N° d'ordre : 12/2023-D/IN

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumedienne

Faculté d'Informatique

THESE DE DOCTORAT EN SCIENCES

Présentée pour l'obtention du grade de DOCTEUR EN SCIENCES

En : INFORMATIQUE

Spécialité : Systèmes Intélligents et Ingénierie du Logiciel

Par

AZOUZYacine

Thème

Approches méta-heuristiques pour la composition des Services Web

Soutenue le 06/06/2023, devant le jury composé de :

M S. KECHID	Professeur àl'USTHB	Président
Mme D. BOUGHACI	Professeur àl'USTHB	Directrice de thèse
M M. GUERROUMI	MC/A àl'USTHB	Examinateur
M A. REZOUG	MC/A, Univ. Boumerdès	Examinateur
M M. BOULIF	Professeur, Univ. Boumerdès	Examinateur

M O. KAZAR Professeur, Univ. Biskra Examinateur

Remerciements

Ce travail de recherche a été réalisé au sein du Laboratoire de recherche en intelligence artificielle (LRIA) à la Faculté d'Informatique, Université desSciences et de la Technologie Houari Boumediene (USTHB), dirigé par Madame Pr Dalila BOUGHACI.

Je tiens tout particulièrement à remercierde nouveau Madame Dalila BOUGHACI, Professeur àl'USTHB, pour m'avoir encadré et guidé pendant toutes ces années ainsi que pour les conseilsprécieux qu'elle m'a apportés et pour sa disponibilité, sa noblesse et sa grande gentillesse.Je tiens à lui exprimer ma profonde gratitude et ma vive reconnaissance.

Je tiens à adresser mes sincères remerciements à Monsieur S. KECHID Professeur à l'USTHB, qui m'a fait l'honneur de présider le jury de ma thèse. Mes remerciements s'adressent également à Monsieur Mohamed GURROUMI, Maitre de conférences à l'USTHB, Mr Abdellah REZOUG, Maitre de conférencesà l'Université de Boumerdès, Monsieur Menouar BOULIF, Professeur à l'Université de Boumerdès et Monsieur Okba KAZAR, Professeur à l'université de Biskra, pour avoir accepté d'examiner ce travail.

Je remercie toute ma famille pour leur compréhension et plus spécifiquement ma chère maman pour son soutien moral, j'espère que ce travailsoit la bonne expression de ma gratitude.

Enfin, je rends hommage et j'exprime ma reconnaissance à tous ceux qui ont contribué, de près ou de loin, à la réalisation de ce travail.

Dédicaces

*A*ma mère et à mon père,

*A*ma petite & grande famille surtout mon fils Abdelbari Joud et à tous mes amis,

A tous mes collègues,

*J*e dédie cette thèse

Résumé :

La composition des services est le processus de combinaison d'un ensemble de services élémentaires ou atomiques. L'objectif est de produire un nouveau service composite pour satisfaire la demande de l'utilisateur qui ne peut être satisfait par les services atomiques. Combiner plusieurs services est un problème complexe qui a fait l'objet de plusieurs études de recherche.

Les approches méta-heuristiques sont des bonnes techniques qui ont été utilisées pour résoudre plusieurs problèmes complexes dans divers domaines. Ces techniques sont capables de découvrir des régions de recherche prometteuses et de localiser des solutions de bonne qualité dans un délai raisonnable sans explorer tout l'espace des solutions.

Dans ce présent travail, nous traitons le problème de la composition optimale des services web en utilisant des approches méta-heuristiques. Étant donné un ensemble de services et un ensemble de tâches à accomplir, le problème est de trouver le meilleur ensemble de composition de services pour accomplir toutes les tâches où chaque service doit être affecté à une tâche donnée. Ce problème peut être modélisé comme un problème d'optimisation combinatoire avec un ensemble de fonctions objectifs qui doivent être optimisées. Nous recherchons un service composite qui nous permette d'exécuter les tâches envisagées et offre la meilleure qualité de services (QoS). Plus précisément, nous recherchons un plan d'exécution qui indique pour chaque tâche le service assigné.

