ni dans la représentation chromosomique.

Le travail dans [83] se concentre sur les modèles de réseau de Petri coloré (CPN) avec des algorithmes de recherche locale comme une stratégie de résolution de différents problèmes d'ordonnancement.

Pour un ordonnancement des productions complexes, les auteurs dans [84] utilisent des réseaux de Petri (PN) et les algorithmes génétiques (GA).

Hadiby G.R et al. [86] développent un algorithme d’ordonnancement méta-heuristique (recuit simulé) pour le modèle de fabrication flexible hybride (HFMS) basé sur un réseau de Petri hiérarchiques. L'objectif de cet algorithme est de résoudre le problème d'affectation des différentes tâches à chaque machine. Le critère d'optimisation est la minimisation de temps de réalisation du processus (*makespan*) sous la contrainte de maximisation de productivité.

Toutes les approches qui coupler les RdPs avec les AGs fournissent une seule solution au lieu d’un ensemble calculé. De plus ses approches ont été appliquées pour l’ordonnancement des tâches pour les systèmes de fabrications. Au meilleur de nos connaissances, aucune des approches susmentionnées n'a été proposée ou appliquée dans le *cloud computing*.

III.4 Conclusion

Nous avons présenté dans ce chapitre les notions de base des réseaux de Petri. Ensuite nous avons présenté un état de l’art des travaux qui traite le problème d’ordonnancement traditionnel avec les réseaux de Petri et les algorithmes génétiques.

Le chapitre qui suit sera consacré essentiellement pour l’algorithme proposé.

Chapitre IV

L'algorithme LEPN/GA

Plan

IV.1 Introduction

La plate-forme *cloud computing* est caractérisée par l'utilisation d'un grand pool de ressources informatiques accessibles à la demande à travers Internet. Il offre une variété de ressources, tels que le réseau, le stockage, aux utilisateurs adoptées par IaaS, PaaS, SaaS ... etc [71]. La plupart de ces services sont sous forme de *workflow* (tâches dépendantes).

De nombreux algorithmes sont utilisés pour fournir aux utilisateurs un ordonnancement optimal de *workflow* affectations tâche-ressource. Cependant, l'ordonnancement dans un environnement de *cloud computing* est très complexe.

L'ordonnancement du *workflow* est défini comme un problème d'optimisation multi-objectif dont la complexité augmente avec le nombre de paramètres pris en compte. Pour résoudre ce problème sur une plate-forme *cloud*, nous proposons une approche ***Labelled Evolutionary Petri Net/Genetic Algorithm (LEPN/GA)***[89]pour la résolution du problème d'ordonnancement multi-objectif de *workflows* dans l'infrastructure IaaS. Cette approche est une combinaison entre deux paradigmes, une technique évolutionnaire (les algorithmes génétiques) et les réseaux de Petri qui vise à fournir une affectation optimale (tâche-MV) pour héberger les applications des utilisateurs en minimisant le *makespan* et le coût d'exécution du *workflow*.

IV.2 Modélisation du problème d'ordonnancement de *workflows* dans le *cloud*

Dans notre modèle d'ordonnancement, le but est de trouver une meilleure affectation des tâches à l'ensemble des instances des machines virtuelles (MVs) basés sur le prix et le temps d'exécution de la demande.

Le problème consiste à ordonnancer les J tâches de client représenté sous forme de *workflow* sur N machines virtuelles de différents types. En outre nous savons que le problème d'ordonnancement des tâches est un problème NP-difficile, donc il nécessite l'utilisation d'une méta-heuristique telle que les algorithmes génétiques.

Dans notre modèle, le client soumet une demande pour exécuter ses tâches, cette demande est définit par deux paramètres de qualité de service (QoS). Le premier paramètre représente le

temps d'exécution du *workflow* (*makespan*) et le deuxième représente le coût d'exécution sur les différents types d'instances.

Les fonctions objectives de notre approche a pour but de minimiser le *makespan* (équation(IV. 5)) et le coût totale d'exécution (équation(IV. **6**)). La figure(IV.1) montre la représentation de notre modèle proposé.

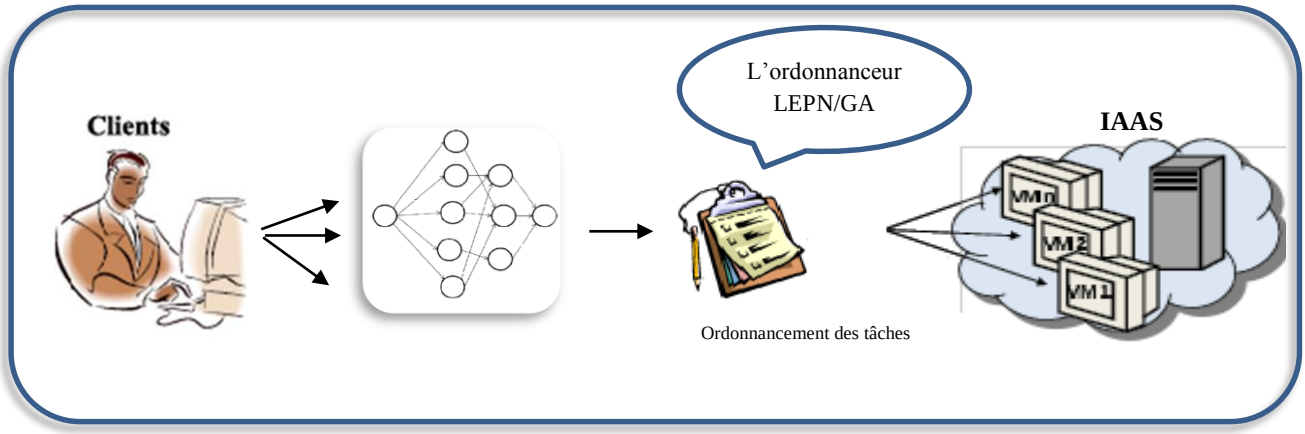


Figure IV.1. Représentation de l'approche proposée

IV-2.1 Modélisation d'ordonnancement faisable avec les Réseaux de Petri redimensionnables (RPRs)

Pour modéliser le problème d'ordonnancement dans le *cloud computing*, nous introduisons deux types de places, la première représente les machines virtuelles et l'autre représente la dépendance entre une tâche t_i et une autre tâche t_j. Chaque ordonnancement faisable représente un chromosome dans notre approche. Dans le problème d'ordonnancement du *workflow*, le chromosome composé d'une séquence de gènes, chaque gène est une affectation de tâche dans le *workflow* à une machine virtuelle (MV) correspondante. Chaque instance des machines virtuelles est caractérisée par sa vitesse de calcul **($MIPS_j$)** correspondant au nombre d'instructions que la ressource peut traiter par seconde et un coût monétaire d'utilisation par unité de temps, prix $\mathbf{U_j}$. Contrairement à une RdP traditionnelle, le RPR se compose d'un nombre variable de places,($\| P \| = m$) et de transitions($\| T \| = n$), et aussi un nombre variable des places cachées (resp.transition) dans l'ensemble P^h(resp,T^h) dont les cardinalités peuvent changer pendant le processus d'évolution à cause des opérateurs génétiques. Les notions et les notations de base se trouvent dans [66].

Nous modélisons un faisable ordonnancement (solution candidate) avec les RPRs avec 8-tuple= $\left(P, P^h, \mathrm{T}, \mathrm{F}, \mathrm{W}, M_0, O_{pre}, \mathrm{O}_{post}\right)$où :

$P = \{C_{ij}\}/i, j \in [1, n]$représente l'ensemble des conditions d'exécutions (conditions de précédences, transfère des données) de chaque tâche, C_{ij}désigne la contrainte de précédence et le transfert de donné entre la tâche t_i et t_j, tel que : $P \bigcap P^h = \emptyset$;

- $P^h = \{p_1, p_2, \dots, p_m\}$ représente l'ensemble des machines virtuelles.
- $T = \{t_1, t_2, \dots, t_m\}$ représente l'ensemble des tâches qui compose le *workflow*.
- $F \subseteq \left((P \cup P^h \times T)\right) \cup \left(T \times (P \cup P^h)\right)$ l'ensemble des arcs.
- $W : F \rightarrow \mathbb{N}$ fonction poids qui associé une valeur entière non négative à chaque arc. Dans ce cas, $W(f) = 1, \forall f \in F$;
- $M_0 : P \rightarrow \mathbb{N}$ marquage initial des places cachées dans le réseau (toutes les places non-cachées sont initialisées par zéro jeton initialement).
- $O_{\substack{pre \\ t \in T}} \in \mathbb{N}$ est le maximal pre-order autorisé dans RPR (équation (IV. 1)) pour chaque tâche $t \in T$:

$$\sum_{\substack{p \in {}^\bullet t \\ p \in (p \cup p^h)}} W(p, t) \preccurlyeq O_{pre} \qquad \text{(IV. 1)}$$

- $O_{\substack{post \\ t \in T}} \in \mathbb{N}$ est le maximal post-order autorisé dans RPR (équation (IV. 2)) pour chaque tâche $t \in T$:

$$\sum_{\substack{p \in t^\bullet \\ p \in (p \cup p^h)}} W(p, t) \preccurlyeq O_{post} \qquad \text{(IV. 2)}$$

IV.2.2 La modélisation du *workflow* avec un réseau de Petri redimensionnable (RPR)

La figure (IV.2) montre un exemple de *workflow* (DAG) composé de sept nœuds (tâche). Les arcs entre les tâches indiqué le données entrée et les données sortie entre les nœuds de tâche. Chaque tâche assigne à une machine virtuelle, un exemple de solution est représenté dans le tableau (IV.1). Nous supposons que la tâche fille ne peut pas être exécutée jusqu'à ce que toutes ses tâches mères soient achevées. Chaque tâche est exécutée une seule fois sur une seule machine. La figure (IV.3) présente la modélisation de *workflow* avec un réseau de Petri redimensionnable.

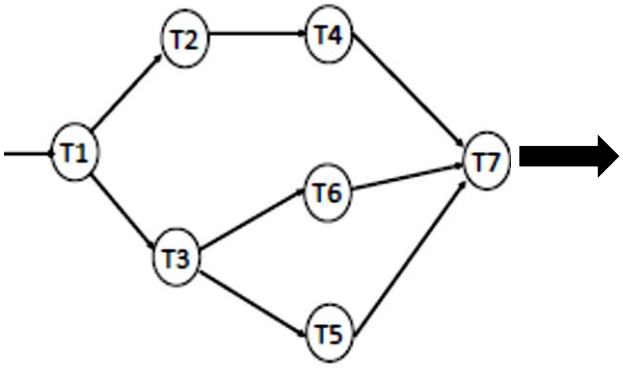


Figure IV.2. Example d'un Workflow (DAG)

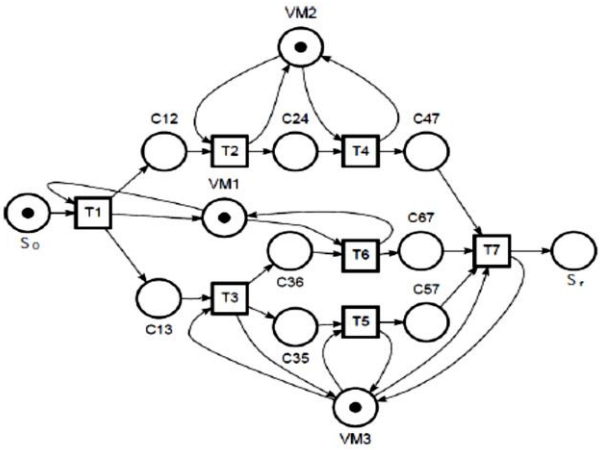


Figure IV.3. Modélisation de l'exemple de workflow avec un RPR

Tableau IV.1. Exemple d'une solution (tâche-MV)

Tache	**T_1**	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**T_7**
MV	MV_1	MV_2	MV_3	MV_2	MV_3	MV_1	**MV3**

IV.2.3 Le processus évolutionnaire pour l'ordonnancement des *workflows*

Dans cette section nous utilisons les réseaux de Petri évolutionnaire (extension des RPRs) pour modéliser le problème d'ordonnancement évolutionnaire. Ce modèle associe à chaque transition une étiquette $L = (FT, C)$ où FT signifie la date de fin d'exécution de chaque tâche, et C signifie le coût d'exécution de chaque tâche T_i sur une machine virtuelle MV_j.

Pour trouver un ordonnancement optimal, un processus évolutionnaire basé sur les algorithmes génétiques a été appliqué. Pour chaque transition, nous calculons le *makespan* et le coût de réseaux jusqu'à le franchissement de tous les transitions.

