الجمـهورية الجزائرية الديمقراطية الشعبيـة
وزارة التعليم العالي والبحث العلمي
جـامعـة سعـيـدة د. مـولاي الطـاهـر
كلية الرياضيات و الإعلام الآلي و الاتصالات السلكية و اللاسلكية
قسم: الإعلام الآلي

Mémoire de Master en informatique

Spécialité : Intelligence Artificielle : Principes et Applications

Thème

Optimisation de Workflows dans un Environnement Cloud par Evolutionary Multitasking

▪ **Présenté par :**

BOUSOUAR Abdelkrim

KHELEF Mohamed Redha

▪ **Dirigé par :**

Dr AZAIZ Mohamed Amine

Année universitaire 2025-2026

Remerciements

Nous exprimons notre profonde gratitude à notre professeur encadrant pour l'excellence de son accompagnement, sa disponibilité constante et la pertinence de ses conseils tout au long de l'élaboration de ce mémoire. Sa rigueur scientifique, son exigence intellectuelle et ses orientations éclairées ont été des atouts déterminants dans la réalisation et l'aboutissement de ce travail.

Nous adressons également nos remerciements les plus sincères à nos familles, dont le soutien constant, la patience et les encouragements ont représenté une source de motivation inestimable tout au long de ce parcours exigeant. Leur confiance et leur appui moral ont été des piliers sur lesquels nous avons pu nous appuyer avec sérénité.

Enfin, nous tenons à remercier toutes les personnes qui, de près ou de loin, ont participé à l'enrichissement de ce mémoire, que ce soit par leurs conseils, leurs échanges constructifs ou leur soutien. Leur contribution, directe ou indirecte, a largement participé à la réalisation de ce travail.

Dédicace

Nous dédions ce mémoire à nos familles, dont le soutien, la patience et les encouragements constants ont été une source précieuse de motivation tout au long de ce parcours. Leur présence et leur confiance nous ont permis d'avancer avec sérénité et persévérance.

Nous adressons également cette dédicace à tous les enseignants qui nous ont transmis leur savoir, guidé et inspiré au cours de notre formation. Leur expertise, leur rigueur et leur bienveillance ont largement contribué à la réussite de ce travail.

ملخص

تناول هذه الأطروحة مشكلة جدولة سير عمل في بيئات الحوسبة السحابية، وتقترح منهجًا قائمًا على تعدد المهام التطوري (EMT). بعد دراسة شاملة لسير العمل وخوارزميات التطور متعددة الأهداف، تم تطوير نموذج رياضي متعدد الأهداف، يتضمن قيود الأسبقية ومواعيد التنفيذ النهائية. يعتمد المنهج المقترح على نموذج تعدد المهام التطوري المُطبق من خلال خوارزمية التطور متعددة العوامل (MFEA)، ضمن بنية برمجية معيارية. وقد تم التحقق من صحته من خلال تجارب على سير عمل حقيقي وافتراضي في بيئة سحابية غير متجانسة. تُظهر النتائج التجريبية أن منهج EMT يُحسّن الأداء العام من خلال تقليل زمن الإنجاز، وتكلفة التنفيذ، واستهلاك الطاقة. كما يُحقق تقاربًا أسرع ومتانة أكبر في بيئات الحوسبة السحابية غير المتجانسة. يُسلط التحليل النقدي الضوء على بعض القيود، بما في ذلك التعقيد الحسابي، والحساسية لضبط المعلمات، والتحديات في التعامل مع سير العمل الديناميكي. على الرغم من هذه القيود، يتفوق نموذج EMT على الطرق التقليدية والاستدلالية من خلال تمكينه الفعال لنقل المعرفة بين مسائل التحسين. وأخيرًا، تُقترح عدة توجهات مستقبلية، تشمل التحسين الفوري، والدمج مع تقنيات التعلم الآلي، والتحقق من صحة النموذج على منصات الحوسبة السحابية الحقيقية، وتطوير أدوات تصوير البيانات لتحسين فهم آليات نقل المعرفة.

Abstract

This thesis addresses the problem of scheduling workflow in cloud computing environments and proposes an approach based on Evolutionary Multitasking (EMT). After a comprehensive study of workflows and multi-objective evolutionary algorithms, a multi-objective mathematical model was developed, incorporating precedence constraints and execution deadlines.

The proposed approach is based on the Evolutionary Multitasking paradigm implemented through the Multifactorial Evolutionary Algorithm (MFEA), within a modular software architecture. It has been validated through experiments on both real and synthetic workflows in a heterogeneous cloud environment.

Résumé

Ce mémoire traite du problème de l’ordonnancement d’un workflow dans les environnements cloud et propose une approche basée sur l’Evolutionary Multitasking (EMT). Après une étude approfondie des workflows et des algorithmes évolutionnaires multi-objectifs, une modélisation mathématique multi-objectif a été développée, intégrant les contraintes de précédence ainsi que les délais d’exécution (deadlines). L’approche proposée repose sur le paradigme d’Evolutionary Multitasking implémenté à travers le Multifactorial Evolutionary Algorithm (MFEA), dans une architecture logicielle modulaire. Elle a été validée à travers des expérimentations sur des workflows réels et synthétiques dans un environnement cloud hétérogène.

Table des matières

Liste des Figures

Liste des Tableaux

Introduction Générale

Avec l'essor rapide des technologies de l'information et de la communication, le cloud computing s'est imposé comme un paradigme incontournable pour l'exécution d'applications complexes et à grande échelle. En offrant un accès à la demande à des ressources de calcul, de stockage et de réseau, le cloud permet aux utilisateurs de déployer des applications sans se soucier des contraintes matérielles sous-jacentes. Ce modèle a profondément transformé la manière dont les systèmes informatiques sont conçus, déployés et exploités.

Parmi les applications exécutées dans le cloud, les applications basées sur des workflows occupent une place centrale. Un workflow peut être défini comme un ensemble de tâches interdépendantes, organisées selon des relations de précédence, où l'exécution d'une tâche dépend des résultats produits par d'autres. Ces workflows sont largement utilisés dans de nombreux domaines, tels que le calcul scientifique, la bio-informatique, l'astrophysique, l'analyse de données massives (Big Data) et les processus métiers complexes.

Cependant, l'exécution efficace d'un workflow dans un environnement cloud soulève un défi majeur : le problème de l'ordonnancement. L'ordonnancement de workflow consiste à déterminer quand et sur quelles ressources chaque tâche doit être exécutée, tout en respectant les contraintes de précédence et les limitations des ressources disponibles. Ce problème est reconnu comme étant NP-difficile, ce qui rend la recherche de solutions optimales particulièrement complexe, surtout lorsque la taille du workflow et le nombre de ressources augmentent.

Dans ce contexte, plusieurs objectifs doivent être pris en compte lors de l'ordonnancement, notamment la minimisation du temps d'exécution global (makespan), la réduction du coût monétaire lié à l'utilisation des ressources et la diminution de la consommation énergétique. Ces objectifs sont souvent contradictoires, ce qui conduit naturellement à un problème d'optimisation multi-objectif.

Les approches classiques d'ordonnancement reposent généralement sur des heuristiques ou des métaheuristiques conçues pour traiter un problème donné. Bien que ces méthodes aient montré leur efficacité dans certains contextes, elles peuvent souffrir d'une convergence lente ou

d'un piégeage dans des optima locaux, en particulier dans des environnements cloud dynamiques et complexes.

Dans ce contexte, l'Evolutionary Multitasking (EMT) apparaît comme une approche prometteuse. Inspiré par la capacité humaine à apprendre et à résoudre plusieurs tâches simultanément, l'EMT permet d'améliorer le processus de recherche en exploitant un transfert de connaissances implicite au sein d'un algorithme évolutionnaire. Le cadre le plus représentatif de cette approche est le Multifactorial Evolutionary Algorithm (MFEA), qui a montré des résultats encourageants dans divers domaines d'optimisation.

Dans ce mémoire, nous exploitons le principe de l'Evolutionary Multitasking dans le cadre de l'optimisation de l'ordonnancement d'un seul workflow. L'idée consiste à structurer le problème de manière à bénéficier du transfert de connaissances entre différentes représentations ou sous-composantes du problème, afin d'accélérer la convergence vers des solutions de haute qualité et d'améliorer les performances globales.

Problématique

La problématique principale de ce travail peut être formulée comme suit : *comment optimiser l'ordonnancement d'un workflow dans un environnement cloud hétérogène, tout en minimisant le temps d'exécution, le coût et la consommation énergétique, en exploitant les mécanismes de transfert de connaissances offerts par l'Evolutionary Multitasking ?*

Cette problématique soulève plusieurs questions secondaires :

— Comment modéliser efficacement un workflow et les ressources cloud ?
— Comment formuler le problème d'ordonnancement comme un problème d'optimisation multi-objectif ?
— Comment adapter l'EMT à un problème portant sur un seul workflow ?

Objectifs du Mémoire

L'objectif principal de ce mémoire est de proposer et d'évaluer une approche basée sur l'Evolutionary Multitasking pour l'optimisation de l'ordonnancement d'un workflow dans un environnement cloud. Plus précisément, les objectifs sont :

— Modéliser formellement le workflow et l'environnement cloud ;
— Concevoir une approche EMT adaptée à l'ordonnancement d'un workflow ;
— Implémenter l'algorithme MFEA et ses mécanismes de transfert de connaissances ;

— Comparer les performances de l'approche proposée avec des méthodes classiques ;
— Analyser les avantages et les limites de l'EMT dans ce contexte.

Organisation du Mémoire

Ce mémoire est organisé comme suit. Le premier chapitre présente les fondements du cloud computing et des workflows. Le deuxième chapitre est consacré à l'étude du problème d'ordonnancement de workflows dans le cloud et aux approches existantes. Le troisième chapitre introduit les algorithmes évolutionnaires et l'optimisation multi-objectif. Le quatrième chapitre détaille le concept d'Evolutionary Multitasking et le cadre MFEA. Le cinquième chapitre présente la modélisation du problème. Le sixième chapitre décrit l'implémentation de l'approche proposée. Le septième chapitre est dédié à l'expérimentation et à l'analyse des résultats. Enfin, une conclusion générale synthétise les contributions du travail et présente les perspectives futures.

Chapitre 1

Fondements du Cloud Computing et des Workflows

1.1 Introduction

Le cloud computing constitue aujourd'hui l'un des piliers fondamentaux des systèmes informatiques modernes. Il a profondément transformé la manière dont les ressources de calcul sont fournies, consommées et facturées [1].

Dans ce chapitre, nous présentons les bases théoriques nécessaires à la compréhension du cloud computing et des workflows, qui constituent le cadre d'application de ce mémoire. Nous retraçons l'évolution historique ayant conduit à l'émergence du cloud computing, détaillons ses concepts, caractéristiques et modèles, puis introduisons les workflows, leur modélisation ainsi que leur importance dans les environnements cloud [12].

1.2 Historique de l'Évolution vers le Cloud Computing

1.2.1 Du calcul centralisé au calcul distribué

Les premiers systèmes informatiques reposaient sur des architectures centralisées (mainframes), partagées par plusieurs utilisateurs [3]. Cette approche présentait des limites en termes d'évolutivité et de flexibilité.