Nous avons apporté trois contributions où en premier temps, nous proposons une approche de recherche locale stochastique multi-objectifs (MOSLS). La deuxième contribution consiste à proposer une approche méta-heuristique basée sur l'algorithme de recherche à voisinage variable (MOVNS) et nous finissons par la proposition d'une troisième contribution d'une approche mémétique multi-objectifs (MOMA).

Quatre fonctions objectifs du modèle QoS sont utilisées pour calculer l'ensemble de solutions optimal de Pareto. L'objectif principal est de minimiser les coûts et le temps et de maximiser la disponibilité et réputation et produire un bon service composite. Les trois approches proposées sont évaluées sur certains ensembles de données générés de manière aléatoire et sur l'ensemble de données QWS bien connu pour sélectionner les services les mieux adaptés en termes de paramètres de qualité de service de bout en bout maximum ou minimum agrégés.

Les résultats numériques sont encourageants et démontrent l'efficacité du MO-MA proposées pour l'optimisation de la composition de services web.

Mots clés :algorithme génétique, recherche locale, algorithme mémétique, optimisation multi-objectifs, modèle QoS,composition de services web.

Abstract :

Service composition is the process of combining a set of elementary or atomic services. The goal is to produce a new composite service to satisfy user demand that cannot be satisfied by atomic services. Combining several services is a complex problem that is the subject of several research studies. Meta-heuristic approaches are good techniques that have been used to solve several complex problems in various fields. These techniques are able to discover promising search regions and locate good quality solutions in a reasonable time without exploring the entire solution space.
In this present work, we address the problem of optimal composition of web services using metaheuristic approaches. Given a set of services and a set of tasks to be performed, the problem is to find the best composition set of services to perform all the tasks where each service must be performed at a given task. This problem can be modeled as a combinatorial optimization problem with a set of objective functions that need to be optimized. We are looking for a composite service that allows us to perform the relevant tasks and offer the best quality of services (QoS). More precisely, we are looking for an execution plan that indicates for each task the assigned service.
We proposed three main contributions where first, we presented a multi-objective stochastic local search (MOSLS) approach. The second contribution consists in proposing a meta-heuristic approach based on the variable neighborhood search algorithm (MOVNS) and we finish by proposing a third contribution of a multi-objective memetic approach (MOMA). Four objective functions of the QoS model are used to compute the set of Pareto optimal solutions. The main objective is to minimize costs and time and to maximize availability and reputation and to produce a good composite service. The three proposed approaches are evaluated on some randomly generated datasets and QWS dataset to select the best fit services in terms of aggregated maximum or minimum end-to-end QoS parameters.
Numerical results are encouraged and improved the effectiveness of the MO-MA proposed for the optimization of the composition of web services.

Keywords: genetic algorithm, local search, memetic algorithm, multi-objective optimization, QoS model, composition of web services.

ملخص

تكوين أوتركيب خدمات الويب هو عملية الجمع بين مجموعة من الخدمات الأولية أو الأساسية. الهدف هو إنتاج خدمة مركبة جديدة لتلبية طلب المستخدم الذي لا يمكن تلبيته بواسطة الخدمات الذرية. يعد الجمع بين العديد من الخدمات مشكلة معقدة تخضع للعديد من الدراسات البحثية. المناهج الفوقية meta-heuristics هي تقنيات جيدة تم استخدامها لحل العديد من المشكلات المعقدة في مختلف المجالات. هذه التقنيات قادرة على اكتشاف مناطق البحث الواعدة وتحديد حلول عالية الجودة في وقت معقول دون استكشاف مساحة الحل بأكملها.

في هذا العمل الحالي، نتناول مشكلة التكوين الأمثل لخدمات الويب باستخدام مناهج metaheurism. بالنظر إلى مجموعة الخدمات ومجموعة المهام التي يتعين القيام بها، تكمن المشكلة في العثور على أفضل مجموعة تكوين من الخدمات لأداء جميع المهام حيث يجب أداء كل خدمة في مهمة معينة. يمكن نمذجة هذه المشكلة كمشكلة تحسين اندماجية مع مجموعة من الوظائف الموضوعية التي تحتاج إلى تحسين. نحن نبحث عن خدمة مركبة تتيح لنا أداء المهام ذات الصلة وتقديم أفضل جودة للخدمات (QoS). بتعبير أدق، نحن نبحث عن خطة تنفيذ تشير إلى الخدمة المعينة لكل مهمة.