IV.2.3.1 Le codage de chromosome

Formellement, nous avons défini le problème d'ordonnancement dans le *cloud* comme un triple (EPN, FT, C) où :

1. EPN est un réseau de Petri évolutionnaire.
2. FT : T × P ⟶ ℝ;

(t, MV) → FT (t, MV);

Tel que :

FT (t, MV***) = FT (t) = ST (t) + ET (t,*** MV ***(t))*** **(**IV. 3**)**

Où:

- $ST\ (t)_{t_p \in pred(t)} = \max\left\{FT\ (t_p) + TT\left(\text{MV}\ (t_p), \text{MV}\ (t)\right)\right\}$
- $ET\ \left(t, \text{MV}\ (t)\right) = \frac{(instruction\ lenght\ of\ task)}{MIPS\ rate\ of\ virtual\ machine}$
- $TT\left(\text{MV}\ (t_i), \text{MV}\ \left(t_j\right)\right) = \frac{data\ [i,j]}{\left(B\left[\text{MV}\ (t_i), \text{MV}\ \left(t_j\right)\right]\right)}$
- Où :
- FT (t): la date de fin d'exécution de la tâche t;
- ST (t): la date du début d'exécution de la tâche t;
- ET (t, MV) : le temps d'exécution de la tâche t sur la machine virtuelle MV ;
- Pred (t): L'ensemble des prédécesseurs immédiats de la tâche t;
- TT (MV (t_i), MV (t_j)): le temps de transfert des données de la tâche t_i (exécuter sur MV (t_i)) vers la tâche t_j (exécuter sur MV (t_j));
- $B\left[\text{MV}\ (t_i), \text{MV}\ \left(t_j\right)\right]$: la bande passante entre MV (t_i) et MV (t_j);

3. C : T ×P → ℝ;

   $(t, \text{MV}\ ) \rightarrow C(t, \text{MV}\ )$;

Tel que :

$$\boldsymbol{C}\ (\boldsymbol{t}) = \mathbf{ET}(\boldsymbol{t}, \mathbf{MV}\ ) \times \boldsymbol{UEC}\ (\mathbf{MV}\ ) \qquad \text{(IV. 4)}$$

Où:

- C (t): le coût d'exécution de la tâche t;
- UEC (MV): le prix par unité de temps de la machine virtuelle.

IV.2.3.2 Opérateurs génétiques

Nous présentons dans cette section les opérateurs génétiques qui représentent le cœur d'un algorithme génétique.

IV.2.3.2.1 L'opérateur de sélection et de croisement

Le but de l'opérateur de croisement est de combiner les caractéristiques de plusieurs solutions pour générer des solutions encore meilleures. Mais, avant d'effectuer le croisement,

nous utilisons un processus de sélection afin de ne conserver que les meilleures solutions. Nous avons utilisé la sélection par tournoi [72], où nous choisissons deux individus (tournoi binaire) au hasard de la population et nous copions le meilleur individu dans la prochaine population.

Sur la base du modèle d'ordonnancement faisable présenté dans la section (IV.2.1), le mécanisme de croisement entre deux RPRs génère de nouveaux enfants qui héritent des meilleures sous-structures des parents. N'existe pas un travail concernant le croisement entre des graphes bipartites, ou plus particulièrement sur des RdPs. En effet, le croisement entre deux RdPs est fait pour identifier certaine sous-structure dans chaque graphe, "Détachez-les" d'un graphe parent et "attachez-les" dans l'autre, et vice versa tout en conservant la cohérence des deux graphes [66].

Notre proposition pour le mécanisme de croisement des deux RPRs : $\xi(\tau)$, $\bar{\xi}(\tau) \in \Xi$ en considérant trois points:

- Une seule transition (représente une tâche dans le problème d'ordonnancement) $t_\chi \in T$ est au hasard sélectionnée à partie de ξ et la même transition est sélectionnée de $\bar{\xi}$.
- Sélectionner la sous-structure pre-set et post de t_χ (resp. $\bar{t}_\chi$).
- Changer les sous-structures entre ξ et $\bar{\xi}$.

L'utilisation de cette méthode sauvegarde les contraintes de précédences dans chaque RPR. La figure (IV.4) décrit un simple croisement, en supposant dans cet exemple $t_\chi = T_2$.

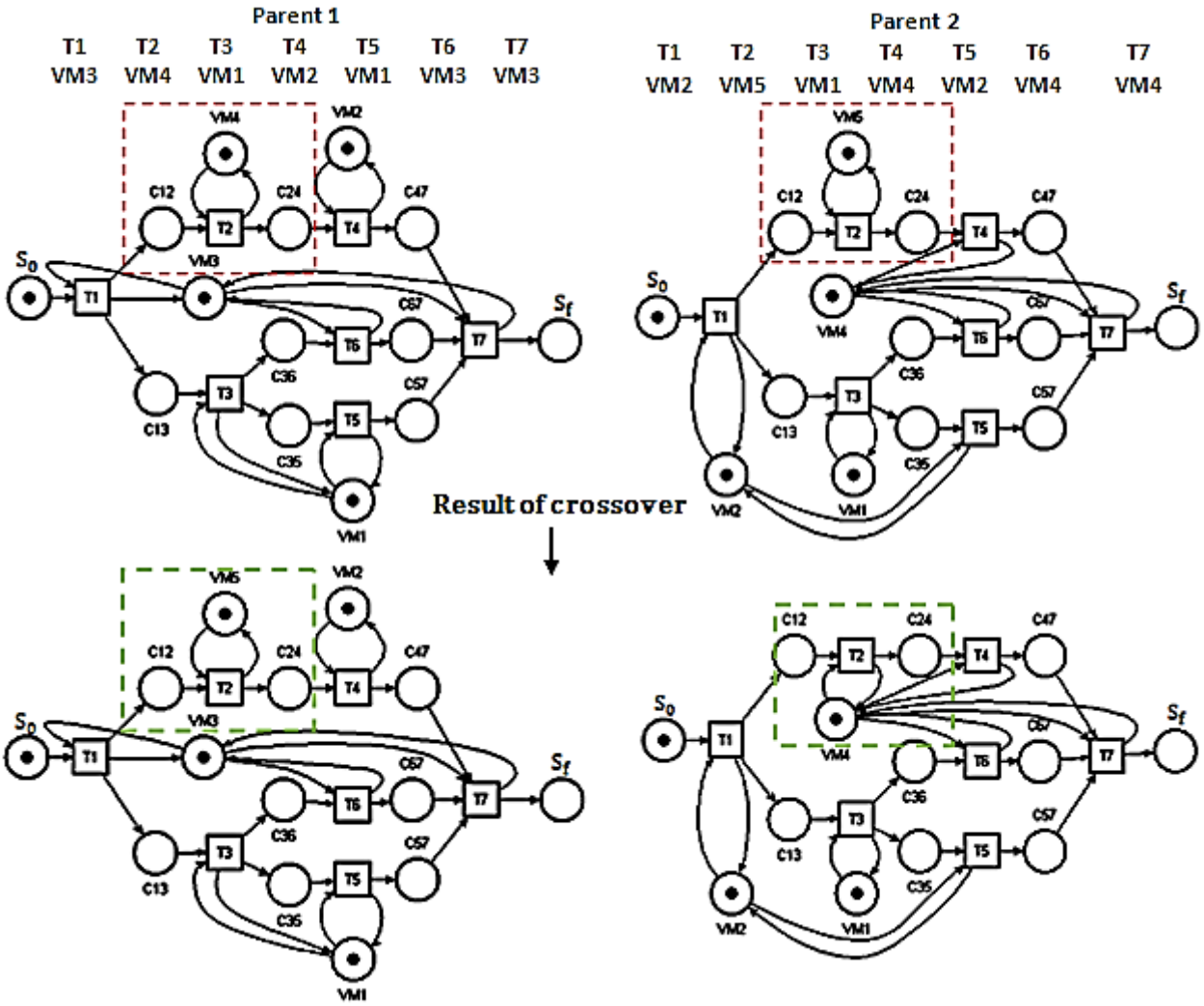


Figure IV.4.Mécanisme d'opérateur de croisement

IV.2.3.2.2 La mutation

L'opérateur de mutation génère un nouveau RPR (fils) d'un seul RPR (parent) dans la population courante. Selon [66], l'opérateur de mutation modifié la structure d'un RPR $\xi(\tau)$ dans Ξ. Nous avons proposé un opérateur de mutation qui associe $\xi(\tau)$ à une nouvelle cohérence RPR ξ $(\tau+ 1)$, selon un couple spécifié $\{p_{in}, t_y\}$. Ce que diffère du travail proposé dans [66], l'opérateur de mutation agissant sur une seule tâche t_y choisie au hasard. L'assignement de cette tâche sera modifié comme suit : l'algorithme vérifie s'il existe une machine virtuelle inutilisable (MV_{in}) dans toute la population initiale ($MV_{in} \in P^{\infty}$), cette MV_{in} exécutera t_y. Sinon ($P^{\infty} = \emptyset$), MV_{in} doivent être choisie au hasard parmi la liste des MV déjà utilisées ($MV_{in} \in P$).

(Le tableau (IV.2) présente les cas des MV_{in} sélectionnées). La figure(IV.5) montre une simple mutation. En suppose dans cet exemple que t_{χ}= T3.

TableauIV.2. Cas des machines virtuelles sélectionnées

N°	Condition	MV_{in} selection	$\xi(\tau + 1)$
1	$P^{\infty} \neq \emptyset$	MV_{in} selectionné de P^{∞}	$P = P \cup MV_{in}$ et $P^{\infty} = P^{\infty} \setminus \{MV_{in}\}$
2	$P^{\infty} = \emptyset$	MV_{in} sélectionné de P	Y'a pas de changement

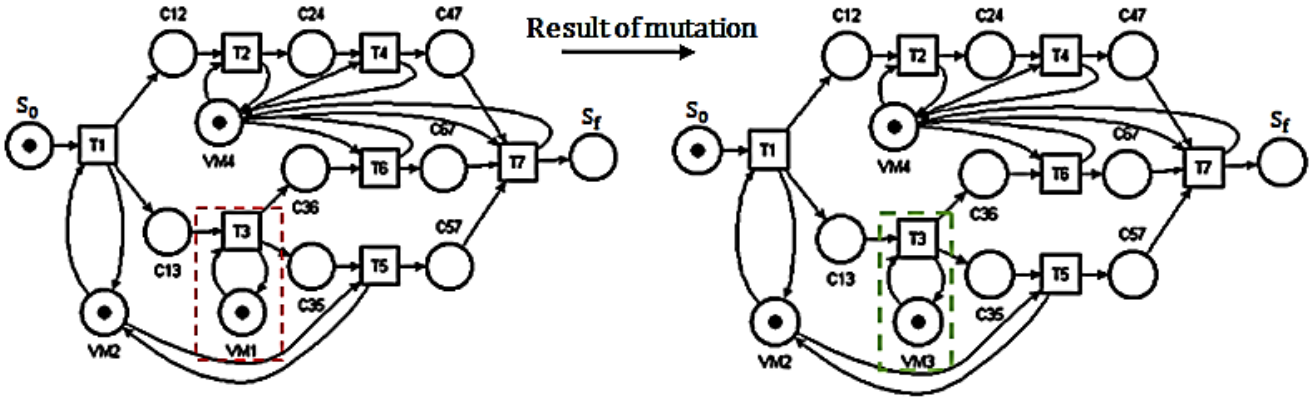


Figure IV.5.Mécanisme d'opérateur de mutation

IV.2.3.2.3 La fonction de fitness

Le rôle de ce concept est de définir la façon d'évaluer la qualité d'une solution. La fonction fitness est calculée pour toutes les solutions possibles, et ensuite une solution avec une meilleure fitness, est obtenue comme minimum ou maximum pour la solution la plus adaptée. L'objectif de notre travail est de minimiser la fitness, en d'autres termes minimiser les deux métriques de qualité de services les plus utilisés dans le *cloud computing* (*makespan* (équation (IV.5), coût (équation (IV.6))). Un chromosome avec une minimum valeur de fitness est considéré comme une meilleure solution. Certains des meilleurs chromosomes seront choisis par sélection du tournoi au cours de l'itération suivante.

$$\text{Fitness} = \begin{cases} \text{Makespan} = \max_{t_i \in T}\{FT(t_i)\} & (\text{IV.}\mathbf{5}) \\ Cost = \sum_{j=1}^{N}\left\{WC_{ex}\left(MV(t_j)\right) + \sum_{p \in pred(t_j)} WC_{tr}\left(MV(t_p), MV(t_j)\right)\right\} & (\text{IV.}\mathbf{6}) \end{cases}$$

Où :

$$WC_{ex} = \begin{cases} EC(t_i, \mathrm{MV}_j) = ET(t_i, \mathrm{MV}_j) \times UEC(\mathrm{MV}) \\ \mathrm{WC_{tr}} = \mathrm{TC}\left(\mathrm{VM(t_i)}, \mathrm{MV(t_j)}\right) = \mathrm{data[i,j]} \times \left(\mathrm{C_{out}(MV(t_i))} + \mathrm{C_{in}}\left(\mathrm{MV(t_j)}\right)\right) \end{cases}$$

Avec:

- TC ($MV(t_i)$,$MV(t_j)$): le coût de transfert des données de la tâche t_i (exécutée sur $MV(t_i)$) vers la tâche t_j (exécutée sur $MV(t_j)$);
- C_{out} (MV (t_i)) : le coût de transfert des données de $MV(t_i)$;
- C_{in} ($MV(t_j)$): le coût de la réception des données sur $MV(t_j)$.