Avec l'apparition des ordinateurs personnels dans les années 1980, le paradigme du calcul distribué a progressivement émergé [4]. Les ressources informatiques ont alors été réparties

sur plusieurs machines interconnectées via des réseaux locaux et étendus.

1.2.2 Grilles de calcul et virtualisation

Dans les années 1990 et 2000, le *grid computing* a permis de mutualiser des ressources distribuées afin d'exécuter des applications intensives [5].

Par ailleurs, la virtualisation a constitué une avancée majeure, permettant l'abstraction du matériel physique et la création de machines virtuelles indépendantes, facilitant ainsi la gestion et l'allocation dynamique des ressources [6].

1.2.3 Émergence du cloud computing

Le cloud computing est né de la convergence entre la virtualisation, les réseaux à haut débit et les modèles économiques basés sur la facturation à l'usage. Des fournisseurs tels qu'Amazon Web Services, Google et Microsoft ont largement contribué à la popularisation de ce modèle [7].

1.3 Cloud

1.3.1 Définition

Le **cloud** (ou *informatique en nuage*) désigne l'ensemble des ressources informatiques (serveurs, stockage, bases de données, réseaux et logiciels) hébergées à distance dans des centres de données et accessibles via Internet, plutôt que localement sur un ordinateur personnel ou au sein d'une organisation [1]. Il permet aux utilisateurs d'accéder à leurs données et applications depuis n'importe quel appareil connecté à Internet.

Selon une autre approche, le cloud peut être considéré comme une infrastructure distante permettant de stocker des données d'exécuter des applications sur des serveurs situés dans des centres de données, accessibles à travers Internet.

Enfin, selon le NIST, le cloud est un modèle permettant un accès réseau ubiquitaire, pratique et à la demande à un ensemble partagé de ressources informatiques configurables, pouvant être rapidement provisionnées et libérées avec un effort de gestion minimal [2].

1.3.2 Caractéristiques essentielles

Le cloud repose sur plusieurs caractéristiques fondamentales :
— **Accès à la demande** : provisionnement automatique des ressources ;
— **Élasticité** : adaptation dynamique des ressources selon les besoins ;
— **Mutualisation** : partage des ressources entre plusieurs utilisateurs ;
— **Facturation à l'usage** : coûts proportionnels à la consommation réelle.

1.3.3 Modèles de services

— **Infrastructure as a Service (IaaS)** : fourniture de ressources matérielles virtualisées ;
— **Platform as a Service (PaaS)** : fourniture de plateformes de développement ;
— **Software as a Service (SaaS)** : fourniture d'applications accessibles via Internet.

TABLE 1.1 – Comparaison des modèles de services cloud

Modèle	Contrôle	Flexibilité	Exemples
IaaS	Élevé	Élevée	EC2, Azure VM
PaaS	Moyen	Moyenne	Google App Engine
SaaS	Faible	Faible	Gmail, Salesforce

1.3.4 Virtualisation

La virtualisation permet d'exécuter plusieurs machines virtuelles sur un même serveur physique [8]. Chaque machine virtuelle dispose de ressources isolées, telles que le CPU, la mémoire et le stockage.

1.3.5 Machines virtuelles et hyperviseurs

Un hyperviseur est un logiciel permettant de gérer les machines virtuelles. On distingue :
— les hyperviseurs de type 1 (*bare-metal*) ;
— les hyperviseurs de type 2 (*hébergés*).

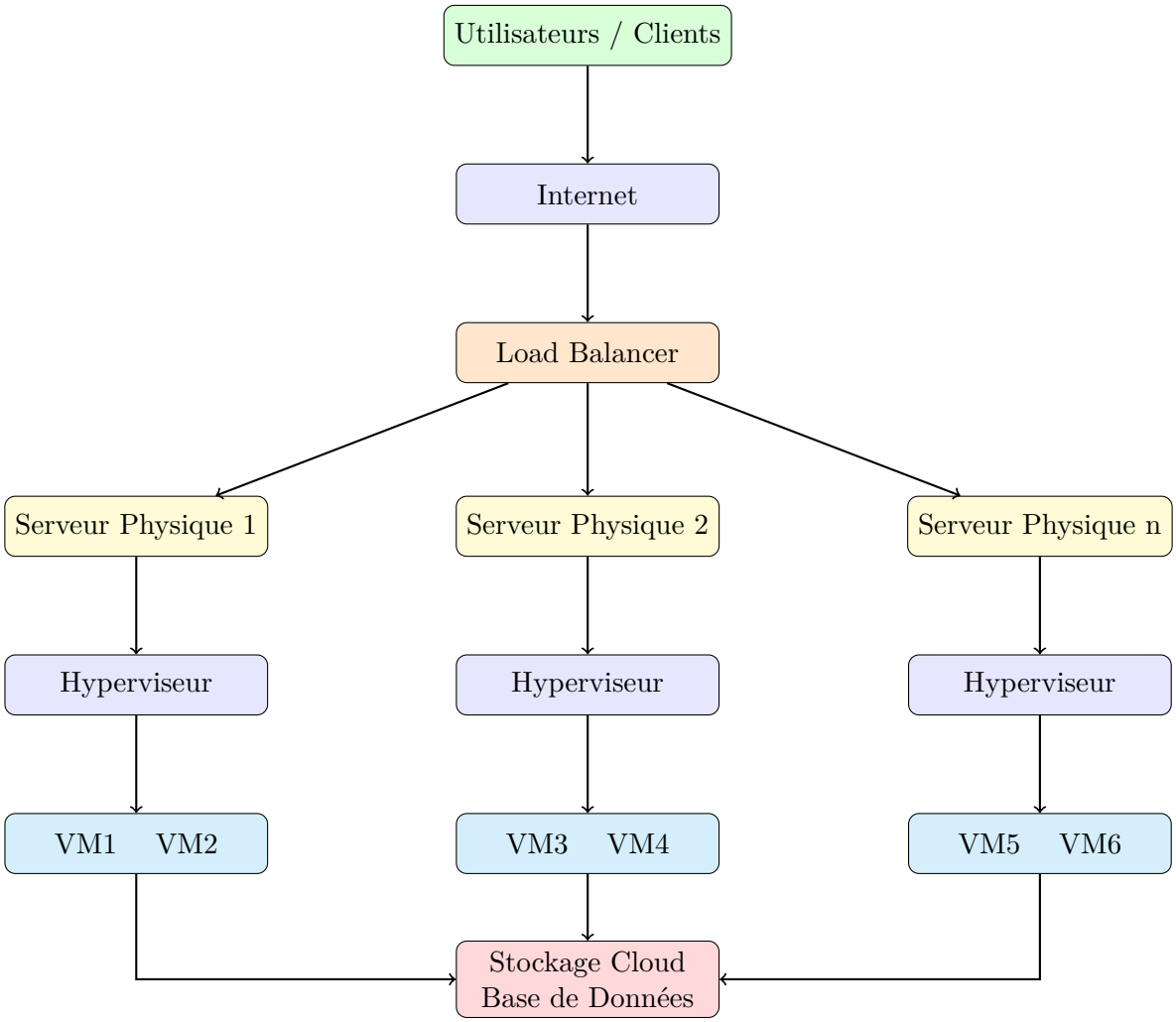


FIGURE 1.1 – Architecture générale du cloud computing

1.4 Workflows

1.4.1 Définition d'un workflow

Définition d'un workflow (résumé)

Un workflow est un ensemble de tâches interconnectées exécutées selon des dépendances de précédence [11].

Une tâche ne peut s'exécuter que lorsque toutes ses tâches prédécesseurs sont terminées. Les tâches peuvent être indépendantes ou dépendantes et avoir des durées d'exécution différentes, ce qui permet de modéliser des applications complexes.

Les workflows sont utilisés dans plusieurs domaines tels que le calcul scientifique, le cloud computing et le traitement de données massives. L'objectif principal de leur ordonnancement est de minimiser le makespan tout en respectant les contraintes de dépendance.

1.4.2 Représentation par graphes DAG

Les workflows sont généralement modélisés par des graphes orientés acycliques (DAG) :

— les nœuds représentent les tâches ;

— les arêtes représentent les dépendances.

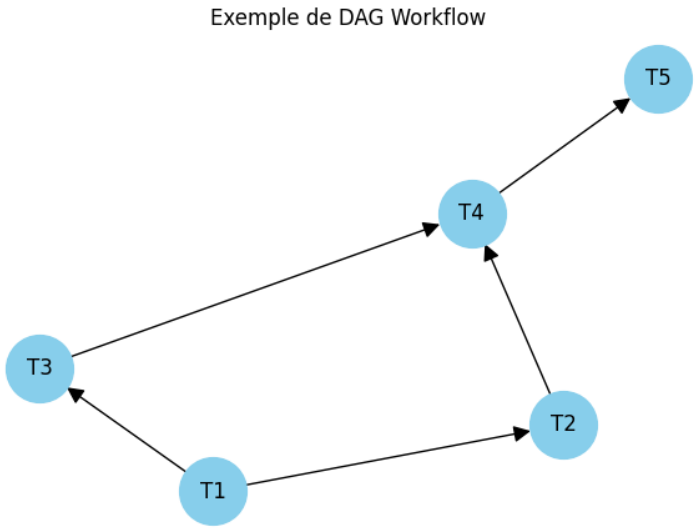


FIGURE 1.2 – Exemple de workflow représenté sous forme de DAG [10]

1.4.3 Types de workflows

On distingue principalement :

— **workflows scientifiques** : simulations et analyses de données ;

— **workflows métiers** : processus d'entreprise ;

— **workflows Big Data** : traitement massif de données.

TABLE 1.2 – Caractéristiques des principaux types de workflows

Type	Domaine	Exemple
Scientifique	Recherche	Montage génomique
Métier	Entreprise	BPM
Big Data	Données massives	MapReduce

1.4.4 Importance des workflows dans le cloud

Le cloud offre un environnement particulièrement adapté à l'exécution des workflows grâce à son élasticité et à sa capacité de mise à l'échelle. Cependant, cette flexibilité introduit des défis importants en matière d'ordonnancement [12].

1.5 Relation entre Cloud Computing et Workflows

L'intégration des workflows dans le cloud permet d'améliorer les performances des applications complexes. Néanmoins, une gestion inefficace des ressources peut entraîner des coûts élevés ainsi que des temps d'exécution importants [14].

1.6 Conclusion

Ce chapitre a présenté les fondements du cloud computing et des workflows ainsi que leur évolution historique. Ces notions constituent la base conceptuelle pour l'étude de l'ordonnancement des workflows dans le cloud, qui sera abordée dans le chapitre suivant.

Chapitre 2

Ordonnancement de Workflows dans le Cloud Computing

2.1 Introduction

L'ordonnancement des workflows constitue l'un des problèmes fondamentaux dans les environnements de cloud computing. Il s'agit de déterminer comment affecter et planifier les tâches d'un workflow sur des ressources distribuées, tout en respectant les contraintes de précédence et les objectifs d'optimisation.