اقترحنا في دراستنا هذه ثلاث مساهمات رئيسية حيث قدمنا أولاً نهج البحث المحلي العشوائي متعدد الأهداف (MOSLS). تتمثل المساهمة الثانية في اقتراح نهج ما بعد الكشف عن مجريات الأمور بناءً على خوارزمية البحث في الجوار المتغير (MOVNS) وننتهي من خلال اقتراح مساهمة ثالثة لنهج memetic متعدد الأهداف (MOMA). تُستخدم أربع أهداف موضوعية لنموذج جودة الخدمة لحساب مجموعة حلول باريتو الأمثل(Pareto-Optimal).

الهدف الرئيسي هو تقليل التكاليف والوقت وزيادة التوافر والسمعة إلى أقصى حد وإنتاج خدمة مركبة جيدة. يتم تقييم الأساليب الثلاثة المقترحة على بعض مجموعات البيانات التي تم إنشاؤها عشوائيًا ومجموعة بيانات QWS لتحديد أفضل الخدمات الملائمة من حيث الحد الأقصى أو الحد الأدنى من معلمات جودة الخدمة من طرف إلى طرف.

النتائج العدديةالتي تم الحصول عليهاشجعتوحسنت فعالية MO-MA المقترحة لتحسين تكوين خدمات الويب.

الكلمات المفتاحية

الخوارزمية الجينية، البحث المحلي، خوارزمية الذاكرة،التحسين متعدد الاهداف،نموذجالخدمات (QoS)، تكوين خدمات الويب.

Liste des tableaux

Liste de figures :

Table des matières

Liste des algorithmes

Liste des abréviations

ACO: AntColony Optimization

BPEL: Business Process Execution Language

BPEL4WS: Business Process Execution Language for Services web

EA: Algorithmes d'évolution (*Evolutionary Algorithms*)

FTP: File Transfer Protocol

GA: Genetic Algorithm

HTTP: Hyper Text Transfert Protocol

IG : Algorithmes Itératif Greedy

ILS: Iterated Local Search

MA: Memetic Algorithm

MO-VNS: Multi-Objective Variable Neighborhood Search

MOACO: Multi-Objective Ant Colony Optimization

MO-GA:Multi-Objective Genetic Algorithm

NSGA-II: Non-dominated Sorting Genetic Algorithm–II

OWL-S**:** Web Ontology Language for Web Services

PSO : Particle Swarm Optimization (Particle Optimisation des essaims)

QoS: Quality of Service

QWS Dataset: The Quality of Service for Web Services Dataset

RII : Randomised Iterative Improvement

SLS: Search Loacal stochastique

SOAP: Simple Object Access Protocol

SMTP: Simple Mail Transfer Protocol

VNS: Variable Neighborhood Search

TS : Tabu Search (Recherche Tabu)

UDDI: Universal Description, Discovery and Integration

WSDL : Services web Description Language

WSFL: Web Service Flow Language

Introduction Générale

Introduction Générale

1 Contexte

Les avancées remarquables pratiques et technologiques des services web qui assurent l'interopérabilité entre les composants logiciels distribués permettront à différentes entreprises avec différentes infrastructures de collaborer et communiquer les uns avec les autres[1, 2].

Le service Web est un type de modèle informatique distribué, avec des traits d'autonomie, de modularité, de faible couplage, basé sur des normes, haute capacité, etc. L'architecture orienté service (SOA) permet l'intégration de composants de service développés indépendamment en processus métier complexes et applications à valeur ajoutée pour répondre aux besoins des utilisateurs et d'offrir une valeur commerciale supplémentaire.

Utilisant telle architecture, les développeurs considèrent les services comme éléments fondamentaux dans leurs processus de développement d'applications.

Les services web sont basés sur les applications web d'entreprise qui traitent les messages SOAP décrit à l'aide de WSDL envoyé sur HTTP, qui utilisent des normes ouvertes basées sur XML et des protocoles de transport pour échanger des données avec des clients appelants [1, 2].