IV.2.3.2.4 Les critères d'arrêt

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

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

IV.2.3.2.5 Le remplacement

Dans ce travail une stratégie de remplacement élitiste est utilisée pour construire la nouvelle population. Dans ce cas, on garde les solutions possédant les meilleures fitness d'une génération à la suivante.

IV.2.3.3 Étapes de l'algorithme LEPN/GA (processus d'ordonnancement)

LEPN/GA est un algorithme élitiste, il assure qu'à chaque nouvelle génération ; les meilleurs individus rencontrés soient conservés. Dans la première étape, l'algorithme ***LEPN/GA*** génère un ensemble des solutions initiales et aléatoires P_0 composé de N individus parents à l'aide de l'algorithme génétique. Cette génération est un ensemble des solutions possibles (tâche-MV). La deuxième étape consiste à convertir cette population à un ***LEPN***, ensuite l'algorithme vérifie tous les individus et déduit l'ordre de franchissement des transitions ; cette dernière opération est guidée par la causalité entre les événements modélisés par l'ensemble des transitions. Chaque transition (événement) sera franchie lorsque ses post-

conditions sont remplies. Une fois une transition franchie, elle rendra un ensemble de post-conditions satisfaites pour tirer d'autres transitions. L'ordre de franchissement est une séquence de transitions qui peut être calculé comme un chemin dans l'arbre d'accessibilité, du marquage initial à un marquage final. Un ordre est défini par une séquence $\sigma=t_1,t_2,...,t_n$. Où t_1 est activée dans le premier marquage M_0: $M_0>t_1$. Et pour tous t_i dansσ, t_i est activée après le franchissement de t_{i-1}. Franchir t_{i-1} cède à un marquage M_{i-1} tel que t_i est activé dans M_{i-1}. Dans l'étape suivante, l'algorithme calcule et étiquette chaque tâche par (équation **(**IV. 3**)**) et équation **(**IV.4**))**. Ensuite, chaque individu sera évalué en utilisant une fonction de fitness (équation(IV.5) et équation (IV.6)). Notre algorithme calcule les rangs pour chaque individu selon le concept de domination de Pareto.

Le processus d'évolution de la population est assuré par différents opérateurs génétiques, notamment, la sélection par tournoi, le croisement, la mutation et le remplacement. L'algorithme itère le processus jusqu'à atteindre un nombre maximum d'itérations ou jusqu'à ce que la valeur globale de la fitness ne s'améliore plus. La figure (IV.6) montre le fonctionnement de l'algorithme proposé (LEPN/GA).

Les grandes lignes de notre algorithme sont décrites dans Algo.1

Algo. 1: LEPN/GA pour l'ordonnancement du *workflow*

Input:

- *Workflow* des tâches (DAG);
- Nombre d'itérations;
- L'ensemble des MVs disponible ((capacité, coût) de chaque VM);

1. Chercher un mappage possible // chaque mappage représente un individu ;
2. Générer une population initial // convertir chaque mappage à un LEPN;
3. Pour tous les individus ; vérifier le réseau et déduire l'ordre de franchissement des transitions ;
4. **Tant que** le critère d'arrêt n'est pas satisfait **Faire**

(a). Calcule l'étiquette de chaque tâche (FT, C);
(b). Calcule de fitness de chaque individu ;
(c). Calcule du rang de chaque individu selon Pareto dominance concept ;
(d). Sélection par Tournoi;
(e). Croisement (taux de croisement = 0.5);
(f). Mutation (taux de mutation = 0.05);

Fin Tant que

Output:l'ensemble des solutions en compromis

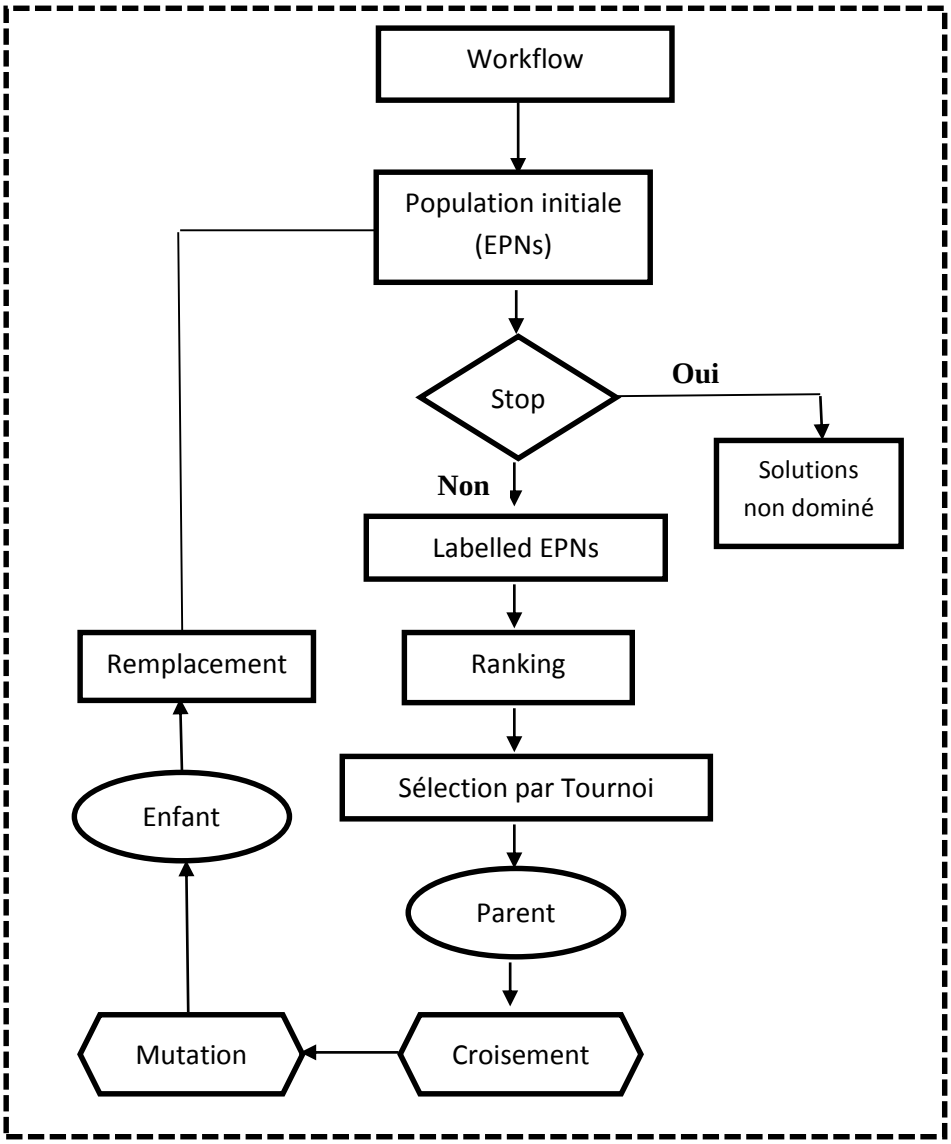


Figure IV.6. Fonctionnement de LEPN/GA

IV.3 Conclusion

Dans ce chapitre nous avons fait une modélisation de notre approche avec une optimisation. La première consiste à modéliser l'approche proposée par ***Labelled Evolutionary Petri Net (LEPN)*** et la deuxième a pour but d'optimiser le réseau de Petri modélisé par l'algorithme génétique (***AG)*** basé sur l'approche Pareto. Le chapitre suivant sera consacré essentiellement à l'implémentation du notre approche.

Chapitre V

Mise en œuvre et résultats

Plan

V.1 Introduction

Ce chapitre décrit la phase de validation de notre approche proposée ***Labelled Evolutionary Petri Net/Genetic Algorithm (LEPN/GA)*** pour l'ordonnancement du *workflow* dans le *cloud computing*. Cette approche consiste à exploiter le paradigme des algorithmes génétiques et le paradigme des réseaux de Petri pour concevoir un système d'ordonnancement du *workflow*.

Dans ce chapitre nous procédons à la validation de notre travail par Petri Net Markup Language (PNML) [73] et du logiciel TiNA Tools [74]. Pour être en mesure de tirer des conclusions fiables sur la précision et la robustesse de notre algorithme, une comparaison a été effectuée entre notre algorithme et trois autres algorithmes. Un test statistique a été effectué afin d'évaluer la qualité des résultats par rapport aux résultats d'autres algorithmes. Un aperçu sur le test statistique est présenté.

L'impact de différents paramètres d'algorithme génétique est examiné. Des résultats généraux de notre algorithme sont présentés avec discussions sur leurs valeurs.

V.2 Validation de l'approche

La validation de notre nouvelle approche se fait en trois processus principaux (figure V.1):

- **Le processus (a)** : ce processus consiste à mapper l'ensemble des tâches à l'ensemble des machines virtuelles pour générer une population initiale (tâche-MV), l'opération de ce processus à besoin de trois entrées qui sont :