Ce chapitre présente les concepts fondamentaux liés à l'ordonnancement de workflows dans le cloud. Nous abordons la problématique générale, les critères d'optimisation, les différentes catégories d'approches, ainsi que leurs avantages et leurs limites.

2.2 Problématique de l'Ordonnancement

2.2.1 Définition de l'ordonnancement

L'ordonnancement (ou *scheduling*) désigne le processus consistant à déterminer l'ordre d'exécution des tâches et leur affectation aux ressources disponibles. Dans le cas des workflows, l'ordonnancement doit respecter les dépendances entre tâches, ce qui rend le problème NP-difficile [10].

2.2.2 Contraintes de précédence

Les workflows sont généralement représentés sous forme de graphes orientés acycliques (DAG), où une tâche ne peut débuter qu'après l'exécution de toutes ses tâches précédentes [16].

2.2.3 Complexité du problème

La complexité du problème augmente avec :
— le nombre de tâches,
— le nombre et l'hétérogénéité des ressources,
— le nombre d'objectifs à optimiser [17].

2.3 Environnement Cloud et Ressources

2.3.1 Hétérogénéité des ressources

Les environnements cloud offrent des machines virtuelles (VMs) variées en termes de capacité de calcul, de performance et de coût [1].

2.3.2 Provisionnement dynamique

Le cloud permet le provisionnement et la libération dynamiques des ressources, ce qui offre une grande flexibilité mais introduit également des défis pour l'ordonnancement [13].

2.3.3 Impact de la virtualisation

La virtualisation peut engendrer un léger surcoût en performance, mais elle permet une meilleure isolation et une gestion plus flexible des ressources [6].

2.4 Critères d'Optimisation

2.4.1 Temps d'exécution total (Makespan)

Le *makespan* correspond au temps nécessaire pour terminer l'exécution du workflow :

$$Makespan = \max_{i \in T}(FT_i)$$

où FT_i est le temps de fin de la tâche i [10].

2.4.2 Coût monétaire

Dans le cloud, les ressources sont facturées à l'usage :

$$Cost = \sum_{j=1}^{m} C_{vm_j} \times T_{vm_j}$$

où C_{vm_j} est le coût de la VM j et T_{vm_j} son temps d'utilisation [13].

2.4.3 Consommation énergétique

L'optimisation énergétique devient un critère important dans les environnements cloud durables :

$$Energy = \sum_{i=1}^{n} P_i \times t_i$$

où P_i est la puissance durant l'intervalle i et t_i sa durée d'utilisation [15].

2.4.4 Respect des délais (Deadlines)

Certains workflows imposent des contraintes temporelles strictes. Leur non-respect peut entraîner des pénalités ou une dégradation de la qualité de service [9].

TABLE 2.1 – Principaux critères d'optimisation

Critère	Description
Makespan	Temps total d'exécution
Coût	Coût d'utilisation des ressources
Énergie	Consommation énergétique
Deadline	Contraintes temporelles

2.5 Classification des Approches d'Ordonnancement

2.5.1 Ordonnancement statique

L'ordonnancement statique est réalisé avant l'exécution et suppose une connaissance complète des tâches et des ressources [9].

L'ordonnancement statique signifie qu'on planifie toutes les tâches à l'avance, avant même que l'exécution ne commence.

Idée principale

On construit un planning complet avant l'exécution du système, en décidant :
— quelle tâche sera exécutée,
— sur quelle ressource (machine, VM, processeur),
— et à quel moment.

Hypothèses importantes

Ce type d'ordonnancement repose sur une connaissance complète et précise :
— des tâches : durée d'exécution, dépendances, charge de travail, etc.
— des ressources : nombre de machines, capacités, performances, coûts, etc.

Conséquence

Comme tout est connu à l'avance :
— le plan est figé avant exécution,
— il ne change pas pendant le traitement (sauf cas exceptionnels),
— il est souvent calculé avec des méthodes d'optimisation ou heuristiques.

Avantages

- Permet une optimisation globale (temps, coût, énergie).
- Bon pour les systèmes prévisibles (batch processing, calcul scientifique).
- Planification plus facile à analyser théoriquement.

Inconvénients

- Peu flexible face aux imprévus :
  - nouvelles tâches,
  - pannes de machines,
  - variations de charge.
- Peut devenir inefficace si les conditions changent.

2.5.2 Ordonnancement dynamique

L'ordonnancement dynamique prend des décisions en temps réel en fonction de l'état du système [11]. Contrairement à l'ordonnancement statique, les décisions ne sont pas fixées à l'avance, mais sont recalculées pendant l'exécution.

Principe À chaque instant, le système observe son état actuel (charge des machines, tâches en attente, disponibilité des ressources) afin de décider quelle tâche exécuter ensuite et sur quelle ressource.

Caractéristiques

- Adaptatif : s'ajuste aux changements du système.
- Réactif : répond aux événements en temps réel (arrivée de tâches, pannes, surcharge).
- Non déterministe à l'avance : le planning complet n'est pas connu initialement.

Avantages

- Bonne flexibilité face aux imprévus.
- Meilleure utilisation des ressources dans des environnements variables.
- Adapté aux systèmes distribués et au cloud computing.

Inconvénients

— Décisions locales pouvant être sous-optimales globalement.
— Surcoût de calcul dû aux décisions en temps réel.
— Plus difficile à analyser théoriquement.

2.5.3 Ordonnancement centralisé et distribué

Ordonnancement centralisé Dans un ordonnancement centralisé, un unique planificateur (scheduler) prend toutes les décisions de planification. Il est responsable de l'affectation des tâches aux ressources, ainsi que de la gestion globale de l'exécution [11].

Ce modèle permet d'avoir une vue globale du système, ce qui facilite souvent l'optimisation, mais peut devenir un point de congestion ou de défaillance unique.

Ordonnancement distribué Dans un ordonnancement distribué, plusieurs entités (nœuds ou planificateurs) coopèrent pour prendre les décisions de planification des tâches [11]. Chaque entité possède une vision partielle du système et communique avec les autres pour coordonner les affectations.

Ce modèle améliore la scalabilité et la tolérance aux pannes, mais rend la coordination plus complexe et peut entraîner des décisions moins optimales globalement.

Comparaison
— Centralisé : décision unique, vision globale, mais risque de goulot d'étranglement.
— Distribué : plusieurs décideurs, meilleure scalabilité, mais coordination complexe [11].

2.6 Heuristiques Classiques

2.6.1 Algorithme HEFT

HEFT (Heterogeneous Earliest Finish Time) est une heuristique largement utilisée pour l'ordonnancement dans des environnements hétérogènes [10].

Principe L'algorithme HEFT vise à minimiser le *makespan* en affectant chaque tâche au processeur (ou à la machine) qui permet son exécution la plus rapide en tenant compte de

l'hétérogénéité des ressources.

Fonctionnement HEFT repose principalement sur deux étapes :

- **Phase de priorisation :** les tâches sont classées selon leur priorité, généralement basée sur la longueur du chemin critique du graphe de tâches.
- **Phase d'affectation :** chaque tâche est assignée à la ressource qui minimise son temps de fin le plus tôt (*Earliest Finish Time*).

Avantages

- Simple et efficace pour les systèmes hétérogènes.
- Produit de bonnes solutions proches de l'optimal.
- Réduit le *makespan*.

Limites

- Heuristique : ne garantit pas une solution optimale.
- Sensible à la précision des estimations de temps d'exécution [10].

2.6.2 Min-Min et Max-Min

Les algorithmes Min-Min et Max-Min sont des heuristiques d'ordonnancement utilisées pour attribuer les tâches en fonction des temps d'exécution estimés sur les différentes ressources [17].

Principe général Ces deux algorithmes reposent sur une matrice des temps d'exécution, où chaque tâche est évaluée chaque machine afin de déterminer les durées possibles d'exécution.

Algorithme Min-Min L'algorithme Min-Min fonctionne comme suit :

- Pour chaque tâche non affectée, on calcule son temps d'exécution minimum sur toutes les machines.
- On sélectionne la tâche ayant le plus petit temps d'exécution minimal.
- Cette tâche est ensuite assignée à la machine correspondante.
- Le processus est répété jusqu'à ce que toutes les tâches soient affectées.

Algorithme Max-Min L'algorithme Max-Min est similaire, mais diffère dans la sélection :

— Pour chaque tâche non affectée, on calcule son temps d'exécution minimum sur toutes les machines.
— On sélectionne la tâche ayant le plus grand temps d'exécution minimal.
— Cette tâche est assignée à la machine qui minimise son temps de fin.
— Le processus est répété jusqu'affectation complète.

Comparaison

— Min-Min favorise les petites tâches et améliore souvent le temps global moyen.
— Max-Min favorise les tâches longues pour éviter qu'elles soient retardées.

Limites

— Sensibles à la distribution des tailles de tâches.
— Peu efficaces dans certains scénarios dynamiques [17].

2.6.3 Algorithme Max-Max

L'algorithme Max-Max est une heuristique d'ordonnancement qui, comme Min-Min et Max-Min, utilise les temps d'exécution estimés des tâches sur les différentes machines pour effectuer l'affectation [17].

Principe Max-Max consiste à sélectionner, à chaque itération, la tâche ayant le plus grand temps d'exécution minimal, puis à l'affecter à la machine qui permet de minimiser son temps de fin.

Fonctionnement

— Pour chaque tâche non encore assignée, on calcule le temps d'exécution minimal sur l'ensemble des machines.
— On sélectionne la tâche ayant la valeur maximale parmi ces minima (la tâche la plus « coûteuse » en premier).
— Cette tâche est ensuite affectée à la machine qui minimise son temps de complétion.
— Le processus est répété jusqu'à ce que toutes les tâches soient affectées.

Intuition L'idée principale est de traiter en priorité les tâches les plus longues afin de réduire les risques de retard et d'améliorer l'équilibrage de la charge.

Avantages

— Réduit le risque de retard des tâches longues.
— Améliore l'équilibrage dans certains cas hétérogènes.

Limites

— Peut être moins efficace pour les petits ensembles de tâches.
— Ne garantit pas une solution optimale.
— Sensible à la distribution des temps d'exécution [17].

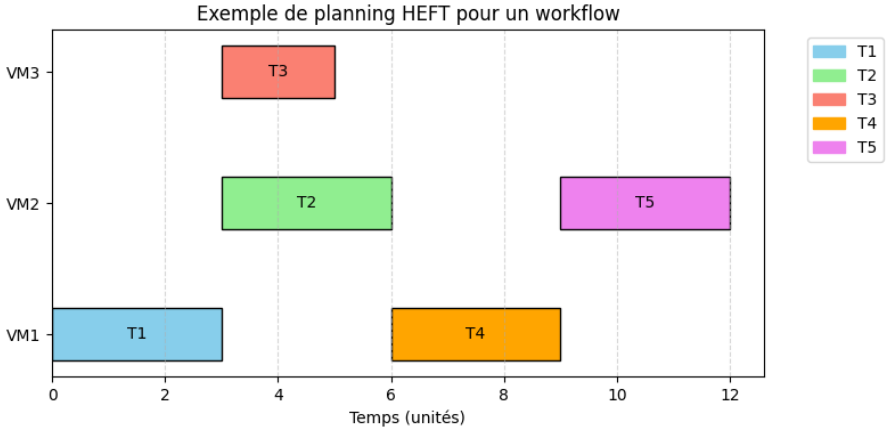


FIGURE 2.1 – Principe de l'algorithme HEFT [10]

2.6.4 Limites des heuristiques

Les approches classiques présentent plusieurs limitations :
— elles sont souvent mono-objectifs ;
— elles s'adaptent difficilement aux environnements dynamiques ;
— elles peuvent être piégées dans des optima locaux [12].