La composition des services web a suscité beaucoup d'intérêtdans la promotion de l'intégration interentrepriseset applications d'entreprise. Par la composition de services Web,les développeurs de logiciels peuvent créer une variété d'applications complexes en informatique native orienté service : description de service, découverte de service et capacité de communication[2,3,4].

Après tout, la composition de services web a quelques faiblesses qui empêchent les services web d'être largement déployé. Des faiblesses typiques au niveau du système incluent la fiabilité, l'évolutivité, les performances et la sécurité[6]. Nous nous intéressons dans notre travail de thèse de doctorat à l'optimisation de performances de la composition de services web en utilisant les critères non-fonctionnels du modèle QoS.

2 Objectif

La composition de services web est une technologie clé pour créer des services à valeur ajoutée en intégrant les services disponibles. Avec le développement rapide du Service Computing, du Cloud Computing, du Big Data et de l'Internet des objets, un nombre croissant de services avec des fonctionnalités similaires mais une qualité de service (QoS) différente

sont disponibles sur Internet, et font de la composition optimale des services web un NP-problème difficile à résoudre.

Pour créer efficacement un service web composite qui non seulement peut répondre aux besoins des clients, mais qui a également la qualité de service globale optimale reste un défi ouvert. Sur ce, plusieurs méthodes existent ont été proposées mais elles rencontrent des faiblesses d'identifier une solution optimale dans un temps de calcul raisonnable.

Dans ce mémoire, nous proposons plusieurs approches méta-heuristiques d'optimisation multi-objectifs afin de trouver une composition optimale des services web qui permet d'avoir un compromis sur les différents critères de qualité du modèle QoS. Nous étudions et analysonsles méthodesde composition de services en basant sur les méta-heuristiques en s'inspirant des algorithmes de recherche locales stochastiques, les algorithmes génétiques, les algorithmes de recherche à voisinage variableet les algorithmes mémétiques,

La première contribution est donnée dans le troisième chapitre. Elle consiste en une approche hybride basée sur les algorithmes de recherche locale stochastique (SLS) et lesalgorithmes génétiques (GA). Dans ce chapitre, nous présentons tout d'abord lesstructures de ces méthodes. Ensuite, nous proposons une méthode hybride SLS+GA. Le chapitre setermine par quelques résultats numériques. Les approches proposées sont validées surquelques datasets. Les résultats trouvés sont intéressants et montrent l'intérêt de l'approche proposée.

La deuxième contribution est donnée dans le quatrième chapitre. Dans cechapitre, une nouvelle méta-heuristique intéressante est proposée. Elle consiste en uneméthode de recherche à voisinage variable pour le problème d'optimisation multi-objectifs dela composition de services web. Dans ce chapitre, nous présentons la recherche àvoisinage variable (VNS), puis nous décrivons sa méthode VNS multi-objectif proposée pourrésoudre sa problématique. Enfin, nous donnons les résultats numériques trouvés.

La troisième contribution est présentée dans le cinquième chapitre. Elle consisteen une méthode inspirée de l'algorithme mémétique et la recherche locale stochastique. Aussile chapitre donne une étude comparative sur les résultats numériques trouvés par chaquesolution.

3 Organisation

La première partie de ce mémoire présente un état de l'art, au chapitre I nous donnons une introduction au domaine des services web d'une façon générale. Le deuxième chapitre donne une description de l'optimisation multi-objectifs et des méta-heuristiques.

La deuxième parte de ce mémoire présente les trois contributions, où nous abordons dans le troisième chapitre la première contribution qu'il s'agit d'une méthode hybride basée sur les algorithmes de recherche locale stochastique et les algorithmes génétiques, et les principes reliés à ces types d'algorithmes. Nous commençons par la citation de quelques définitions, puis les structure de ces méthodes, et les principes types des méthodes SLS, GA et SLS+GA et nous finissons par exposer les résultats numériques trouvés et nous finissons par une étude comparative.

Le chapitre quatre donne une méthode méta-heuristique de recherche à voisinage variable pour le problème d'optimisation multi-objectifs de la composition de services web.