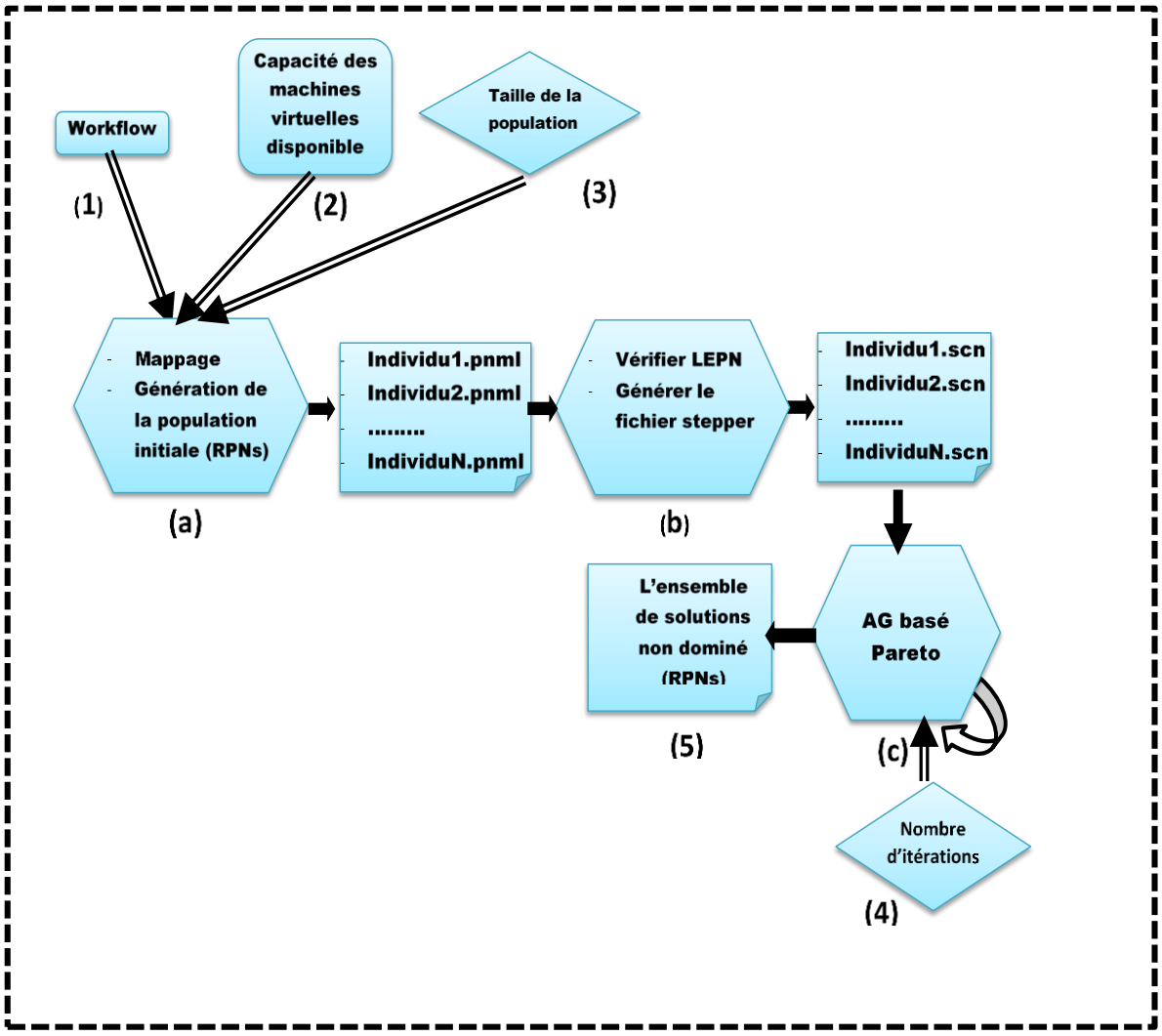


Figure V.1. Les étapes de validation expérimentale

1) *Workflow* : c'est un ensemble de tâches, chaque tâche est reliée par ses prédécesseurs et ses successeurs, sa taille et sa quantité des données sortant vers la tâche suivante. Le code suivant montre le fichier.xml d'un *workflow* composé de sept tâches

```
<workflow  id="00">
<T id="1">
<cond></cond>
<out><C>12</C><C>13</C></out>
<size>9600</size>
<DATA_O>12</DATA_O>
</T>
<T id="2">
<cond><C>12</C></cond>
<out><C>24</C></out>
<size>54000</size>
<DATA_O>46</DATA_O>
</T>
<T id="3">
<cond><C>13</C></cond>
<out><C>35</C><C>36</C></out>
<size>5100</size>
<DATA_O>30</DATA_O>
</T>
<T id="4">
<cond><C>24</C></cond>
<out><C>47</C></out>
<size>54000</size>
<DATA_O>88</DATA_O>
</T>
<T id="5">
<cond><C>35</C></cond>
<out><C>57</C></out>
<size>30600</size>
<DATA_O>34</DATA_O>
</T>
<T id="6">
<cond><C>36</C></cond>
<out><C>67</C></out>
<size>14400</size>
<DATA_O>55</DATA_O>
</T>
<T id="7">
<cond><C>47</C><C>57</C><C>67</C></cond
>
<out></out>
<size>25500</size>
<DATA_O>23</DATA_O>
</T>
</workflow>
```

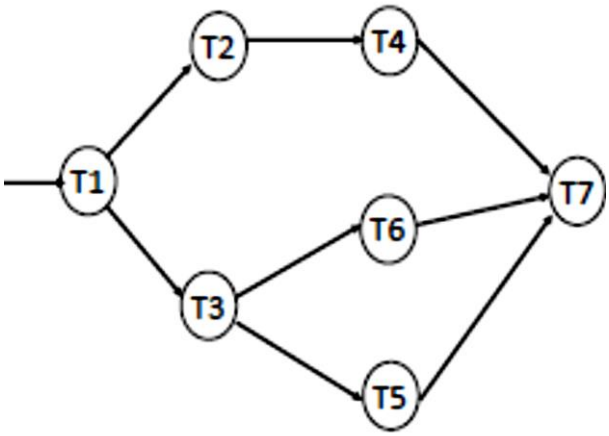


Figure V.3. Exemple d'un workflow

Figure V.2. Représentation de l'exemple d'un workflow par le langage XML

2) La configuration des MVs : c'est un ensemble des machines virtuelles qui sont disponible pendant tout le cycle l'ordonnancement, le code suivant montre le fichier.xml d'un exemple de trois machines virtuelles avec ses différentes caractéristiques.

```
<Config id="00">
<VM id="1">
<MIPS>4800</MIPS>
<COST><in>0,37</in><out>0,37</out></COST
>
<BW><VM2>6</VM2><VM3>7</VM3></BW
>
<UEC>0,3</UEC>
</VM>
<VM id="2">
<MIPS>5400</MIPS>
<COST><in>0,45</in><out>0,98</out></COST
>
<BW><VM1>1</VM1><VM3>3</VM3></BW
>
<UEC>0,9</UEC>
</VM>
<VM id="3">
<MIPS>8500</MIPS>
<COST><in>0,79</in><out>0,55</out></COST
>
<BW><VM1>3</VM1><VM2>5</VM2></BW
>
<UEC>0,5</UEC>
</VM>
</Config>
```

Figure V.4. Représentation de l'exemple de configuration des MVs par le langage XML

Tableau V.1. Exemple d'une configuration des machines virtuelles

MV	*Vitesse*	*Cout*		*Bande Passante*		*Cout unitaire*
		in	*out*	*in*	*out*	
MV1	**4800**	**0,37**	**0,37**	**6**	**7**	**0,3**
MV2	**5400**	**0,45**	**0,98**	**1**	**3**	**0,9**
MV3	**8500**	**0,79**	**0,55**	**3**	**5**	**0,5**

3) La taille de la population : qui reste constante pendant toute l'exécution de l'algorithme.

Une fois le processus (a) termine son exécution, il génère un ensemble des individus sous format des fichiers Pnml (individu1.pnml,…,individun.pnml) selon la taille de la population.

Le processus (b) : ce processus consiste à convertir chaque individu représenter sous format Pnml à un réseau de Petri étiqueté (étiqueter chaque tâche du réseau et calcule la fitness de tout le réseau), ce dernier sera vérifié (détection des inter-blocages) et ordonnancé par le simulateur stepper de TiNA Tools. Le résultat de ce processus est un ensemble des individus ordonnancés et étiquetés).

- **Le processus (c) :** un algorithme génétique et l'opération de *ranking* de Pareto a été appliquésur chaque individu. L'AG termine son exécution selon le nombre d'itération donné.

Le résultat de ce processus est un ensemble de solutions ***LEPNs*** non-dominé (figure V.5).

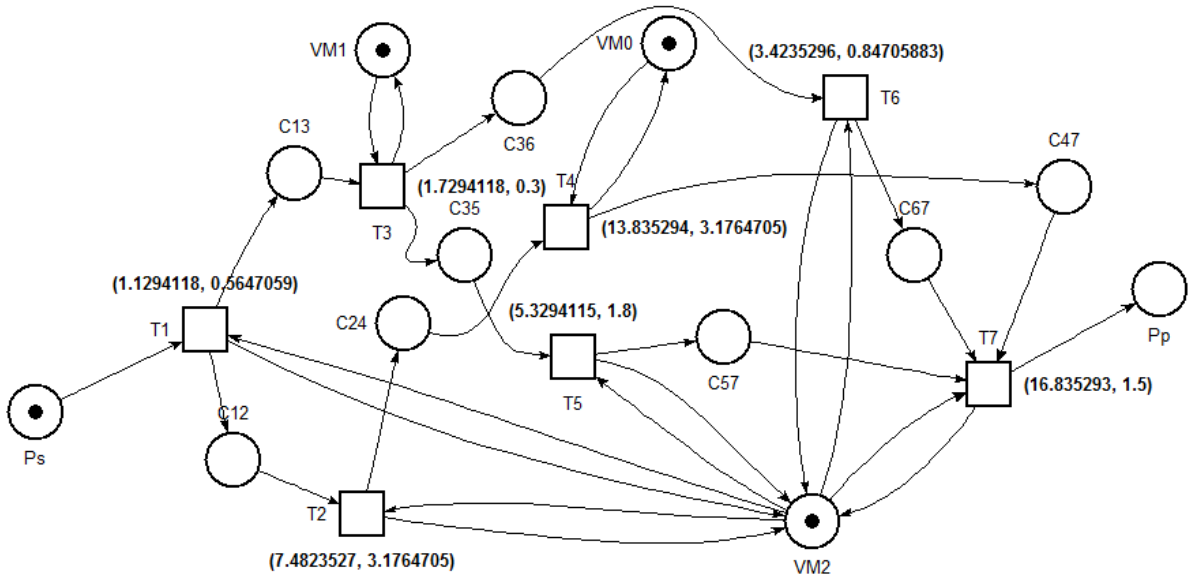


Figure V.5. Représentation d'une solution sous forme de LEPN.

• V.3 Outil de simulation

TiNA Tools est utilisé pour vérifier le réseau de Petri (représentation graphique du fichier PNML), la vérification analyse si le réseau est correcte ou non. Il est possible d'analyser toutes les séquences de franchissement d'un réseau de Petri (RdP) correspondant à un problème d'ordonnancement. Chaque séquence de franchissement de transitions commençant du marquage initial vers un marquage final correspond à un ordonnancement réalisable.

Une séquence de transition de l'état initial S_0 à l'état final S_f modélise une exécution ordonnée des tâches et une utilisation ordonnée des machines virtuelles. Cette séquence est enregistrée dans un fichier stepper utilisé pour étiqueter et pour calculer la fitness de chaque individu. L'utilisation du format PNML est argumentée par : la lisibilité, l'universalité et la mutualité. Le format devrait nous permettre d'extraire autant d'informations que possible d'un RdP même si le type de réseau Petri est inconnu.

Par conséquent, le format doit extraire les principes communs et les notations communes des réseaux de Petri, ce qui facilite la manipulation de réseau de Petri évolutionnaire (RPE), en particulier dans la phase d'étiquetage, et l'application des deux opérateurs : croisement et mutation.

D'autre part, dans notre implémentation, nous avons utilisé le langage java, l'utilisation du format PNML facilite l'exportation de réseaux Petri vers les outils TiNA et pour l'importation de réseaux à partir d'Eclipse.

V.4 Paramètre expérimentaux

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

V.4.1 Paramètres du *workflow*

Nous utilisons une application *workflow FFT (Fast Fourier Transform)* [75] composé de 194 tâche figure (V.6), dont certaines données sont générées aléatoirement. La quantité de données à transférer entre les tâches varie de10 MB à 1024 MB.

V.4.2 Paramètres de l'infrastructure IaaS

Ces expériences ont été menées à l'aide des ressources dont les caractéristiques sont basées sur les données provenant des benchmarks effectués sur les ressources *cloud Amazon EC2* [76] et les ressources *cloud* similaires provenant d'autres fournisseurs [6]. Les caractéristiques de ces ressources sont présentées dans le tableau (V.2). Nous utilisons le modèle de facturation donné par *Amazon cloud Front* [77] pour le coût de transfert unitaire de données entre les MV. Nous supposons que les vitesses de transferts moyens entre les instances sont fixées à 100 Mbps.

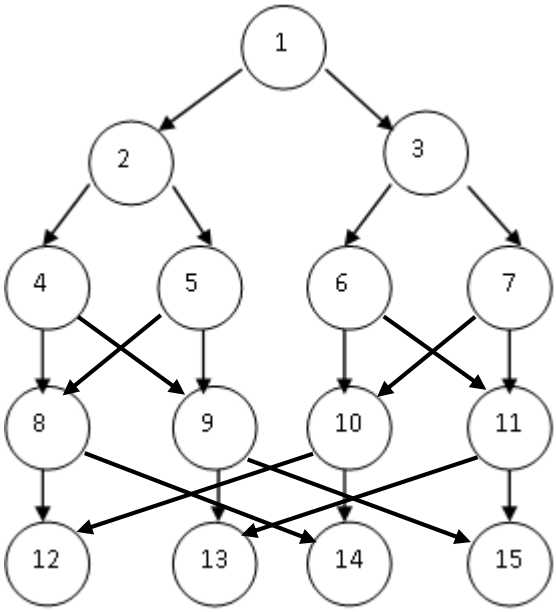


Figure V.6. Graphe FFT

Tableau V.2. Caractéristiques des ressources

	Vitesse (Mips)	**Prix ($/h)**
m1.xlarge	5400	0.92
m2.xlarge	4800	0.57
m2_2xlarge	8500	1.14

V.4.3 Paramètres de l'algorithme génétique (AG)

Les différents paramètres de l'algorithme AG sont décrits dans le tableau (V.3).

Tableau V.3. Paramètres de l'algorithme génétique

Paramètre	**Valeur**
Taille de population	100
Nombre de générations	100
Taux de croisement	50%
Taux de mutation	5%
Taux de sélection	50%

V.5 Résultats et discussion

Dans cette section, nous présentons les résultats obtenus après l'exécution de notre algorithme, la figure (V.7) et la figure (V.8) montrent respectivement la distribution de la population initiale de la première itération et ses différents rangs.