2.7 Métaheuristiques pour l'Ordonnancement

2.7.1 Algorithmes génétiques (GA)

Les algorithmes génétiques sont des métaheuristiques inspirées de la théorie de l'évolution de Darwin. Ils manipulent une population de solutions qui évolue au fil des générations grâce à des opérateurs de sélection, de croisement et de mutation. Les solutions les mieux adaptées ont une plus grande probabilité d'être conservées et de transmettre leurs caractéristiques aux

générations suivantes. Cette approche favorise à la fois l'exploration de nouvelles régions de l'espace de recherche et l'exploitation des solutions prometteuses, ce qui en fait une technique particulièrement adaptée aux problèmes d'optimisation combinatoire de grande dimension [18].

2.7.2 Algorithmes à colonies de fourmis (ACO)

Les algorithmes à colonies de fourmis sont des métaheuristiques inspirées du comportement des fourmis réelles lorsqu'elles recherchent le chemin le plus court entre leur nid et une source de nourriture. En utilisant des traces de phéromones virtuelles pour guider la recherche, ils favorisent progressivement les solutions les plus performantes. Cette approche distribuée et adaptative est particulièrement efficace pour résoudre des problèmes d'optimisation combinatoire de grande taille [19].

2.7.3 Optimisation par essaims particulaires (PSO)

Les algorithmes PSO sont des métaheuristiques inspirées du comportement collectif des oiseaux et des poissons. Ils utilisent un ensemble de particules qui explorent l'espace de recherche en partageant des informations sur les meilleures solutions trouvées. Cette coopération permet d'équilibrer l'exploration et l'exploitation de l'espace des solutions, conduisant souvent à des résultats proches de l'optimum pour des problèmes d'optimisation complexes [20].

Table 2.2 – Comparaison des métaheuristiques

Méthode	Exploration	Convergence	Adaptabilité
GA	Élevée	Moyenne	Bonne
ACO	Moyenne	Élevée	Moyenne
PSO	Élevée	Rapide	Bonne

2.8 Positionnement de l'Evolutionary Multitasking

L'Evolutionary Multitasking (EMT) constitue une approche récente permettant d'améliorer les performances des algorithmes évolutionnaires grâce au transfert de connaissances. Dans le contexte de ce mémoire, cette approche sera adaptée à l'optimisation de l'ordonnancement d'un workflow, afin d'améliorer la convergence et la qualité des solutions [30].

2.9 Conclusion

Ce chapitre a présenté une vue d'ensemble de l'ordonnancement des workflows dans le cloud. Les critères d'optimisation, les approches existantes ainsi que leurs limites justifient l'utilisation de méthodes avancées telles que l'Evolutionary Multitasking pour améliorer les performances dans des environnements complexes.

Chapitre 3

Algorithmes évolutionnaires et Optimisation Multi-Objectif

3.1 Introduction

Les algorithmes évolutionnaires (AE) constituent une famille de méthodes d'optimisation stochastiques inspirées des mécanismes de l'évolution naturelle [18]. Leur capacité à explorer efficacement des espaces de recherche complexes et de grande dimension en fait des outils particulièrement adaptés à l'optimisation de problèmes NP-difficiles, tels que l'ordonnancement de workflows dans le cloud [11].

Dans ce chapitre, nous présentons les principes fondamentaux des algorithmes évolutionnaires, leurs principaux opérateurs ainsi que les concepts essentiels de l'optimisation multi-objectif. Nous abordons également les principaux algorithmes multi-objectifs de la littérature, notamment NSGA-II, SPEA2 et MOEA/D [25, 27, 23].

3.2 Algorithmes Évolutionnaires

3.2.1 Principes généraux

Un algorithme évolutionnaire repose sur l'évolution d'une population de solutions candidates. Chaque solution, appelée individu, est évaluée à l'aide d'une fonction de fitness [24]. L'évolution s'effectue à travers des opérateurs inspirés de la sélection naturelle.

Les étapes générales d'un AE sont :

1. Initialisation de la population ;
2. Évaluation des individus ;
3. Sélection des parents ;
4. Application des opérateurs génétiques (croisement, mutation) ;
5. Remplacement de la population.

3.2.2 Représentation des individus

Le choix de la représentation est crucial pour les performances de l'algorithme. Les représentations les plus utilisées sont :

— codage binaire [18] ;
— codage réel [24] ;
— codage permutationnel, adapté aux problèmes d'ordonnancement [17].

Dans le cadre de l'ordonnancement de workflows, la représentation permutationnelle est généralement privilégiée.

3.3 Algorithmes Génétiques

3.3.1 Principe de base

Les algorithmes génétiques (AG) sont les AE les plus utilisés. Ils reposent sur l'évolution d'une population de chromosomes selon des opérateurs de sélection, croisement et mutation [18].

3.3.2 Sélection

Les principales méthodes de sélection sont :

— sélection par roulette [18] ;
— sélection par tournoi [25] ;
— sélection élitiste [25].

3.3.3 Croisement et mutation

Le croisement combine les informations de deux parents pour générer de nouvelles solutions [18]. La mutation introduit des variations aléatoires afin de maintenir la diversité et

éviter la convergence prématurée [18].

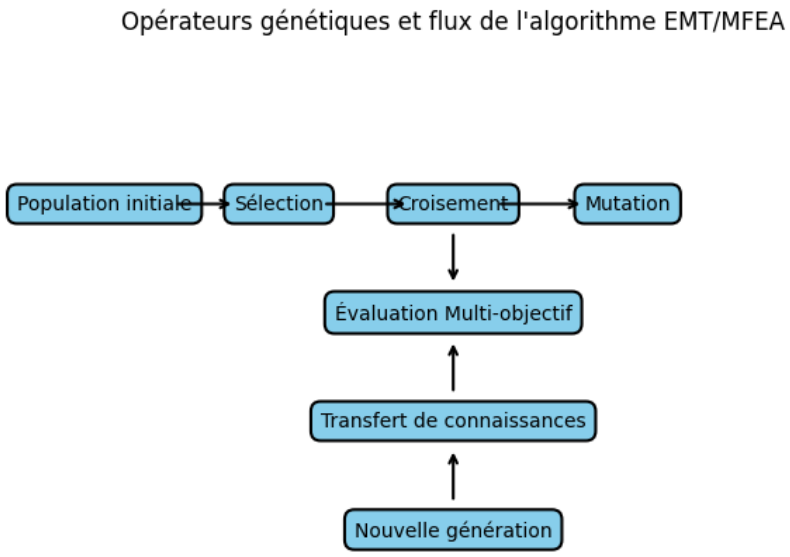


FIGURE 3.1 – Opérateurs génétiques d'un algorithme génétique

3.4 Autres Algorithmes Évolutionnaires

3.4.1 Stratégies d'évolution

Les stratégies d'évolution (ES) utilisent des vecteurs réels et adaptent dynamiquement les paramètres de mutation pour améliorer la convergence [26].

3.4.2 Programmation génétique

La programmation génétique (GP) permet de faire évoluer des programmes ou expressions mathématiques sous forme d'arbres [24].

3.4.3 Comparaison

Le Tableau 3.1 compare les principaux algorithmes évolutionnaires.

Les algorithmes génétiques sont flexibles et adaptés aux problèmes combinatoires. Les stratégies d'évolution sont efficaces pour les espaces continus. La programmation génétique permet la génération automatique de structures complexes mais reste coûteuse en calcul.

Aucun algorithme n'est universellement supérieur ; leur performance dépend du problème traité.

TABLE 3.1 – Comparaison des algorithmes évolutionnaires

Algorithme	Représentation	Convergence	Applications
GA	Binaire / Permutation	Moyenne	Ordonnancement
ES	Réelle	Rapide	Optimisation continue
GP	Arbre	Lente	Modélisation

3.5 Optimisation Multi-Objectif

3.5.1 Définition

Un problème d'optimisation multi-objectif consiste à optimiser simultanément plusieurs fonctions objectif souvent conflictuelles [25].

Cas de minimisation

$$\min \mathbf{F}(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$

Cas de maximisation

$$\max \mathbf{F}(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$

3.5.2 Dominance de Pareto

Pour un problème de minimisation Une solution x_1 domine une solution x_2 si :

$$\forall i,\ f_i(x_1) \leq f_i(x_2) \quad \text{et} \quad \exists j,\ f_j(x_1) < f_j(x_2)$$

Pour un problème de maximisation Une solution x_1 domine une solution x_2 si :

$$\forall i,\ f_i(x_1) \geq f_i(x_2) \quad \text{et} \quad \exists j,\ f_j(x_1) > f_j(x_2)$$

3.5.3 Front de Pareto

L'ensemble des solutions non dominées constitue le front de Pareto [25].

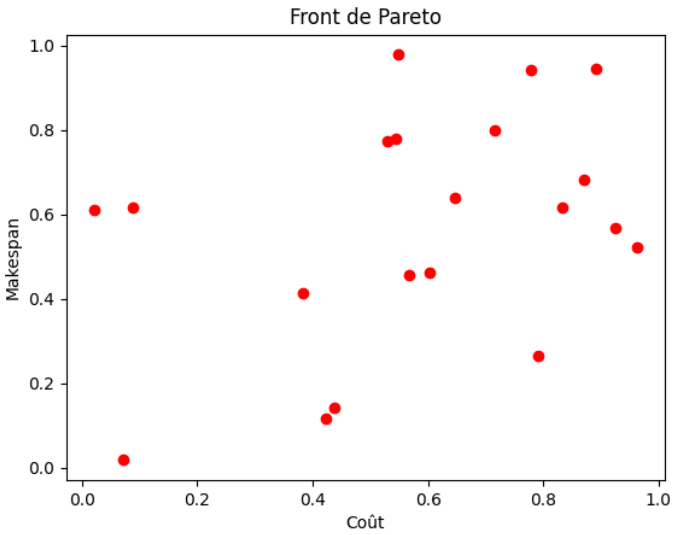


FIGURE 3.2 – Illustration d'un front de Pareto

3.6 Concepts des Métaheuristiques Multi-Objectifs

Les algorithmes multi-objectifs reposent sur :

— la dominance de Pareto ;
— le maintien de la diversité ;
— l'élitisme ;
— l'équilibre exploration/exploitation ;
— l'évaluation par indicateurs (hypervolume, GD, spacing) [28].

3.7 Algorithmes Multi-Objectifs

3.7.1 NSGA-II

NSGA-II est une métaheuristique d'optimisation multiobjectif basée sur un tri non dominé des solutions et sur la distance de foule pour maintenir la diversité de la population. Cette combinaison permet d'identifier efficacement un ensemble de solutions non dominées

représentant différents compromis entre les objectifs considérés. L'algorithme est reconnu pour sa capacité à converger vers le front de Pareto tout en préservant une bonne répartition des solutions [25].