Nous présentons un aperçu sur la recherche à voisinage variable, puis nous abordons certains types de méthodes VNS, le processus général VNS puis une approche hybride multi-objectifs VNS, nous finissons par exposer les résultats numériques trouvés.

Dans le cinquième chapitre nous présentons la troisième contribution où il y a une méthode inspirée de l'algorithme mémétique et la recherche locale stochastique puis une étude comparative sur les résultats numériques trouvés par chaque solution.

Enfin, nous terminons par une conclusion générale, et quelques perspectives qui peuvent améliorer notre travail.

Toutes les méthodes proposées ont été implémentées et évaluées sur des instances de problèmes.

Chapitre I : Problème de composition des services web: Etat de l'art

I. Problème de composition des services web: Etat de l'art

I.1 Introduction

Dans ce chapitre, nous présentons en première partie les concepts de services et d'Architecture Orientée Service (SOA – Service Oriented Architecture) puis nous abordons les concepts de services web, et par la suite, nous présentons la problématique de la composition des services web. Pour une construction rapide & à faible coût de nouvelles solutions logiciels, le mécanisme de composition de services consiste actuellement un des challenges les plus importants de l'informatique orientée services. Grace à ce mécanisme il devient possible de combiner un ensemble de services fournis afin de satisfaire la demande du client qui ne peut être satisfaite par les services fournis individuellement.

Nous constatons qu'il y a deux niveaux d'abstraction des services. Les services de base qui fournissent des fonctionnalités élémentaires, et les services composites qui agrègent un ensemble de fonctionnalités dans des applications de haut niveau proches des besoins de l'utilisateur. Les services composites sont construits par le processus de composition de services qui permet la liaison entre les besoins de l'utilisateur et les services de base disponibles.

Les services composites se font développer pour résoudre des problèmes complexes en combinant les services de base disponibles et les ordonner pour mieux satisfaire les exigences de l'utilisateur. De plus, le processus de composition de services accélère le développement d'applications, la réutilisation de services, et la réalisation de services plus complexes.

Dans ce chapitre, nous allons exposer les concepts fondamentaux de l'architecture orientée service et les notions de services web. Puis, nous présentons les différents types de composition de services existent dans la littérature.

I.2 L'architecture orientée services

Ces dernières années ont vu l'émergence d'architectures orientées services (SOA) conçues pour faciliter la création, l'exposition, l'interconnexion et la réutilisation d'applications à base de services.

Plusieurs définitions attribuées à la notion d'architectures orientées services.L'architecture orientée services[7]« is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains ». L'architecture orientée services: « SOA represents a model in which functionality is decomposed into small, distinct units (services), which can be distributed over a network and can be combined together and reused to create business applications. »[8].

L’architecture orientée services« est une architecture logicielle visant à mettre en place un système d’information constitué de services applicatifs indépendants et interconnectés » L’architecture SOA permet la réutilisation des applications et des services existants, ainsi de nouveaux services peuvent être créés à partir d’une infrastructure informatique des systèmes déjà existante, en leur offrant une interopérabilité entre applications et technologies hétérogènes. L’objectif d’une architecture SOA est de décomposer une fonctionnalité en un ensemble de services et de décrire leurs interactions[9].

Par ailleurs, l'architecture orientée services (SOA) peut être réalisée par des services web ou des microservices. L'approche des services web conduit à la SOA, tandis que l'architecture des microservices est une extension de la SOA [10]. L'intégration d'applications d'entreprise (EAI) basée sur SOA implémente généralement un bus de services d'entreprise partagé (ESB) pour que les applications d'entreprise échangent des messages. Les microservices sont hébergés dans un ou plusieurs conteneurs collaborant sous l'orchestration de Google Kubernetes (K8S), Docker Swarm ou Apache Mesos. (Voir la Figure I-1).