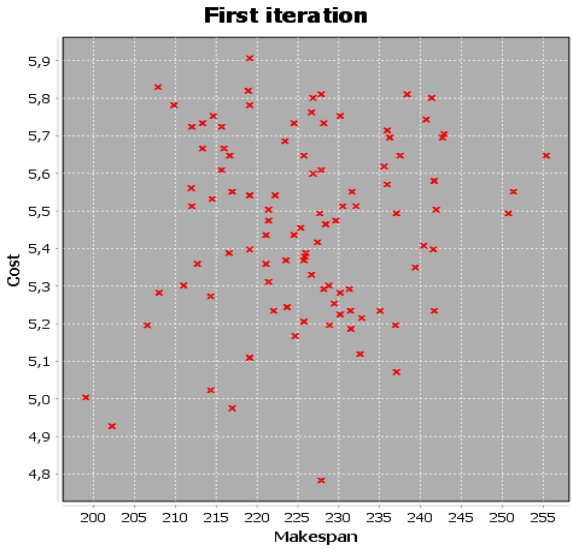


Figure V.7. La distribution de la population initiale

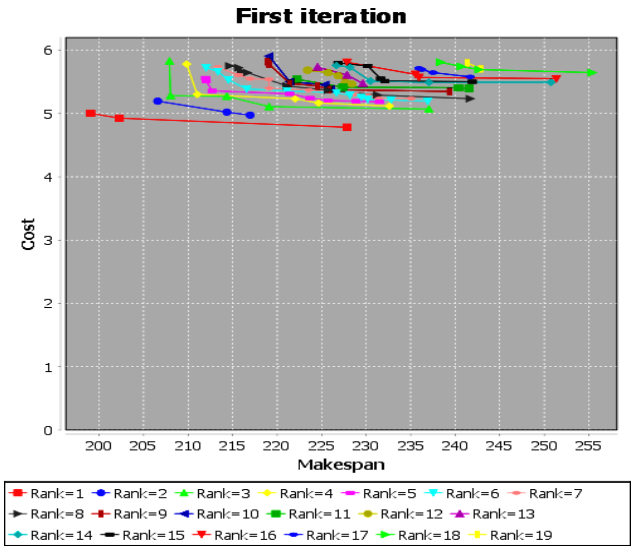


Figure V.8. Les rangs de la population initiale

Les résultats obtenus (la distribution des solutions après 100 itérations et ses différents rangs) sont illustrés sur les figures (V.9) et (V.10). La difficulté de cet algorithme est le choix final de la solution Pareto optimale, parmi toutes les solutions présentées.

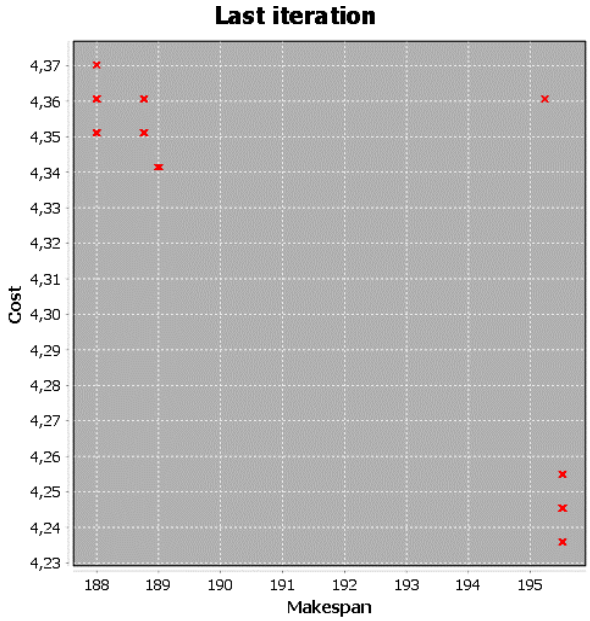


Figure V.9. La distribution des solutions

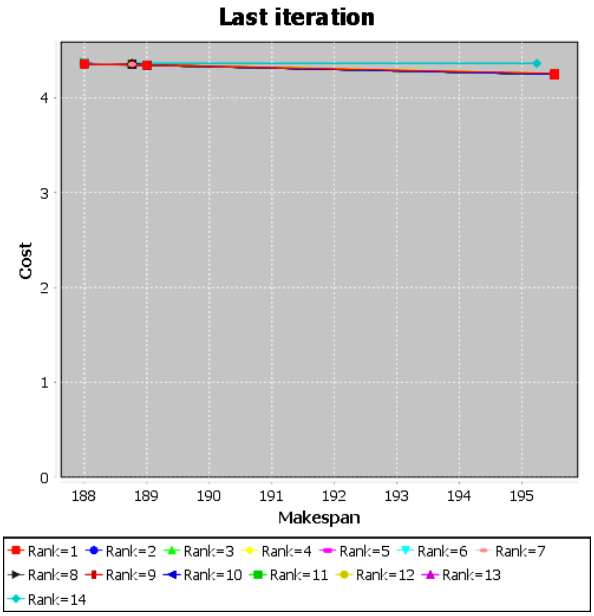


Figure V.10. Les rang des solutions

V.5.1 La convergence de l'algorithme LEPN/GA

Après l'exécution de notre algorithme ***LEPN/GA***, les résultats montrent l'efficacité de génération de la solution optimale pour les deux métriques (coût, *makespan*).

La figure (V.11) et la figure (V.12) montrent la convergence de l'algorithme après 100 générations.

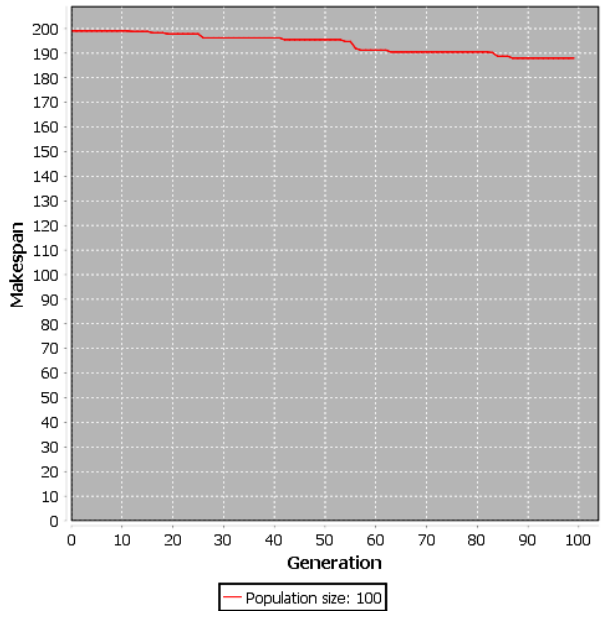


Figure V.11. Convergence de l'algorithme (*makespan*)

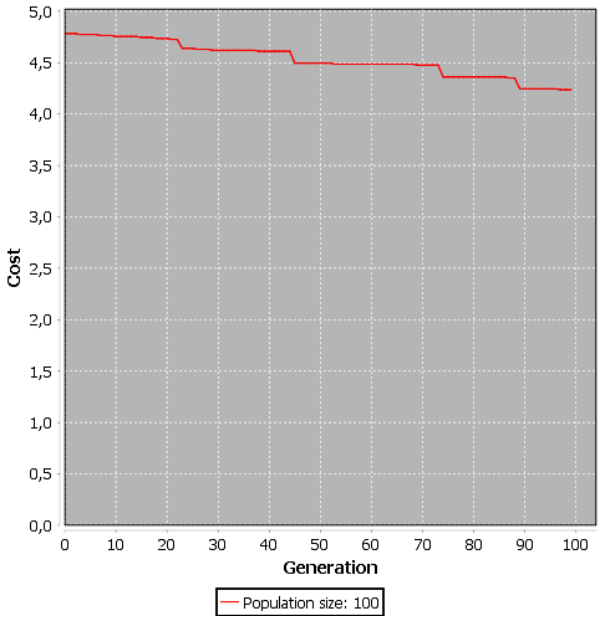


Figure V.12. Convergence de l'algorithme (coût)

V.5.2 Comparaison des différents algorithmes

Pour confirmer la qualité des solutions obtenue par l'algorithme proposé ***LEPN/GA***, nous avons comparé notre résultats avec les résultats des trois autre algorithmes : l'algorithme génétique amélioré (GA*), l'algorithme *NSGAII* et l'algorithme *HEFT*. Nous avons appliqué ces trois algorithmes sur l'application *workflow Fast Fourier Transform* (*FFT*) composé par 194 tâches pour10 machines virtuelles. La figure (V.13) et (V.14) montrent un histogramme de des solution(s) trouvée(s) par chaque méta-heuristique d'ordonnancement du *workflow* sur 10 MVs, optimisant deux métriques de *QoS* (*makespan*, coût).

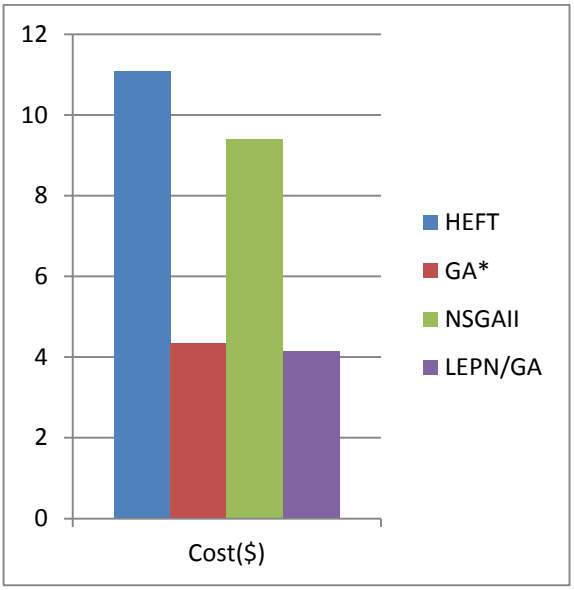


Figure V.13.les valeurs de coût obtenu par l'ensemble des algorithmes

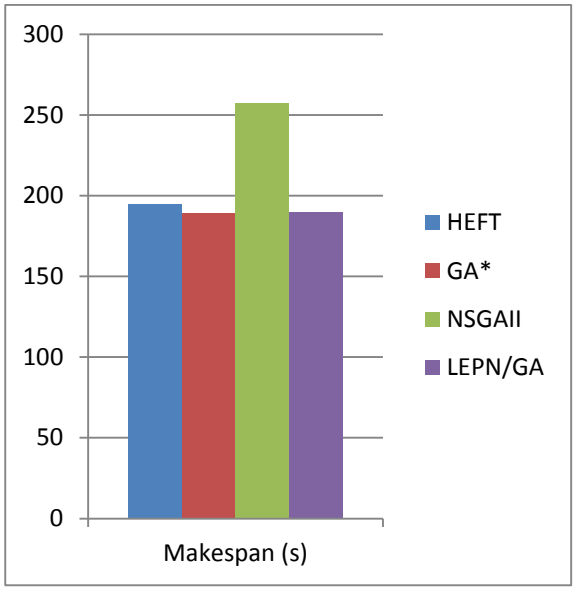


Figure V.14.les valeurs de *makespan* obtenue par l'ensemble des algorithmes

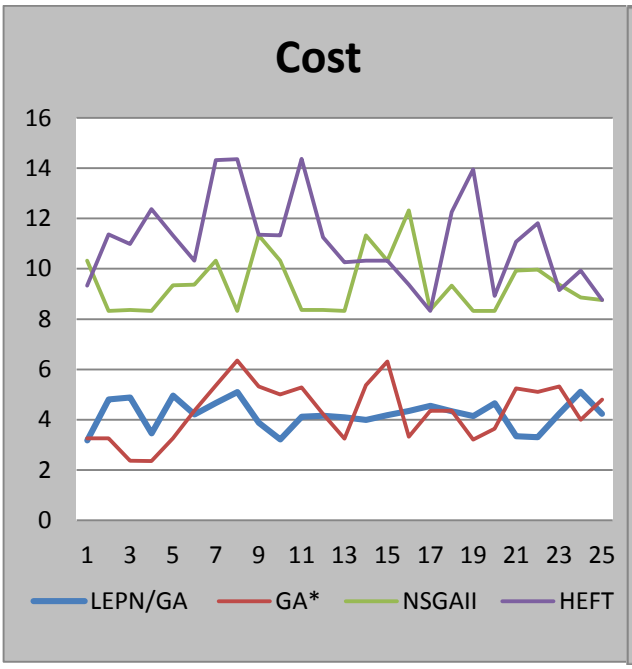
Le tableau (V.4) présente les résultats des moyennes et l'écart type obtenue après de 25 exécutions. Dans chaque exécution, la population initiale est composée de 100 chromosomes et le nombre d'itération égale à 100 générations. Nous observons que notre algorithme donne l'écart-type le plus bas para port les deux objectifs ; donc les valeurs de fonction fitness sont très proches et la dispersion des mesures autour de la moyenne est faible.