3.7.2 SPEA2

SPEA2 est une métaheuristique d'optimisation multiobjectif basée sur la dominance de Pareto. Il maintient une archive externe des solutions non dominées afin de préserver les meilleurs compromis trouvés au cours de la recherche. Son mécanisme de fitness combine des informations de dominance et de densité, ce qui améliore à la fois la convergence vers le front de Pareto et la diversité des solutions obtenues [27].

3.7.3 MOEA/D

MOEA/D est une métaheuristique d'optimisation multiobjectif qui décompose un problème en plusieurs sous-problèmes mono-objectifs interconnectés. Chaque sous-problème est résolu en collaboration avec ses voisins, ce qui favorise à la fois la convergence vers le front de Pareto et la diversité des solutions. Cette approche permet de traiter efficacement les problèmes comportant un grand nombre d'objectifs [23].

3.8 Application à l'ordonnancement de workflows

Les algorithmes multi-objectifs sont adaptés à l'ordonnancement de workflows dans le cloud car ils permettent d'optimiser simultanément le makespan, le coût et l'énergie [14].

3.9 Limites des approches classiques

Malgré leur efficacité, ces méthodes présentent :

— convergence parfois lente ;
— absence de transfert de connaissances ;
— difficulté dans les environnements très dynamiques.

Ces limites motivent l'utilisation de l'Evolutionary Multitasking (EMT), qui sera présenté dans le chapitre suivant [30].

3.10 Conclusion

Ce chapitre a présenté les fondements des algorithmes évolutionnaires et de l'optimisation multi-objectif. Ces bases théoriques constituent le socle nécessaire pour comprendre les approches avancées basées sur l'EMT, présentées dans le chapitre suivant.

Chapitre 4

Evolutionary Multitasking pour l'Ordonnancement de Workflows

4.1 Introduction

Les chapitres précédents ont montré que l'ordonnancement d'un workflow dans le cloud computing constitue un problème d'optimisation complexe, multi-objectif et NP-difficile [10].

Bien que les algorithmes évolutionnaires multi-objectifs soient efficaces pour traiter ce type de problème, ils présentent des limites en termes de convergence et d'exploration de l'espace de recherche.

Dans ce contexte, l'Evolutionary Multitasking (EMT) constitue une approche prometteuse permettant d'améliorer les performances des algorithmes évolutionnaires grâce au transfert de connaissances entre différentes instances d'un même problème [30].

Contrairement à son usage classique visant plusieurs problèmes distincts, l'EMT est ici adapté à un seul problème d'ordonnancement de workflow. Le transfert de connaissances est exploité entre différentes configurations du problème afin d'améliorer l'exploration de l'espace de recherche et la convergence.

Nous utilisons ainsi le paradigme de l'EMT pour accélérer la convergence de l'optimisation multi-objectif d'un workflow unique, en exploitant le transfert de connaissances entre différentes configurations du problème.

Ainsi, l'EMT constitue le moteur de recherche permettant d'explorer efficacement l'espace de solutions, tandis que l'optimisation multi-objectif représente le critère de sélection

basé sur l'évaluation des solutions selon les objectifs (makespan, coût et énergie).

Ce chapitre présente les fondements de l'EMT, le cadre du Multifactorial Evolutionary Algorithm (MFEA), ainsi que son adaptation au problème étudié [21].

4.2 Fondements de l'Evolutionary Multitasking

4.2.1 Définition de l'EMT

L'Evolutionary Multitasking est un paradigme d'optimisation basé sur l'idée de résoudre simultanément plusieurs tâches en partageant des connaissances entre elles afin d'améliorer les performances globales [30].

Dans ce travail, les tâches ne correspondent pas à plusieurs workflows différents, mais à différentes instances générées à partir du même problème d'ordonnancement.

4.2.2 Motivation

L'EMT est inspiré de la capacité d'apprentissage transférable observée dans les systèmes naturels. Une connaissance acquise sur une tâche peut être réutilisée pour améliorer la recherche dans une autre tâche.

Appliqué à l'optimisation, ce mécanisme permet d'améliorer l'exploration et de réduire le risque de convergence vers des optima locaux.

4.2.3 Avantages

— amélioration de la convergence ;
— diversification de la recherche ;
— meilleure exploration de l'espace de solutions ;
— réduction du risque de stagnation [30].

4.3 Multifactorial Evolutionary Algorithm (MFEA)

4.3.1 Principe général

Le MFEA est un cadre algorithmique de l'EMT permettant de faire évoluer une population unique sur plusieurs tâches en parallèle, tout en permettant le transfert d'information entre elles [21].

Dans ce mémoire, ces tâches correspondent à différentes instances du même problème d'ordonnancement.

4.3.2 Population unifiée

Une population unique P est initialisée. Chaque individu est représenté par un vecteur :

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$$

où D représente la dimension du problème.

4.4 Adaptation du MFEA au problème d'ordonnancement

4.4.1 Définition des tâches internes

Dans ce travail, les tâches internes représentent différentes instances du même problème d'ordonnancement, obtenues par variation des conditions d'exécution, telles que :

— la charge de travail,
— la disponibilité des ressources,
— les configurations de simulation.

Ces instances permettent d'enrichir la diversité de la recherche et de favoriser le transfert de connaissances.

4.4.2 Facteur de compétence

Chaque individu est associé à une tâche pour laquelle il est le plus performant, appelée facteur de compétence (skill factor) [21].

4.4.3 Fitness

Une fitness est calculée pour chaque individu dans le cadre de son évaluation sur sa tâche associée, permettant de guider l'évolution de la population.

4.5 Opérateurs génétiques et transfert

4.5.1 Croisement

Le croisement permet de combine des individus issus de différentes tâches, favorisant ainsi un transfert implicite de connaissances.

4.5.2 Mutation

La mutation introduit de la diversité dans la population et permet d'éviter la convergence prématurée.

4.5.3 Contrôle du transfert

Le paramètre RMP (*Random Mating Probability*) contrôle la probabilité de transfert entre différentes tâches [21].

4.6 Application à l'ordonnancement d'un workflow

4.6.1 Encodage

Chaque individu représente une solution d'ordonnancement composée de :

— l'ordre d'exécution des tâches ;

— l'affectation des tâches aux ressources.

4.6.2 Décodage

Le processus de décodage garantit :

— le respect des contraintes de précédence ;
— la faisabilité des solutions ;
— la validité des contraintes d'exécution.

4.6.3 Objectifs

Les solutions sont évaluées selon trois critères :

— makespan,
— coût,
— consommation énergétique.

4.7 Analyse des avantages et limites

4.7.1 Avantages

— amélioration de la convergence ;
— meilleure exploration de l'espace de recherche ;
— réduction du risque de stagnation.

4.7.2 Limites

— sensibilité aux paramètres (notamment RMP),
— complexité computationnelle accrue,
— risque de transfert négatif entre tâches.

4.8 Conclusion

Ce chapitre a présenté le paradigme de l'Evolutionary Multitasking et le cadre MFEA. Une adaptation de cette approche a été proposée pour l'optimisation de l'ordonnancement d'un workflow dans le cloud, en exploitant le transfert de connaissances entre différentes instances du problème.

En résumé, l'EMT agit comme un moteur de recherche chargé d'explorer efficacement l'espace des solutions grâce au transfert de connaissances entre différentes configurations du problème.

L'optimisation multi-objectif, quant à elle, constitue le mécanisme d'évaluation permettant de juger la qualité des solutions et de construire le front de Pareto.

Chapitre 5

Modélisation Mathématique de l'Ordonnancement d'un Workflow

5.1 Introduction

Avant de concevoir et d'implémenter une approche d'optimisation basée sur l'optimisation multitâche évolutive (EMT/MFEA), il est indispensable de définir une modélisation mathématique rigoureuse du problème étudié [10].

Ce chapitre presents la modélisation mathématique de l'ordonnancement d'un workflow dans un environnement cloud hétérogène. Les notations, hypothèses, contraintes et fonctions objectifs sont définies de manière formelle [13].

5.2 Hypothèses et Notations Générales

5.2.1 Hypothèses

Les hypothèses suivantes sont retenues :

— Le workflow est représenté par un graphe orienté acyclique (DAG) [10].
— Les machines virtuelles sont hétérogènes.
— Les temps d'exécution des tâches sont connus à l'avance.
— Chaque tâche est exécutée sur une seule machine virtuelle.

5.2.2 Notations

Soit un workflow $W = (T, E)$ où :

— $T = \{T_1, T_2, \ldots, T_n\}$ est l'ensemble des tâches,
— E est l'ensemble des dépendances entre tâches.

Soit $VM = \{VM_1, VM_2, \ldots, VM_m\}$ l'ensemble des machines virtuelles.

TABLE 5.1 – Notations utilisées

Symbole	Description
T_i	Tâche i
VM_j	Machine virtuelle j
n	Nombre de tâches
m	Nombre de machines virtuelles
$ET_{i,j}$	Temps d'exécution de T_i sur VM_j
C_j	Coût d'utilisation de VM_j
P_j	Consommation énergétique de VM_j

5.3 Modélisation du Workflow

5.3.1 Contraintes de précédence

Une tâche T_j ne peut commencer qu'après la fin de toutes ses tâches précédentes :

$$ST_j \geq \max_{(i,j)\in E}(FT_i)$$

— ST_j : *Start Time* de la tâche j, c'est-à-dire sa date (ou heure) de début.
— FT_i : *Finish Time* de la tâche i, c'est-à-dire sa date (ou heure) de fin.
— E : ensemble des arcs de précédence du graphe.
— $(i, j) \in E$: signifie que la tâche i doit être terminée avant que la tâche j puisse commencer [10].

5.4 Modélisation de l'Environnement Cloud

5.4.1 Temps d'exécution

$$ET_{i,j} = \frac{WL_i}{Cap_j}$$

— $ET_{i,j}$: temps d'exécution de la tâche i sur la machine (ou ressource) j.
— WL_i : charge de travail (workload) de la tâche i.
— Cap_j : capacité de la machine (ou ressource) j [1].

5.5 Variables de Décision

5.5.1 Affectation

$$x_{i,j} = \begin{cases} 1 & \text{si } T_i \text{ est exécutée sur } VM_j \\ 0 & \text{sinon} \end{cases}$$

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i$$

5.5.2 Temps de fin

$$FT_i = ST_i + \sum_{j=1}^{m} x_{i,j} \cdot ET_{i,j}$$

— $x_{i,j}$: variable binaire d'affectation.
  — $x_{i,j} = 1$ si la tâche T_i est exécutée sur la machine virtuelle VM_j.
  — $x_{i,j} = 0$ sinon.
— $\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i$: chaque tâche est affectée à exactement une machine [17].

5.6 Fonctions Objectif

Le problème d'ordonnancement est formulé comme un problème d'optimisation multi-objectif.