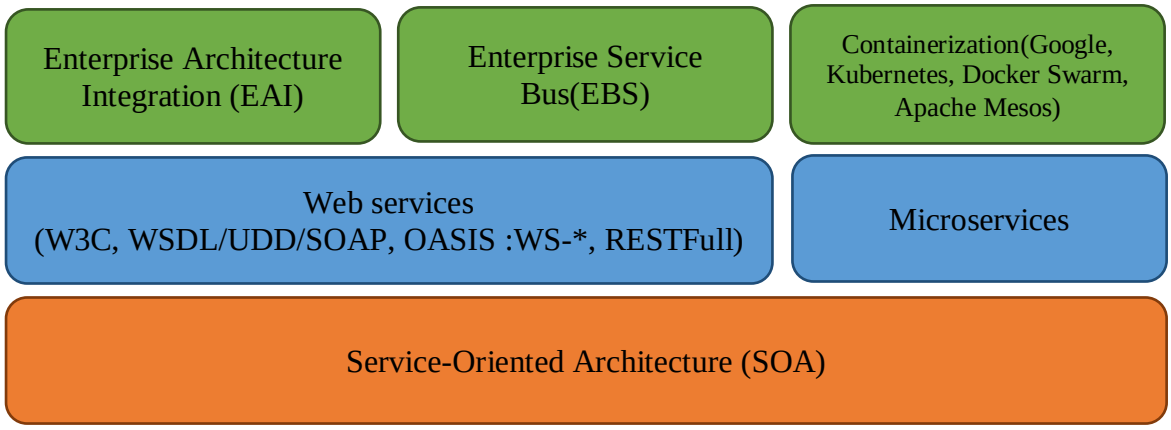

Figure I-1 Architecture orientée services [10]

Une architecture orientée service se conforme à divers principes de gestion des services influençant directement le comportement intrinsèque d’une solution logicielle et le style de sa conception [11]:

- L’encapsulation des services.
- Le faible couplage des services avec la maintenance d’une relation réduisant les dépendances.
- Le contrat de service adhère à un accord de communication, collectivement défini avec un ou plusieurs documents de description.
- L’abstraction des services dissimulant la logique du service à l’extérieur.
- La réutilisation des services partageant la logique entre plusieurs services avec l’intention de promouvoir la réutilisation.

- La composition des services.
- L'autonomie des services.
- L'optimisation des services.
- La découverte des services depuis leur description extérieure.

L'architecture orientée service représente un moyen technique d'intégration des divers systèmes d'information de l'entreprise considérant chaque ressource informatique comme un service. Elle permet de construire les buildings blocks qui composeront l'urbanisme du système d'information [12].

La notion d'interface est importante dans l'approche orientée service. En effet, elle représente le point d'entrée unique vers les fonctionnalités de la solution logicielle et assure la communication grâce à l'échange de messages. Chaque message est porteur de la sémantique particulière à la solution logicielle. De plus ce message est rédigé dans un langage compréhensible aux deux parties en présence. Les services proposés d'une architecture agile décrivent la structure des messages qu'ils attendent du client[12].

L'architecture orientée service est une solution logicielle distribuée. Elle propose un mécanisme d'échange de messages sécurisé entre les systèmes d'informations sous-jacents en employant des protocoles de communication standardisés. Cette approche offre à l'architecture une opportunité d'ouverture sur un large éventail de solutions logicielles existantes[12].

L'architecture orientée services (SOA pour Service Oriented Architecture) est une forme d'architecture de médiation, qui est un modèle d'interaction applicative mettant en œuvre des services (composants logiciels)[13].

- Avec une forte cohérence interne,
- Utilisation d'un format d'échange pivot (le plus souvent XML),
- Couplages externes « lâches » (par l'utilisation d'une couche d'interface interopérable, le plus souvent un service Web).