Les résultats expérimentaux ont tendance à confirmer aussi que notre algorithme donne une meilleure moyenne de coût que les trois autres algorithmes, et il donne également une meilleure moyenne de *makespan* que *NSGAII* et *HEFT*.

Tableau V.4. La moyenne et l'écart type des différents algorithmes

	LEPN/GA		**GA***		**NSGAII**		**HEFT**	
	Coût	**Makespan**	**Coût**	**Makespan**	**Coût**	**Makespan**	**Coût**	**Makespan**
Moyenne	4,151	189,513	4,349	189,005	9,385	257,591	11,086	194,977
L'écart type	0,174	2,886	1,104	2,967	1,153	16,295	1,770	6,524

La figure (V.15) et (V.16) montrent la convergence moyenne de chaque algorithme après 25 itérations, nous observons que la fitness moyenne pour les deux objectifs de l'algorithme ***LEPN / GA*** donne de meilleurs résultats pendant 25 itérations, ce qui n'est pas le cas pour deux autres algorithmes (*NSGAII* et *HEFT*).



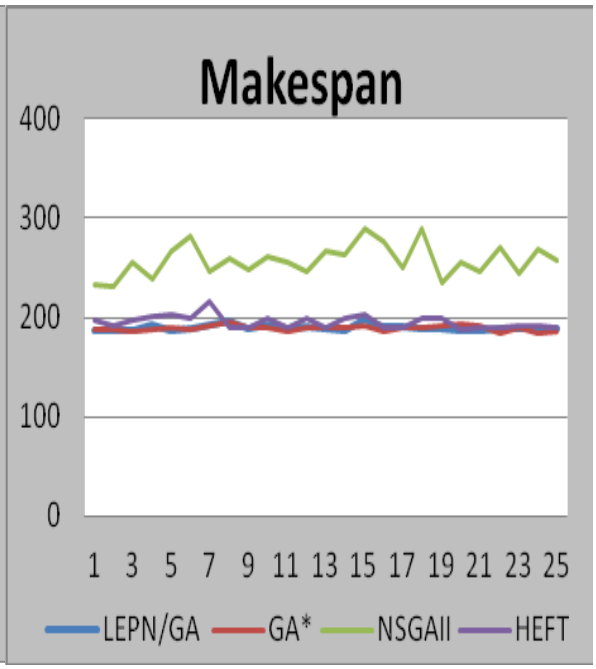


Figure V.15.Résultats pour le coût.

Figure V.16. Résultats pour le *makespan*.

La figure (V.17) et (V.18) montrent le changement des valeurs de l'écart type de l'algorithme ***LEPN / GA*** pendant les 25 itérations pour les deux objectifs.

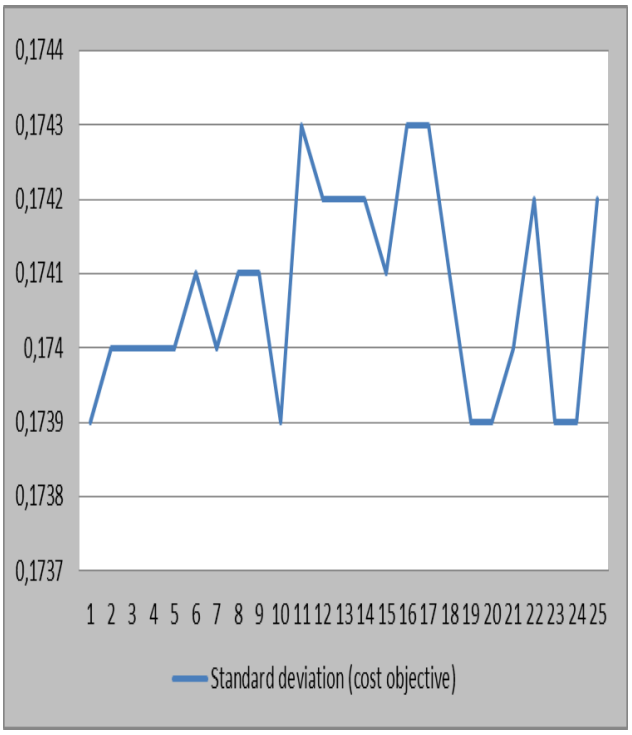


Figure V.17.l'écart type pour le coût

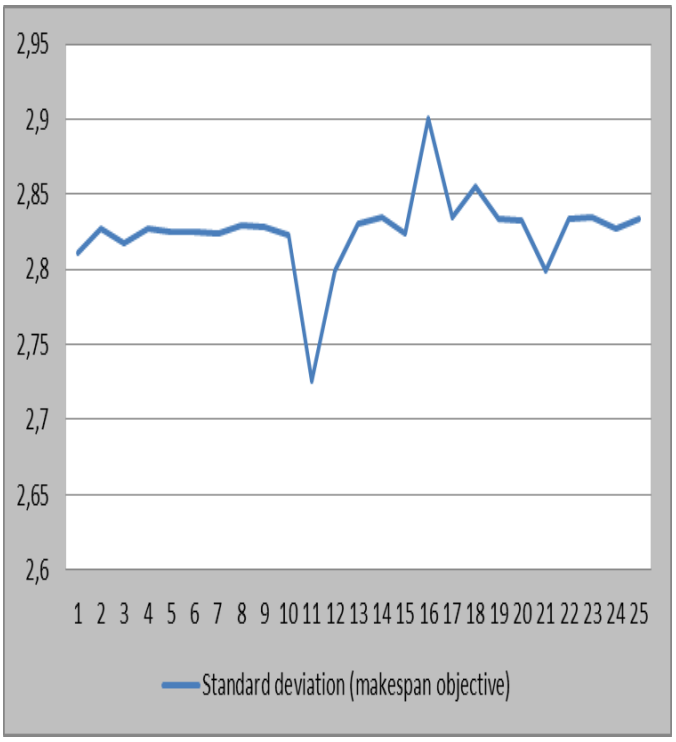


Figure V.18.l'écart type pour le *makespan*

V.6 Test statistique

Afin d'évaluer la qualité des résultats, nous avons comparé notre moyenne de fitness de *makespan* et le coût avec les autres moyennes de fitness des autres algorithmes. Nous utilisons un test statistique (test d'hypothèse) [78]. La méthode de test d'hypothèse utilise des tests d'importance pour détermine la probabilité qu'une déclaration (souvent liée à la moyenne ou à la variance d'une distribution donnée) est vrai, et à quelle probabilité nous accepterions la déclaration comme vrai [79].

Étant donné que les résultats obtenus sont normalement distribués (confirmés à l'aide du test en ligne de Shapiro-Wilk[1]), nous utilisons le test t apparié [80] car les deux conditions sont remplies: (i) la taille de l'échantillon est inférieure à 30 (N = 25), et (ii) les trois algorithmes sont exécutés sur la même configuration des machines virtuelles et du *workflow* graphe.

Les hypothèses de test t d'échantillon appariées sont définies comme suit:

1. L'hypothèse nulle H_0 (suppose que la différence des moyennes réelles (μ_d) est égale à zéro, il n'y a pas de différence significative entre deux moyennes).

[1]http://www.xlstat.com/fr/solutions/fonctionnalites/tests-de-normalite

2. The upper-tailed hypothèse alternativeH_1 (suppose que la différence des moyennes (μ_d) est supérieur à zéro, il y'a une différence significative entre deux moyennes).

Pour calculer le simple t-test apparié, nous avons besoin de calculer la moyenne, la variance ($\sigma 2$), et t (t-statistique) avec freedom degré égale à($N-1$) (dans notre cas le degré de freedom = $25-1= 24$).

Pour évaluer les résultats, nous avons besoin de calculer p-value (la valeur de probabilité). Autant que p est petit, autant que nous pouvons rejeter l'hypothèse nulle, nous pouvons donc rejeter l'hypothèse nulle lorsque la valeur de p est inférieure à un niveau de signification $\alpha = 0,05$.Le tableau (V.5) et le tableau (V.6) présentent les valeurs de t et p pour les deux objectives entre LEPN/GA et les trois autres algorithmes.

Le t-test apparié confirme la différence significative entre les moyennes de ***LEPN/GA*** et les deux algorithmes *NSGAII*, *HEFT* pour les deux objectives (rejeté l'hypothèse nulle).
D'autre part, p-values confirme que la différence entre les moyennes obtenue pour ***LEPN/GA*** et *GA** est zéro (il y'a pas une différence significative entre les deux moyennes) pour les deux objectifs.

Tableau V.5. Résultats de t-test apparié (coût).

	GA*	**NSGAII**	**HEFT**
t	-0,928	-23,366	-19,850
p-value	0,181	< 0,0001	< 0,0001
interprétation	p-value>0,05 We cannot reject the null hypothesis H_0.	p-value < 0,05 We must reject the null hypothesis H_0, and retain the alternative hypothesis H_1.	p-value < 0,05 We must reject the null hypothesis H_0, and retain the alternative hypothesis H_1.

Tableau V.6. Résultats de t-test appariée (*makespan*).

	GA*	**NSGAII**	**HEFT**
t	0,780	-21,499	-4,355
p-value	0,788	< 0,0001	0,000
interpretation	p-value>0,05 We cannot reject the null hypothesis H_0.	p-value < 0,05 We must reject the null hypothesis H_0, and retain the alternative hypothesis H_1.	p-value < 0,05 We must reject the null hypothesis H_0, and retain the alternative hypothesis H_1.

V.7 Influence les paramètres de l'AG sur la convergence de l'algorithme

V.7.1 Influence la taille de la population

Dans cette expérimentation, nous avons testé notre algorithme avec plusieurs populations des différentes tailles. Par conséquent, la mise à l'échelle de la taille donne une meilleure valeur du *makespan* et du coût. Cette amélioration est due à la diversité de la population initiale (solutions réalisables). La figure (V.19) et la figure (V.20) montrent les résultats de *makespan* et le coût de la population composée de 60, 80 et 100 individus.

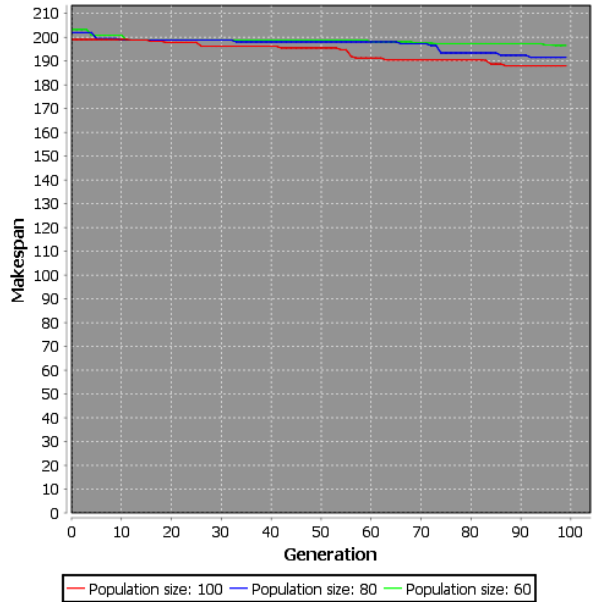


Figure V.19. Influence la taille de la population sur le *makespan*

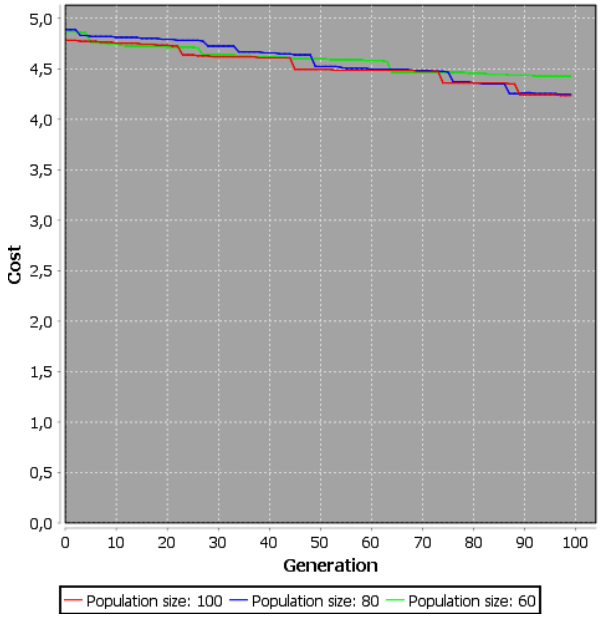


Figure V.20. Influence la taille de la population sur le coût

V.7.2 Influence le taux de croisement

Dans notre test expérimental, nous avons testé notre algorithme avec des différents taux de croisement. Nous avons obtenu une meilleure valeur pour le *makepan* et le coût lorsque le taux de croisement est égal à 0.5. La figure (V.21) et la figure (V.22) montrent les résultats du *makepan* et le coût lorsque le taux de croisement varie dans l'intervalle [0.3, 0.7].