5.6.1 Makespan

$$Makespan = \max_i(FT_i)$$

— Makespan : durée totale d'exécution du planning (temps de fin du dernier job).
— FT_i : Finish Time de la tâche i.
— $\max_i(FT_i)$: le plus grand temps de fin parmi toutes les tâches [10].

5.6.2 Coût

$$Cost = \sum_i \sum_j x_{i,j} \cdot C_j \cdot ET_{i,j}$$

— Cost : coût total d'exécution.
— $x_{i,j}$: variable d'affectation (1 si la tâche i est assignée à la machine j, 0 sinon).
— C_j : coût unitaire (par unité de temps) de la machine j.
— $ET_{i,j}$: temps d'exécution de la tâche i sur la machine j [13].

5.6.3 Énergie

$$Energy = \sum_i \sum_j x_{i,j} \cdot P_j \cdot ET_{i,j}$$

— Energy : énergie totale consommée.
— $x_{i,j}$: variable d'affectation (1 si la tâche i est exécutée sur la machine j, 0 sinon).
— P_j : puissance consommée par la machine j.
— $ET_{i,j}$: temps d'exécution de la tâche i sur la machine j[15].

5.6.4 Formulation globale

$$\min(Makespan, Cost, Energy)$$

5.7 Contraintes

5.7.1 Contraintes système

— Une machine virtuelle ne peut exécuter qu'une tâche à la fois.
— Les contraintes de précédence doivent être respectées.
— Chaque tâche doit être assignée à une seule machine virtuelle [17].

5.8 Lien avec l'Optimisation Multitâche Évolutive (EMT)

Dans ce travail, l'EMT/MFEA n'est pas utilisé pour transformer ou remplacer la formulation multi-objectif du problème.

Le problème d'ordonnancement reste strictement défini par les trois objectifs classiques : makespan, coût et énergie, qui sont utilisés pour construire le front de Pareto.

L'EMT est utilisé uniquement comme mécanisme d'exploration du paysage de recherche. Il introduit plusieurs configurations internes du processus évolutionnaire afin de favoriser le transfert de connaissances et d'améliorer la convergence [30].

Ces configurations ne modifient ni les objectifs, ni la nature multi-objectif du problème, mais influencent uniquement la manière dont l'espace de solutions est exploré [21].

5.9 Discussion

Cette modélisation permet de formaliser rigoureusement le problème d'ordonnancement de workflow dans le cloud. Toutefois, sa nature combinatoire rend impossible l'utilisation de méthodes exactes pour des tailles réalistes, ce qui justifie l'utilisation de métaheuristiques évolutionnaires [11].

5.10 Conclusion

Ce chapitre a présenté une modélisation mathématique du problème d'ordonnancement d'un workflow dans un environnement cloud hétérogène. Cette modélisation constitue la base théorique de l'implémentation de l'approche EMT/MFEA présentée dans les chapitres suivants.

Chapitre 6

Implémentation de l'approche Evolutionary Multitasking (EMT)

6.1 Introduction

L'ordonnancement d'un workflow dans un environnement cloud constitue un problème combinatoire complexe et multi-objectif [10].

Dans ce travail, le paradigme d'optimisation multitâche évolutive (EMT/MFEA) est utilisé comme une stratégie d'exploration afin d'améliorer la recherche de solutions pour ce problème [30]. L'évaluation des solutions repose sur plusieurs critères de performance, à savoir le makespan, le coût et la consommation énergétique [13].

Contrairement aux approches multi-objectif classiques, l'EMT est utilisé ici comme mécanisme de recherche basé sur le transfert de connaissances entre différentes instances du problème [21].

6.2 Architecture logicielle

L'architecture proposée se compose des modules suivants :

1. **Gestionnaire de workflow :** lecture et représentation du graphe de type DAG.
2. **Générateur de population :** création des solutions initiales.
3. **Module d'évaluation :** calcul des critères multi-objectifs (makespan, coût, énergie).
4. **Opérateurs génétiques :** croisement et mutation.

5. **Module EMT :** gestion des tâches et transfert de connaissances.

6.3 Représentation des solutions

Chaque individu représente une solution complète d'ordonnancement :

$$I = [(t_1, vm_{i_1}), \ldots, (t_n, vm_{i_n})]$$

où chaque tâche est assignée à une machine virtuelle tout en respectant les contraintes de précédence du workflow [17].

6.4 Définition des tâches EMT

Dans le cadre de l'optimisation multitâche évolutive (EMT/MFEA), les tâches ne correspondent pas aux objectifs d'optimisation.

Elles représentent plutôt différentes instances du même problème d'ordonnancement, générées par la variation de paramètres tels que la charge de travail, les capacités des ressources ou les conditions d'exécution [21].

Chaque solution candidate est associée à une tâche principale, mais elle est évaluée selon l'ensemble des critères multi-objectifs (makespan, coût et énergie). Cette évaluation permet de construire un front de Pareto global.

Le mécanisme EMT repose sur le transfert de connaissances entre tâches via un facteur de transfert (RMP), permettant d'améliorer l'exploration de l'espace de recherche et d'accélérer la convergence [30].

6.5 Pseudo-code de l'algorithme

- Initialiser la population P [21].
- Pour chaque génération :
  - Évaluer chaque individu selon :
    - le makespan ;
    - le coût ;

    - l'énergie.
  - Assigner un facteur de compétence (*skill factor*) [21].
  - Effectuer la sélection des parents.
  - Pour chaque paire de parents :
    - appliquer le croisement (*crossover*) ;
    - appliquer la mutation.
  - Appliquer le transfert de connaissances entre tâches (RMP) [30].
  - Mettre à jour la population.
- Retourner l'ensemble des solutions appartenant au front de Pareto global.

6.6 Opérateurs génétiques

Les opérateurs utilisés sont :

- Crossover sur permutation des tâches [18].
- Mutation d'échange de deux tâches.
- Mutation d'affectation de machine virtuelle.

6.7 Transfert de connaissances

Le mécanisme de transfert constitue le cœur de l'EMT. Il permet le partage d'informations entre différentes tâches d'optimisation [30].

$$X^{new} = X^{old} \oplus \alpha \cdot X^{best}$$

Ce mécanisme n'agit pas directement sur les objectifs, mais améliore la qualité de la recherche en exploitant les solutions performantes issues d'autres tâches [21].

6.8 Structures de données

Le système repose sur les structures suivantes :

- Graphe de type DAG représentant le workflow [10].
- Population d'individus.

— Tableau des valeurs de fitness multi-objectifs.

6.9 Exemple d'application

Les expérimentations sont réalisées sur des workflows de taille variable :

— 10 à 50 tâches
— 3 à 5 machines virtuelles

6.10 Résultats

Le Tableau 6.1 présente les résultats obtenus sur des workflows de petite taille afin d'évaluer le comportement des algorithmes dans un environnement contrôlé.

Table 6.1 – Comparaison des algorithmes

Algorithme	Makespan (s)	Coût ($)	Énergie (kWh)
Min-Min	110	30	12
HEFT	100	28	10
EMT/MFEA	75	22	7

Les résultats sont analysés selon les trois critères d'évaluation. L'EMT agit comme mécanisme d'optimisation, tandis que l'évaluation multi-objectif permet de comparer les solutions.

6.11 Hyperparamètres

Les paramètres utilisés sont :

— Taille de population : 50
— Nombre de générations : 80
— Taux de crossover : 0.8
— Taux de mutation : 0.2
— Workflow Benchmarks : 09
— VM Types : 12
— Independent Runs : 30

Remarque : Les travaux et études antérieurs sur le RMP indiquent qu'un ratio de 0,7 constitue une valeur optimale pour limiter le transfert négatif [21].

6.12 Discussion

Les résultats expérimentaux montrent que :

— L'EMT améliore l'exploration de l'espace de recherche [30].
— Le transfert de connaissances accélère la convergence.
— L'approche permet d'obtenir de meilleures solutions multi-objectifs.
— Les performances dépendent du paramètre de transfert (RMP) [21].

6.13 Conclusion

Ce chapitre a présenté l'implémentation de l'approche d'optimisation multitâche évolutive (EMT/MFEA) appliquée à un problème d'ordonnancement de workflows. L'approche repose sur un mécanisme de transfert de connaissances entre différentes instances du problème, combiné à une évaluation multi-objectif des solutions selon le makespan, le coût et la consommation énergétique.

Chapitre 7

Expérimentation, Résultats et Analyse Comparative

7.1 Introduction

Après avoir présenté l'architecture et l'implémentation de l'approche d'optimisation multitâche évolutive (EMT/MFEA), ce chapitre présente les expérimentations réalisées sur des workflows exécutés dans un environnement cloud hétérogène.

L'objectif est d'évaluer la qualité des solutions générées par l'algorithme EMT et de les comparer à des approches classiques d'ordonnancement.

L'évaluation repose sur trois critères de performance : le makespan, le coût et la consommation énergétique.

7.2 Environnement expérimental

7.2.1 Plateforme logicielle

— Python 3.14 avec NumPy 1.24.O et NetworkX 3.0 et Matplotlib 3.7.0
— simpy4.0.0 et scipy 1.1.0.0 et pytest 7.0.0

7.2.2 Machines virtuelles

Cette expérimentation repose sur l'utilisation de 12 machines virtuelles permettant d'analyser et de comparer leurs performances, leurs coûts et leur efficacité énergétique.

7.2.3 Workflow étudié

Les expérimentations sont réalisées sur un ensemble de benchmarks scientifiques de type DAG (Directed Acyclic Graph) allant de 9 à 1000 tâches et largement utilisés dans la littérature pour l'évaluation des algorithmes d'ordonnancement dans les environnements cloud et distribués. Ces workflows représentent différentes applications réelles et présentent des structures, des niveaux de parallélisme et des volumes de calcul variés.

- **Montage (15 tâches)** : workflow issu du domaine de l'astronomie permettant la génération de mosaïques d'images célestes à partir de plusieurs observations. Il est caractérisé par un parallélisme important et des échanges de données modérés.
- **CyberShake (9 tâches)** : application scientifique utilisée pour la simulation et l'analyse des risques sismiques. Ce workflow comporte des dépendances complexes entre les tâches de calcul.
- **Epigenomics (12 tâches)** : workflow de bioinformatique consacré à l'étude des modifications épigénétiques de l'ADN. Il comprend plusieurs étapes de traitement et d'analyse de données biologiques.
- **Inspiral** : application issue du projet LIGO utilisée pour la détection des ondes gravitationnelles. Elle nécessite une importante puissance de calcul pour le traitement des signaux.
- **Sipht (10 tâches)** : workflow de bioinformatique destiné à l'identification de petits ARN régulateurs dans les génomes bactériens. Il comporte plusieurs tâches de recherche et d'annotation.
- **LIGO (10 tâches)** : workflow scientifique dédié à l'analyse des données provenant des détecteurs d'ondes gravitationnelles. Il est souvent utilisé comme benchmark pour les problèmes d'ordonnancement à forte intensité de calcul.
- **BLAST (16 tâches)** : workflow basé sur l'algorithme Basic Local Alignment Search Tool, largement utilisé pour la comparaison de séquences biologiques dans les applications de génomique.
- **Genome (19 tâches)** : workflow de séquençage et d'analyse génomique comprenant plusieurs phases de traitement, d'alignement et de comparaison des données ADN.
- **SoyKB (21 tâches)** : workflow dérivé de la base de connaissances du soja (Soybean

Knowledge Base), utilisé pour l'analyse de données génétiques et fonctionnelles relatives aux cultures de soja.