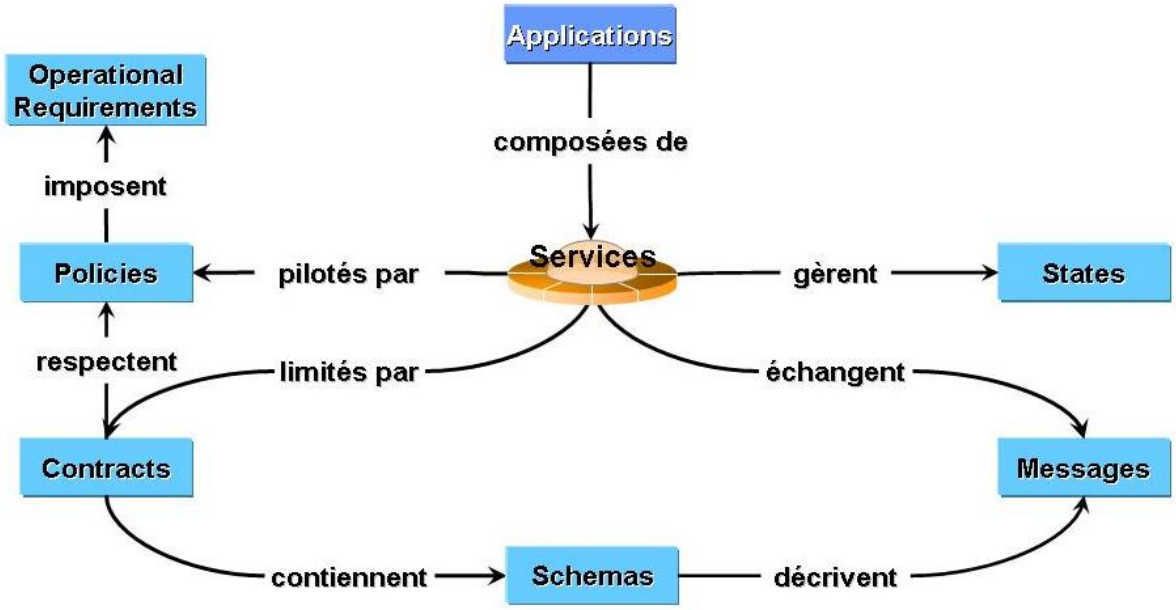

Figure I-2 Concepts de l'architecture SOA

SOA correspond à une démarche architecturale qui vise à rendre les systèmes d'information plus agiles et plus flexibles.

I.2.1 Caractéristiques de SOA

Dans une architecture SOA, comme le montre Figure I-3, les fournisseurs de services publient (ou enregistrent) leurs services dans un annuaire de services (qui peut être publique ou local à un réseau d'entreprise par exemple). Cet annuaire est utilisé par les utilisateurs (i.e. les clients de services) pour trouver des services vérifiant certains critères ou correspondant à une certaine description. Si l'annuaire contient de tels services, il envoie au client les descriptions de ces services avec un contrat d'utilisation. Le client fait alors son choix, s'adresse au fournisseur et invoque le service web [9].

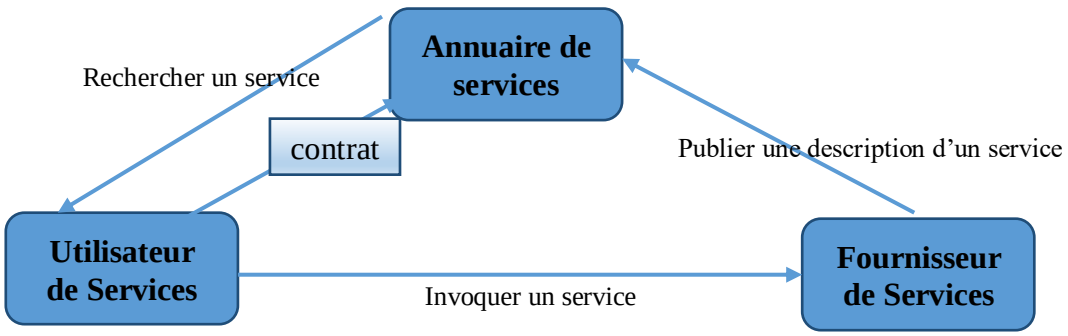

Figure I-3 Le modèle fonctionnel d'une architecture orientée SOA [10]

I.2.2 Les acteurs de SOA

Les services web s'articulent essentiellement autour de trois acteurs principaux illustrés par le schéma précédant :

a. Fournisseur de service : Le fournisseur de service crée le service web, publie son interface et décrit le service en termes de mise en œuvre ainsi que des informations d'accès au service en WSDL dans un registre de services web UDDI et implémente le service dans un serveur d'applications tel qu'apache[14].

b. Utilisateur de services : C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion réseau et en envoyant une demande en XML (REST, XML-RPC, SOAP) [14].

c. Annuaire de services : Le registre de service est un annuaire de services. Le registre fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver.