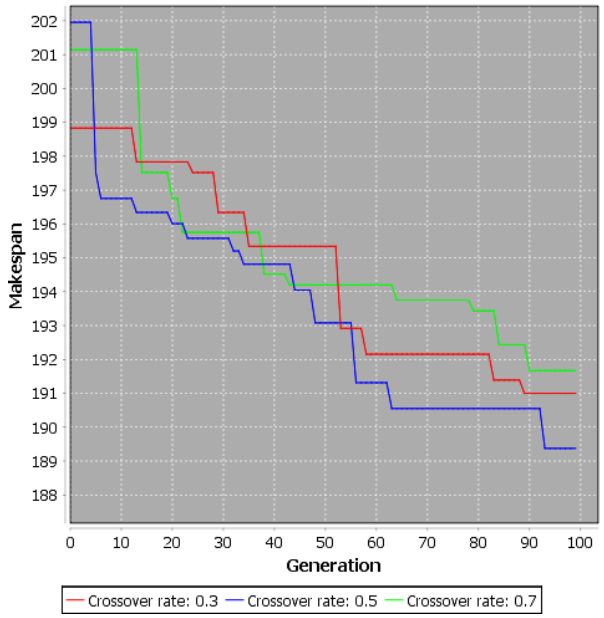


Figure V.21. Influence le taux de croisement sur le *makespan*

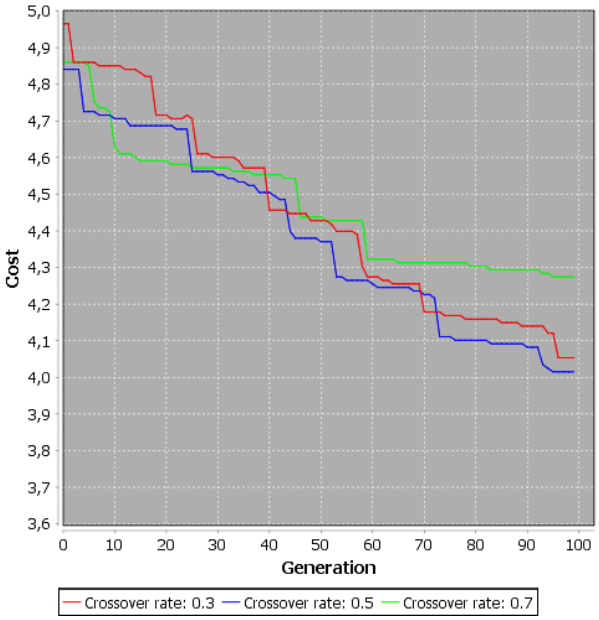


Figure V.22. Influence le taux de croisement sur le coût

V.7.3 Influence le taux de mutation

Nous avons testé notre algorithme avec des différents taux de mutation. Nous observons que lorsque la valeur est égale à 0,05, l'algorithme converge vers une meilleure valeur de coût que pour *makespan,* ce qui n'est pas le cas pour la valeur 0,07. D'autre part, lorsque le taux est égal à 0,03, l'algorithme donne de mauvais résultats pour les deux objectifs. La figure (V.23) et (V.24) montrent les résultats du *makepan* et le coût lorsque le taux de mutation varie dans l'intervalle [0.03, 0.07].

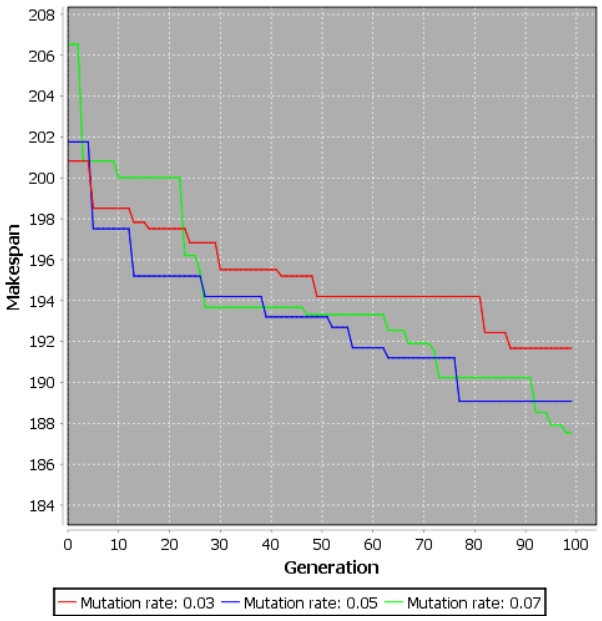


Figure V.23. Influence le taux de mutation sur le *makespan*

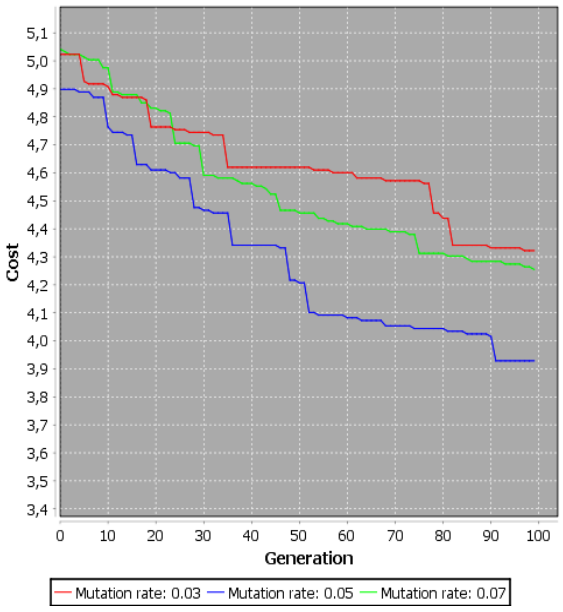


Figure V.24. Influence le taux de mutation sur le coût

V.8 Conclusion

Au cours de cette dernière étape de notre travail, nous avons réalisé un algorithme basé sur les algorithmes génétiques pour l'ordonnancement de *workflow* dans le *cloud computing*, et nous avons présenté les outils utilisés pour l'implémentation.

Nous avons montré à travers cet algorithme l'intérêt de l'utilisation des réseaux de Petri pour l'ordonnancement des *workflows*, ainsi que l'intérêt des algorithmes génétiques pour la résolution des problèmes multi-objectifs. Nous avons utilisé un test statistique pour évaluer les différents résultats obtenus et confirmer la robustesse de notre algorithme.

Conclusion générale

Plan

I. Conclusion

L'un des problèmes fondamentaux de *cloud computing* est l'ordonnancement des tâches d'une façon optimale. Un ordonnancement consiste à déterminer l'ordre dans lequel les différentes tâches des processus doivent s'exécuter et les ressources que chacune d'elles doit utiliser. En outre, les utilisateurs sont confrontés au problème du choix des meilleures ressources (fournisseurs) à utiliser.

Le problème d'ordonnancement reste ouvert dans le domaine du *cloud computing*. En outre, il n'existe aucun algorithme d'ordonnancement traitant le problème de la tâche concurrente aux ressources disponibles.

Dans notre étude, nous avons discuté le problème d'ordonnancement des *workflows*. Ce problème est considéré comme étant un problème d'optimisation multi-objectif en raison de la nature conflictuelle des métriques de *QoS* à prendre en compte.

Dans le cadre de cette étude, nous avons proposé une approche de calcul évolutif LEPN/GA pour l'optimisation de *workflow* dans l'environnement *cloud*. Notre approche est basée à la fois sur les réseaux de Petri et l'algorithme génétique pour trouver la solution optimale de Pareto. Premièrement, nous avons modélisé toutes les solutions réalisables (mapping) par ***Labeled Evolutionary Petri Nets (LEPN)***, cette représentation garantit le respect de contrainte de précédence entre les différentes tâches dans lesquelles le calcul de la fitness est généré automatiquement et évolutif.

Les solutions modélisées représentent la population initiale de l'AG (l'algorithme génétique). Après l'exécution de plusieurs processus d'AG, un ensemble de solutions de compromis est généré. Les évaluations expérimentales montrent l'efficacité de notre approche et prouvent l'utilité de l'approche basée sur les réseaux de Petri.

Le test statistique confirme que l'algorithme proposé "*LEPN*/GA" donne de meilleurs résultats que deux autres algorithmes (*NSGA* II et *HEFT*). Ce test confirme aussi qu'il n'y a pas de différence significative entre la moyenne de LEPN/GA et GA* et confirme que GA* est plus performant en ce qui concerne l'algorithme *NSGA* II et *HEFT*.

II. Perspectives

Suite au travail présenté dans cette thèse, de nouvelles activités de recherche peuvent être lancées afin d'améliorer le travail présenté. Les perspectives que nous proposons peuvent donc s'orienter vers les directions suivantes :

- Prendre en compte d'autre métrique de qualité de service (énergie, sécurité, disponibilié …etc),
- Renforcer notre approche avec une autre technique d'optimisation,
- Introduire une nouvelle notion d'optimalité à part Pareto,
- Nous avons l'intention d'améliorer notre algorithme en considérant l'utilisation d'autres techniques de sélection et de croisement,
- Tester notre approche sur d'autres études de cas pour prouver son utilité.

Bibliographie

[1] Malvaut-Martiarena, W., Vers une architecture pair-à-pair pour l'informatique dans le nuage, thèse de Doctorat, Grenoble, 2011.

[2] Nabil B. and Abdellah T., Cloud-based Web Application Firewal, Mémoire de master en informatique, Université Saad Dahleb, Algérie, 2013.

[3] Mell, P., and Grance, T. (2011). The NIST definition of cloud computing.

[4] Buyya, R., Broberg, J., and Goscinski, A. M. (Eds.). (2010). Cloud computing: Principles and paradigms (Vol. 87). John Wiley and Sons.

[5] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), (pp 50-55).

[6] Yassa, S. Allocation optimale multicontraintes des workflows aux ressources d'un environnement Cloud Computing, thése de Doctorat, Cergy-Pontoise.2014.

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

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

[9] Buyya, R., Yeo, C. S., and Venugopal, S. (2008). Market-oriented cloud and atmospheric computing: Hype, reality, and vision. In 10th IEEE Proc International Conference on High Performance Computing and Communications (HPCC-08) on (pp. 5-13).

[10] Rimal, B. P., Choi, E., & Lumb, I. (2009). A Taxonomy and Survey of Cloud Computing Systems. NCM, 9, (pp.44-51).

[11] Egwutuoha, I. P., Chen, S., Levy, D., Selic, B., and Calvo, R. (2012, November). A proactive fault tolerance approach to High Performance Computing (HPC) in the cloud. In Cloud and Green Computing (CGC), Second International Conference on (pp. 268-273).

[12] http://baremetalcloud.com/index.php/en/

Bibliographie

[13] http://www.softlayer.com/, 2014.

[14] http://www.grid5000.fr, 2014

[15] http://aws.amazon.com/ec2/, 2012.

[16] http://www.rackspace.com/cloud/

[17] http://www.eucalyptus.com/eucalyptus-cloud, 2014.

[18] http://www.nimbusproject.org/, 2014.

[19] http://www.opennebula.org/, September 2012.

[20] Daubert E., Adaptation et Cloud Computing : un besoin d'abstraction pour une gestion transverse", thèse de Doctorat en informatique, Institut national des sciences appliquées de Rennes, France, 2013.

[21] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., and Leaf, D. (2011). NIST cloud computing reference architecture. NIST special publication, 500(2011), 292.

[22] Wood, T., Shenoy, P. J., Venkataramani, A., and Yousif, M. S. (2007, April). Black-box and Gray-box Strategies for Virtual Machine Migration. In NSDI (Vol. 7), (pp. 17-17).

[23] Femmam, M., Kaza,O., Baarir, S. , Fareh, M. El-k., , "A Comparative Study of Levels Scheduling Algorithms in the Cloud Computing," Int'l Journal of Computing, Communications & Instrumentation Engg. (IJCCIE), (vol. 2), (pp. 196-200).