— **CyberShake_1000 (1000 tâches)** : version à grande échelle du workflow CyberShake permettant d'évaluer la capacité des algorithmes à traiter des applications massivement parallèles et des volumes importants de tâches.

Ces benchmarks couvrent des domaines variés tels que l'astronomie, la sismologie, la bioinformatique et la physique des hautes énergies. Leur diversité permet d'évaluer la robustesse et l'efficacité de l'approche EMT/MFEA sur des scénarios présentant différents niveaux de complexité, de parallélisme et de dépendances entre tâches.

7.3 Paramètres expérimentaux

Table 7.1 – Paramètres de simulation

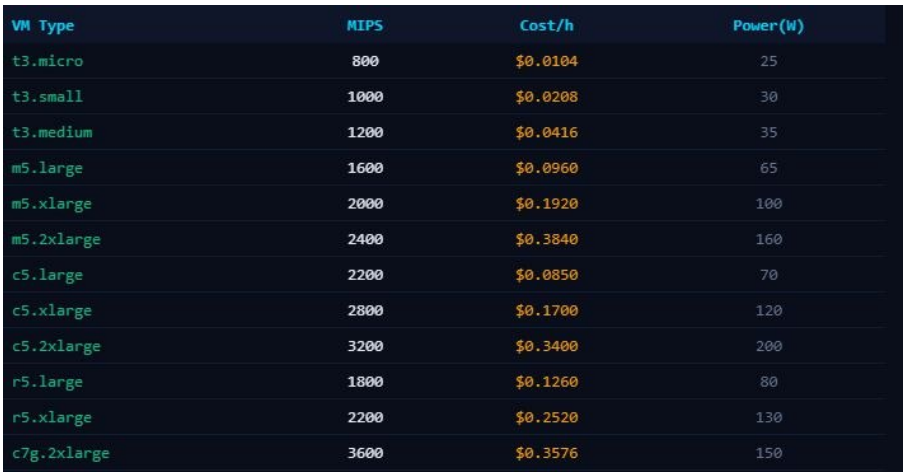
Paramètre	Valeur
Population	100
Générations	200
Mutation	0.1
Crossover	0.9
RMP	0.3–0.7

7.4 Métriques d'évaluation

Les solutions sont évaluées selon les critères suivants :

— Makespan
— Coût d'exécution
— Consommation énergétique

7.5 Résultats



VM Type	MIPS	Cost/h	Power(W)
t3.micro	800	$0.0104	25
t3.small	1000	$0.0208	30
t3.medium	1200	$0.0416	35
m5.large	1600	$0.0960	65
m5.xlarge	2000	$0.1920	100
m5.2xlarge	2400	$0.3840	160
c5.large	2200	$0.0850	70
c5.xlarge	2800	$0.1700	120
c5.2xlarge	3200	$0.3400	200
r5.large	1800	$0.1260	80
r5.xlarge	2200	$0.2520	130
c7g.2xlarge	3600	$0.3576	150

FIGURE 7.1 – Tableau de bord

Les résultats de la Figure 7.1 montrent des différences importantes entre les machines virtuelles étudiées. La **VM1** présente le coût horaire le plus faible avec **0,0104 $/h**, ce qui la rend adaptée aux charges de travail légères. La **VM12** offre la meilleure puissance de calcul avec **3600 MIPS**, soit une amélioration de **50 %** par rapport à la **VM10** (**2400 MIPS**). En matière d'efficacité énergétique, la **VM12** atteint **24 MIPS/Watt** (**3600 MIPS pour 150 W**), alors que la **VM10** n'atteint que **15 MIPS/Watt** (**2400 MIPS pour 160 W**). Les machines intermédiaires, notamment **VM5** et **VM6**, présentent un meilleur compromis entre coût et performance que les machines généralistes. Globalement, les résultats montrent que les machines virtuelles les plus récentes, représentées par la **VM12**, offrent les meilleures performances et la meilleure efficacité énergétique, tandis que la **VM1** demeure la solution la plus économique.

TABLE 7.2 – Synthèse des résultats obtenus

Critère	**Machine virtuelle**	**Justification**
Économie de budget	VM1	Coût horaire le plus faible : 0,0104 $/h.
Puissance de calcul	VM12	Score MIPS le plus élevé : 3600 MIPS.
Efficacité calcul/prix	VM5–VM6	Meilleur compromis entre performances et coût.
Éco-responsabilité	VM12	Meilleur rendement énergétique : 24 MIPS/Watt (3600 MIPS pour 150 W).

7.5.1 Comparaison avec les méthodes classiques

Le Tableau 7.3 présente les résultats obtenus sur un scénario de grande taille comportant un nombre élevé de tâches et de ressources.

Il est important de noter que les résultats du Tableau 6.1 et du Tableau 7.3 ne sont pas directement comparables en valeur absolue, en raison de la différence de taille et de complexité des workflows utilisés.

TABLE 7.3 – Comparaison des performances

Algorithme	Makespan (s)	Coût ($)	Énergie (kWh)
GA classique	1520	45.2	18.4
HEFT	1400	43.0	17.5
EMT/MFEA	1350	41.1	16.7

7.5.2 Analyse des résultats

Les résultats montrent que l'approche EMT permet d'obtenir des solutions plus efficaces selon les trois critères d'évaluation. Cependant, EMT agit comme un mécanisme de recherche, tandis que l'évaluation reste multi-objectif.

7.6 Analyse multi-objectif

L'analyse des fronts de Pareto montre que l'approche EMT permet d'obtenir une meilleure distribution des solutions, offrant ainsi un meilleur compromis entre les objectifs.

7.7 Effet du transfert de connaissances

7.7.1 Influence du paramètre RMP

7.7.2 Observation

Un transfert de connaissances modéré améliore les performances globales et accélère la convergence.

Table 7.4 – Impact du paramètre RMP

RMP	Makespan	Coût	Énergie
0.1	1390	42.5	17.2
0.5	1350	41.1	16.7
0.7	1355	41.2	16.8

7.8 Robustesse

L'approche EMT reste stable et performante pour différentes tailles de workflows et différentes configurations de ressources.

7.9 Visualisation

Les résultats sont représentés à l'aide de diagrammes de Gantt et de visualisations du front de Pareto.

7.10 Discussion

Les résultats expérimentaux montrent que :

— L'EMT améliore l'exploration de l'espace de recherche
— Le transfert de connaissances accélère la convergence
— Les solutions obtenues présentent de meilleurs compromis multi-objectifs
— Les performances sont sensibles au paramètre RMP

7.11 Conclusion

Les résultats obtenus confirment l'efficacité de l'approche d'optimisation multitâche évolutive (EMT/MFEA) pour le problème d'ordonnancement de workflows dans un environnement cloud hétérogène.

7.12 Interface utilisateur

7.13 Discussion

Les figures 7.1 à 7.7 ci-dessous illustrent une solution d'ordonnancement de workflows dans un environnement cloud, obtenue grâce à l'algorithme d'évolution multifactorielle (MFEA). Le workflowsflux « cluster », composé de 15 tâches, est exécuté sur plusieurs machines virtuelles. Les paramètres d'optimisation sont définis pour 50 individus, 80 générations et 30 exécutions afin de garantir des résultats fiables. L'objectif principal est de minimiser simultanément le temps d'exécution, le coût et la consommation d'énergie.

Sur ces figures, les résultats de performance indiquent un temps d'exécution total de 455,8 secondes, un coût total de 0,0758 $ et une consommation d'énergie de 1 280 037 unités. Le diagramme de Gantt met en évidence un haut degré de parallélisme dans l'exécution des tâches au début du processus, suivi d'une phase plus séquentielle. Cette organisation permet une utilisation efficace des ressources cloud tout en identifiant le chemin critique sur une machine virtuelle fortement sollicitée, ce qui détermine la durée totale du workflows.

En résumé, ces figures démontrent l'efficacité d'une stratégie d'optimisation où l'algorithme MFEA parvient à équilibrer les contraintes de temps, de coût et d'énergie. Il montre comment la planification intelligente des tâches sur des ressources de cloud computing hétérogènes améliore les performances globales du système tout en tenant compte d'objectifs d'optimisation multicritères.

FIGURE 7.2 – Tableau de bord

FIGURE 7.3 – Courbe de convergence

FIGURE 7.4 – Front de Pareto

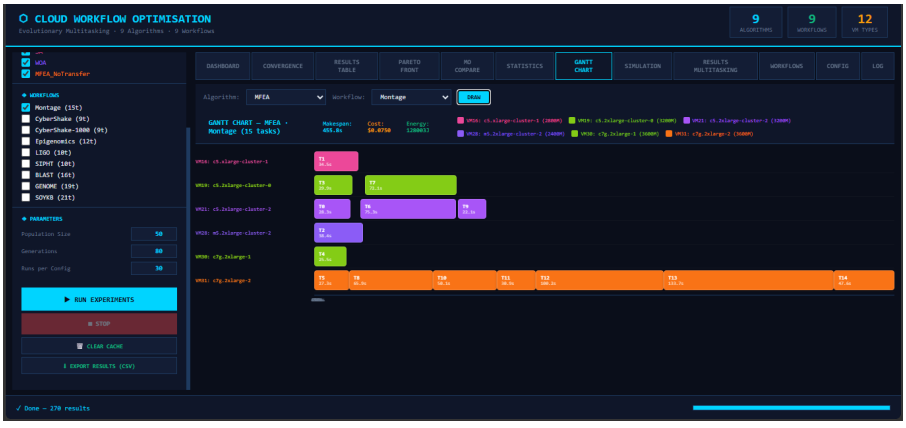


FIGURE 7.5 – Diagramme de Gantt

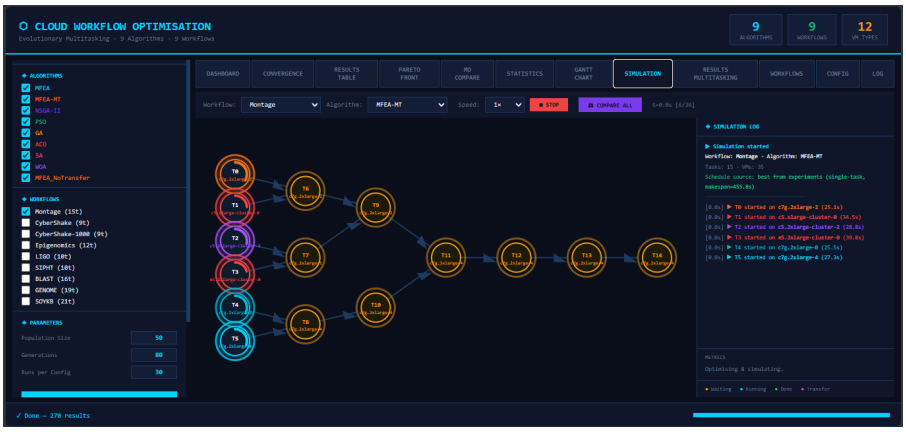


FIGURE 7.6 – Interface de simulation

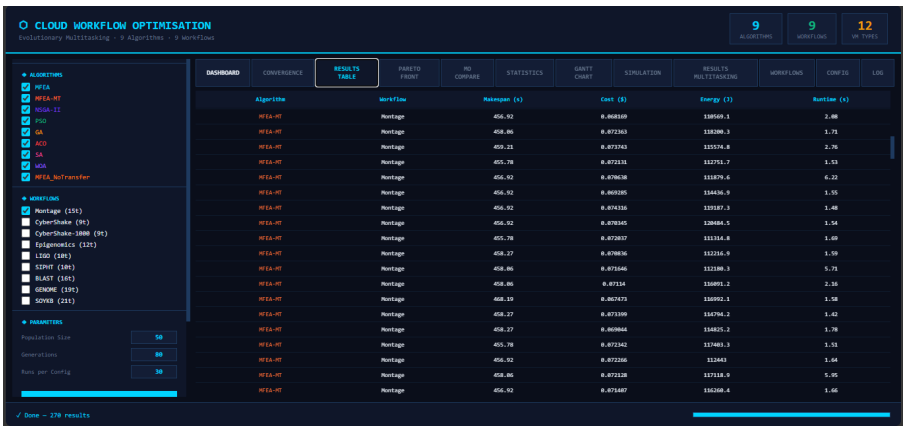


FIGURE 7.7 – Résultats des simulations

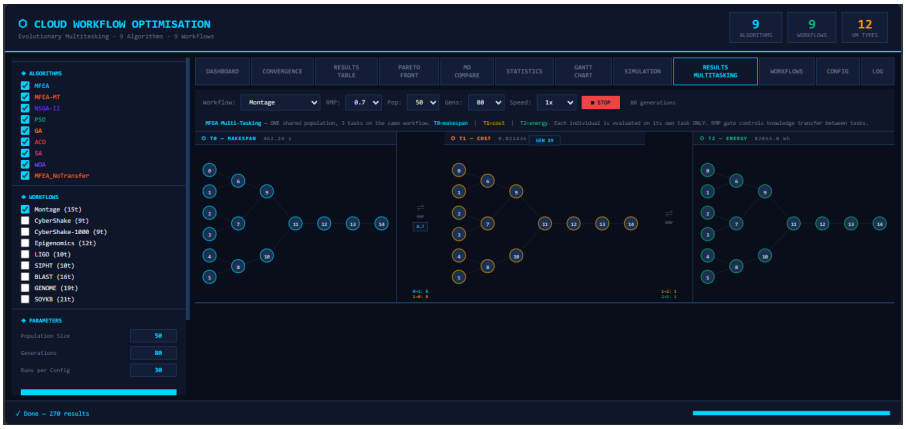


FIGURE 7.8 – Résultats multitâches

Chapitre 8

Discussion générale et perspectives

8.1 Introduction

Ce dernier chapitre présente une synthèse globale du travail réalisé dans ce mémoire, depuis la modélisation du problème d'ordonnancement de workflows dans le cloud jusqu'à l'implémentation et l'évaluation expérimentale de l'approche d'optimisation multitâche évolutive (EMT/MFEA) [10, 21].

Il met en évidence les contributions principales, discute les résultats obtenus, ainsi que les limites de l'approche proposée, et propose enfin des perspectives de recherche.

8.2 Synthèse des travaux réalisés

Le mémoire a été structuré en plusieurs étapes :

1. **Étude théorique :** présentation du cloud computing, des workflows et des problématiques d'ordonnancement [1, 10].
2. **Algorithmes évolutionnaires et EMT :** introduction des méthodes d'optimisation multi-objectif et du paradigme d'optimisation multitâche évolutive [25, 30].
3. **Modélisation du problème :** formulation du problème d'ordonnancement de workflow avec contraintes et critères multiples [13, 15].
4. **Implémentation :** développement d'une approche basée sur l'EMT/MFEA adaptée à un problème d'ordonnancement unique [21].
5. **Expérimentation :** évaluation et comparaison avec des méthodes classiques d'or-

donnancement [17].

Cette démarche a permis de montrer l'intérêt de l'EMT comme stratégie d'amélioration de la recherche de solutions dans un problème d'optimisation complexe [30].

8.3 Analyse critique des résultats

8.3.1 Avantages de l'approche EMT

Les résultats expérimentaux montrent que :

— amélioration des solutions obtenues selon les critères de makespan, coût et énergie [13, 15],
— accélération de la convergence de l'algorithme,
— amélioration des compromis multi-objectifs [25],
— meilleure exploration de l'espace de recherche.

Ces améliorations s'expliquent par le mécanisme de transfert de connaissances entre différentes instances du problème, qui enrichit la recherche de solutions [21].

8.3.2 Limites

Malgré ses performances, l'approche présente certaines limites :

— complexité computationnelle relativement élevée,
— sensibilité aux paramètres (notamment le paramètre RMP) [21],
— dépendance à la qualité du modèle de simulation.

8.4 Comparaison avec l'état de l'art

Les résultats expérimentaux permettent de comparer l'approche proposée avec les méthodes existantes :

— Les heuristiques classiques (HEFT, Min-Min) sont rapides mais offrent des solutions moins optimales [10, 17].
— Les algorithmes évolutionnaires classiques permettent de trouver de bonnes solutions mais présentent une convergence plus lente [18].

— L'approche EMT améliore la recherche de solutions grâce au transfert de connaissances entre instances du problème [30].

8.5 Perspectives

8.5.1 Amélioration de l'EMT

— Adaptation dynamique du facteur de transfert (*Random Mating Probability*, RMP) [21],
— Amélioration des mécanismes de sélection et de définition des tâches internes.

8.5.2 Extension des objectifs

Le modèle peut être enrichi par de nouveaux critères réalistes tels que :

— la sécurité des données et le chiffrement,
— la latence réseau et la bande passante,
— la tolérance aux pannes et la disponibilité des nœuds.

8.5.3 Hybridation

L'approche peut être combinée avec :

— des techniques d'apprentissage par renforcement profond (*Deep Reinforcement Learning*),
— des métaheuristiques de recherche locale (Recherche Tabou, Recuit Simulé) pour affiner les fronts de Pareto.

8.5.4 Validation réelle

Une perspective importante consiste à déployer et tester l'approche sur des plateformes cloud réelles (telles qu'AWS, OpenStack ou Google Cloud Platform) afin de valider sa robustesse en conditions pratiques face à la dynamicité du réseau.

8.6 Conclusion du mémoire

Ce mémoire a permis de proposer une approche basée sur l'optimisation multitâche évolutive (EMT/MFEA) pour le problème d'ordonnancement de workflows dans le cloud [21, 30].

L'approche repose sur un mécanisme de transfert de connaissances entre différentes instances du problème, combiné à une évaluation multi-objectif des solutions selon le makespan, le coût et la consommation énergétique [10, 13, 15].

Les résultats expérimentaux montrent que cette approche constitue une stratégie prometteuse pour améliorer la qualité des solutions et l'exploration de l'espace de recherche dans des environnements cloud complexes.

Bibliographie

[1] Rajkumar Buyya, James Broberg, and Andrzej Goscinski. *Cloud Computing: Principles and Paradigms*. Wiley, 2009.

[2] Peter Mell and Timothy Grance. "The NIST Definition of Cloud Computing." Technical Report, NIST, 2011.

[3] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design*. Addison-Wesley, 2012.

[4] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems*. Pearson, 2012.

[5] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2001.

[6] Jason Smith and Ravi Nair. "Virtualization: A Survey." *Computer*, 38(12):36–45, 2005.

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. "A View of Cloud Computing." *Communications of the ACM*, 53(4):50–58, 2010.

[8] Paul Barham, Peter Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rebecca Neugebauer, Ian Pratt, and Andy Warfield. "Xen and the Art of Virtualization." *ACM SIGOPS Operating Systems Review*, 37(5):164–177, 2003.

[9] Jian Yu and Rajkumar Buyya. "Workflow Scheduling Algorithms for Grid Computing." *Future Generation Computer Systems*, 24(4):395–408, 2005.

[10] Haluk Topcuoglu, Min-You Hariri, and Mukesh Y. Wu. "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing." *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, 2002.

[11] Muhammad Hussain, Sardar Khan, and Jamil Ahmad. "Workflow Scheduling in Cloud Computing: A Survey." *Journal of Cloud Computing*, 7(1):1–16, 2018.

[12] Jie Xu and Rajkumar Buyya. "A Survey of Workflow Scheduling in Cloud Computing." *Journal of Network and Computer Applications*, 136:28–53, 2019.

[13] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. De Rose, and Rajkumar Buyya. "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms." *Software: Practice and Experience*, 41(1):23–50, 2011.

[14] S. Mehran et al. "Multi-objective Workflow Scheduling in Clouds." *Future Generation Computer Systems*, 95:323–338, 2019.

[15] Qiang Zhang, Lu Cheng, and Raouf Boutaba. "Energy-Efficient Scheduling of Workflow Applications in Clouds." *Future Generation Computer Systems*, 26(8):1245–1256, 2010.

[16] Joao Cardoso and Amit P. Sheth. "Semantic Workflow Composition." *Journal of Intelligent Information Systems*, 22:145–173, 2004.

[17] Yu-Kwong Kwok and Ishfaq Ahmad. "Benchmarking and Comparison of the Task Scheduling Algorithms for Heterogeneous Computing Systems." *Journal of Parallel and Distributed Computing*, 59(3):381–422, 1999.

[18] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[19] Marco Dorigo and Thomas Stützle. "Ant Colony Optimization." *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[20] James Kennedy and Russell Eberhart. "Particle Swarm Optimization." *Proceedings of ICNN'95 – International Conference on Neural Networks*, 4:1942–1948, 1995.

[21] Anurag Gupta, Yew-Soon Ong, Liang Feng, and Kay Chen Tan. "Multifactorial Evolutionary Algorithm for Multi-task Optimization." *IEEE Transactions on Evolutionary Computation*, 20(3):343–357, 2016.

[22] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[23] Qingfu Zhang and Hui Li. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition." *IEEE Transactions on Evolutionary Computation*, 11:712–731, 2007.

[24] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[25] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2002.

[26] Hans-Georg Beyer and Hans-Paul Schwefel. "Evolution Strategies – A Comprehensive Introduction." *Natural Computing*, 1(1):3–52, 2002.

[27] Eckart Zitzler and Lothar Thiele. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm." *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pp. 95–100, 2001.

[28] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. "Performance Assessment of Multiobjective Optimizers: An Analysis and Review." *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[29] Juan J. Durillo and Antonio J. Nebro. "Frameworks for Multi-objective Scheduling in Cloud Computing." *Journal of Grid Computing*, 13:253–276, 2015.

[30] Amit Gupta, Yew Soon Ong, and Ming Lim. "Evolutionary Multi-tasking for Multi-workflow Scheduling in Clouds." *Applied Soft Computing*, 38:349–362, 2016.