[24] Li, Y., Liu, Y., and Qian, D. (2009, December). A heuristic energy-aware scheduling algorithm for heterogeneous clusters. In Parallel and Distributed Systems (ICPADS), 15th International Conference on (pp. 407-413).

[25] Hind. R, Approche formelle pour la modélisation et la vérification du controle d'accès et des contraintes temporelle dans les systèmes d'information, thèse de doctorat en génie informatique, École Polytechnique de Montréal, 2009.

[26]

Barker, A., and Van Hemert, J. (2007, September). Scientific workflow: a survey and research directions. In International Conference on Parallel Processing and Applied Mathematics (pp. 746-753). Springer, Berlin, Heidelberg.

[27] Davenport, T. H. (1993). Process innovation: reengineering work through information technology. Harvard Business Press.

Bibliographie

[28] Weske, M. (2010). Business process management: concepts, languages, architectures. Springer Publishing Company, Incorporated.

[29] Mehta, C., Freeman G., Deelman, T., Keahey, E. , Berriman, K., B., and Good, J. Hoffa, (December 2008), On the Use of cloud computing for Scientific Workflows, Proc. of Fourth IEEE Int. Conf. on e-Science, (pp. 640-645).

[30] Yang, Y., Liu, K., Chen, J., Liu, X., Yuan, D., and Jin, H. (2008, December). An algorithm in SwinDeW-C for scheduling transaction-intensive cost-constrained cloud workflows. In eScience, eScience'08. IEEE Fourth International Conference on (pp. 374-375).

[31] Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., and Good, J. (2008, December). On the use of cloud computing for scientific workflows. In eScience, eScience'08. IEEE Fourth International Conference on (pp. 640-645).

[32] Pareto, V. (1896). Politique, Cours D'economie. Rouge, Lausanne, Switzerland.

[33] Collette, Y. Contribution à l'évaluation et au perfectionnement des méthodes d'optimisation multiobjectif: application à l'optimisation des plans de rechargement de combustible nucléaire thèse de Doctorat, Université Paris-Est Créteil Val de Marne (UPEC), 2002.

[34] Cheriet, A. Métaheuristique hybride pour l'optimisation multi-objectif, thèse de Doctorat, Université Mohamed Khider-Biskra, 2016.

[35] Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning, 1989. Reading: Addison-Wesley.

[36] Fonseca, C. M., and Fleming, P. J. (1993, June). Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization. In Icga (Vol. 93, No. July), (pp. 416-423).

[37] Srinivas, N., and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary computation, 2(3), (pp.221-248).

[38] Zitzler, E., and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE transactions on Evolutionary Computation, 3(4), (pp.257-271).

Bibliographie

[39] E. Laumanns, M and Thiele,L, Z. (2002), SPEA2: Improving the strength pareto evolutionary algorithm, Proc. of Evolutionary Methods for Design Optimisation and control with application to industrial problems, (pp. 95-100).

[40] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 6(2), 182-197.

[41] Paquete, L. and Stutzle, T., (2003), A two-phase local search for bi-objective traveling salesman problem, Evolutionary Multi-Criterion Optimization, no. 2632, (pp. 479-493).

[42] Qingfu, Zhang and Hui, Li, (2007). A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation, (vol. 11, no. 6), (pp. 712–731).

[43] Bessai, K. Gestion optimale de l'allocation des ressources pour l'exécution des processus dans le cadre du Cloud (thèse de Doctorat, Université Paris1 Panthéon-Sorbonne), 2014.

[44] Maheswaran, M., Ali, S., Siegal, H. J., Hensgen, D., and Freund, R. F. (1999). Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In Heterogeneous Computing Workshop, (HCW'99) Proceedings. Eighth (pp. 30-44).

[45] Park, G. L., Shirazi, B., and Marquis, J. (1997, April). DFRN: A new approach for duplication based scheduling for distributed memory multiprocessor systems. In Parallel Processing Symposium, Proceedings., 11th International (pp. 157-166).

[46] Yassa, S., Chelouah, R., Kadima, H., and Granado, B. (2013, July). A Genetic Algorithm Approach to a Cloud Workflow Scheduling Problem with Multi-QoS Requirements. In 26th European Conference on Operational Research Workshop.

[47] Yassa, S., Sublime, J., Chelouah, R., Kadima, H., Jo, G. S., and Granado, B. (2013). A genetic algorithm for multi-objective optimisation in workflow scheduling with hard constraints. International Journal of Metaheuristics, 2(4), (pp.415-433).

[48] Elbernoussi,S., Lakhbab, S., Douiri, S. M., Cours des méthodes de résolution exactes heuristiques et métaheuristiques, master codes, cryptographie et sécurité de l'information, Université Mohammed V, Faculté des Sciences de Rabat.

[49] Talbi, E. G. (2009). Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

[50] Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithm. In Proceeding of the First International Conference of Genetic Algorithms and Their Application (pp. 93-100).

[51] Chr, P., and Steiglitz, K. (1982). Combinatorial Optimization: Algorithms and Complexity. Prentice Hall, Englewood Cliffs, New Jersey.

[52] Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications, 45(1), (pp. 41-51).

[53] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers & operations research, 13(5), (pp.533-549).

[54] Feo, T. A., and Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. Operations research letters, 8(2), (pp.67-71).

[55] Feo, T. A., and Resende, M. G. (1995). Greedy randomized adaptive search procedures. Journal of global optimization, 6(2), (pp.109-133).

[56] Boussaid, I. (2013). Perfectionnement de métaheuristiques pour l'optimisation continue (Doctoral dissertation, Université Paris-Est).

[57] Kammarti, R. (2006). Approches évolutionnistes pour la résolution du 1-PDPTW statique et dynamique (Doctoral dissertation, Ecole Centrale de Lille).

[58] Liu, J., Luo, X. G., Zhang, X. M., Zhang, F., and Li, B. N. (2013). Job scheduling model for cloud computing based on multi-objective genetic algorithm. IJCSI International Journal of Computer Science Issues, 10(1), (pp.134-139).

[59] Wu, Z., Liu, X., Ni, Z., Yuan, D., and Yang, Y. (2013). A market-oriented hierarchical scheduling strategy in cloud workflow systems. The Journal of Supercomputing, (pp.1-38).

[60] Kessaci, Y., Melab, N., and Talbi, E. G. (2013, June). A pareto-based genetic algorithm for optimized assignment of vm requests on a cloud brokering environment. In Evolutionary Computation (CEC), IEEE Congress on (pp. 2496-2503).

[61] Xu, J., and Fortes, J. A. (2010, December). Multi-objective virtual machine placement in virtualized data center environments. In Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing (pp. 179-188). IEEE Computer Society.

[62] Murata, T. (1989). Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4), (pp.541-580).

Bibliographie

[63] Hillah, L. M., and Petrucci, L. (2010). Standardisation des réseaux de Petri: état de l'art et enjeux futurs. Génie logiciel, (vol 93), (pp.5-10).

[64] Larbi, S., and Mohamed, S. (2014). Modeling the Scheduling Problem of Identical Parallel Machines with Load Balancing by Time Petri Nets. International Journal of Intelligent Systems and Applications, 7(1), (pp.42).

[65] Houda. H, Approche mixte de modélisation par Réseaux de Petri et SMA, mémoire de magistère en informatique, Ecole Doctorale en Informatique de l'Est Pôle de CONSTANTINE ,2010.

[66] Nobile, M. S., Besozzi, D., Cazzaniga, P., and Mauri, G. (2013). The foundation of Evolutionary Petri Nets. In BioPPN@ Petri Nets (pp. 60-74).

[67] Van der Aalst, W. M. P. (1996). Petri net based scheduling. Operations-Research-Spektrum, 18(4), (pp.219-229).

[68] Kim, H. J., Lee, J. H., and Lee, T. E. (2012, October). A Petri net-based modeling and scheduling with a branch and bound algorithm. In Systems, Man, and Cybernetics (SMC), IEEE International Conference on (pp. 1779-1784).

[69] Huang, H., Du, H., and binatorial Optimization (pp.1667-1711). Springer New York.

[70] Wang, Q., & Wang, Z. (2012). Hybrid heuristic search based on Petri net for FMS scheduling. Energy Procedia, 17, (pp.506-512).

[71] FAREH, M. E. K. Une approche basée agents pour l'allocation des ressources dans le Cloud Computing, mémoire de Magistère, Université Mohamed Khider-Biskra, 2015.

[72] Blickle, T., and Thiele, L. (1995, July). A Mathematical Analysis of Tournament Selection. In ICGA (pp. 9-16).

[73] Weber, M., & Kindler, E. (2003). The Petri Net Markup Language. Petri Net Technology for Communication Based Systems, LNCS Vol. 2472.

[74] http://projects.laas.fr/tina/home.php

[75] Olteanu, A., and Marin, A. (2011). Generation and evaluation of scheduling DAGs: How to provide similar evaluation conditions. Computer Science Master Research, 1(1), (pp.57-66).

[76] http://aws.amazon.com/ec2/, 2012

Bibliographie

[77] http://aws.amazon.com/cloudfront/

[78] V Jean. (2016) www.monnano.weebly.com/upkoad/1/6/6/3/1663287/precrit.pdf.

[79] Massey, A., and Miller, S. J. (2006). Tests of hypotheses using statistics. Mathematics Department, Brown University.

[80] www.statstutor.ac.uk/resources/uploaded/paired-t-est.pdf.

[81] Cybok, D. (2008). Workflow Management for Grid Computing: A Grid Workflow Infrastructure, msg systems. Munich, Germany.

[82] Mejía, G., Montoya, C., Cardona, J., and Castro, A. L. (2012). Petri nets and genetic algorithms for complex manufacturing systems scheduling. International Journal of Production Research, 50(3), (pp.791-803).

[83] Mušič, G. (2012). Schedule optimization based on coloured petri nets and local search. IFAC Proceedings Volumes, 45(2), (pp.352-357).

[84] Caballero-Villalobos, J. P., Mejía-Delgadillo, G. E., and García-Cáceres, R. G. (2013). Scheduling of complex manufacturing systems with Petri nets and genetic algorithms: a case on plastic injection moulds. The International Journal of Advanced Manufacturing Technology, 69(9-12), (pp.2773-2786).

[85] Kessaci, Y., Ordonnancement multi-critère sur Clouds, thèse de Doctorat, ENS Lyon, 2013.

[86] Ghoul, R. H., Benjelloul, A., Kechida, S., and Tebbikh, H. (2007). A scheduling algorithm based on petri nets and simulated annealing. American Journal of Applied Sciences, 4(5), (pp.269-273).

[87] Fareh, M. E. K., Kazar, O., Femmam, M., and Bourekkache, S. (2015, November). An agent-based approach for resource allocation in the cloud computing environment. In Telecommunication Systems Services and Applications (TSSA), 9th International Conference on (pp. 1-5).

[88] Femmam, M., Kazar,O., Baarir,S., Fareh, M. E. K., A Comparative Study of Levels Scheduling Algorithms in the Cloud Computing ,2nd International Conference on Computer, Control and Communication Technologies (CCCT'15) Dec. 3-4, 2015 Antalya, Turkey, (pp. 80-86).

[89] Femmam, M., Kazar, O., Kahloul, L., Fareh, M. E. K., Labeled Evolutionary Petri Nets/Genetic Algorithm based approach for workflow scheduling in cloud computing . International Journal of Grid and Utility Computing, 9(2), (pp.157-169).

Activités scientifiques

1. Revues Internationales

- Manel Femmam, Okba Kazar, Laid Kahloul, Mohamed El-kabir Fareh, Labeled Evolutionary Petri Nets/Genetic Algorithm based approach for workflow scheduling in cloud computing. International Journal of Grid and Utility Computing, 9(2), (pp.157-169).
- Manel Femmam, Okba Kaza, Soheib Baarir, Mohamed Elkabir Fareh, A Comparative Study of Levels Scheduling Algorithms in the Cloud Computing, Int'l Journal of Computing, Communications & Instrumentation Engg. (IJCCIE), vol. 2, Issue 2, (pp. 196-200).

2. Conférences Internationales

- Manel Femmam, Okba Kazar, Soheib Baarir, Mohamed El-kabir Fareh, A Comparative Study of Levels Scheduling Algorithms in the Cloud Computing, 2nd International Conference on Computer, Control and Communication Technologies (CCCT'15), 2015, Antalya, Turkey, (pp. 80-86).

- Mohamed El-kabir Fareh, Okba Kazar, Manel Femmam, Samir Bourekkache. An agent-based approach for resource allocation in the cloud computing environment. In 9th International Conference on Telecommunication Systems Services and Applications (TSSA), 2015, IEEE, (pp. 1-5).