Dans une architecture SOA, les ressources d'un réseau sont rendues disponibles en tant que services indépendants, auxquels on peut accéder sans rien connaître de la façon dont ils sont réalisés[14].

Cette architecture caractérisée par :

- Un couplage faible entre les services : implique qu'un service n'appelle pas directement un autre service .En effet les interactions sont gérées par une fonction d'orchestration.

- La réutilisation d'un service et plus facile, du fait qu'il n'est pas directement lié aux autres services de l'architecture dans laquelle il évolue[14].

- L'indépendance par rapport aux aspects technologiques : est obtenue grâce aux contrats d'utilisation associés à chaque service. En effet, ces contrats sont indépendants de la plate-forme technique utilisée par le fournisseur du service[14].

- La découverte des services disponibles, afin d'être sûr que les utilisateurs potentiels découvriront le service dont ils ont besoin[14].

• La mise à l'échelle : est rendue possible grâce à la découverte et à l'invocation de nouveaux services lors de l'exécution[14].

I.3 Les services web

Un service web permet à des applications et des systèmes divers de communiquer et d'échanger des données dans des environnements partagés. Cette communication se fait à travers un réseau (Internet) indépendamment des langages de programmation et de plates-formes. Dans ce cadre, via un service web, une application « A » codée en « Java » peut communiquer avec une application « B » codée en « C# »[15].

Autrement dit, un web service (ou service web)est une application appelable via Internet par une autre application d'un autre site Internet permettant l'échange de données (de manière textuelle) afin que l'application appelante puisseintégrer le résultat de l'échange à ses propres analyses. A chaque échange, les requêtes et les réponses sont standardisées et normalisées [16].

Les requêtes et les réponses s'effectuent à l'aide de formats ouverts tels que :

- XML
- JSON
- TEXT

Un service web repose le plus souvent sur le protocole HTTP, mais celui-ci peut également utiliser d'autres protocoles comme le FTP ou le SMTP.

Pour simplifier, un service web fournit des interfaces qui répondent généralement à des demandes HTTP afin de fournir des données venant de différentes sources (base de données ou fichiers)[15] (Voir la Figure I-4) :

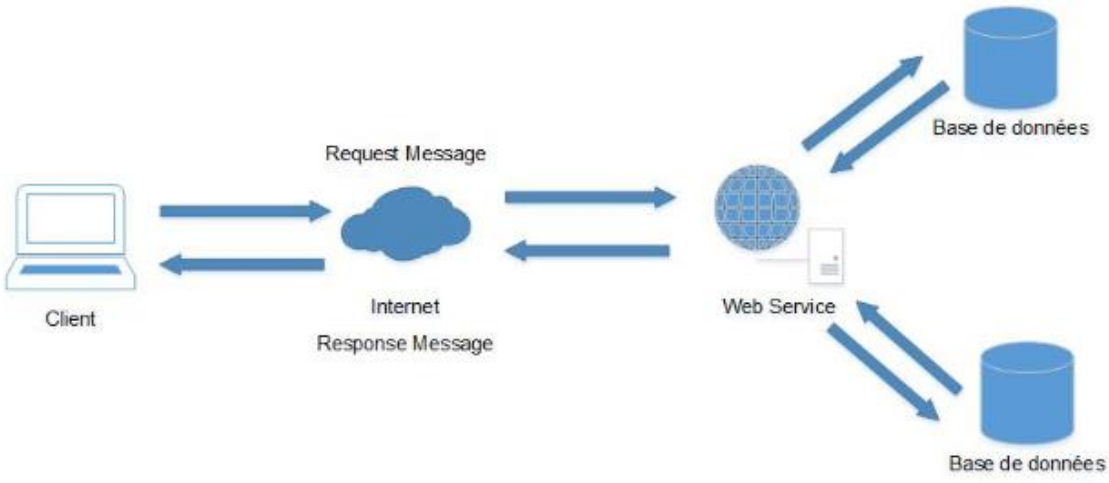

Figure I-4 : Schéma du service web [10]

Principalement utilisé comme un moyen de communication, un service web permet à une entreprise de communiquer avec d'autres entreprises « B2B » et avec leurs clients « B2C » sans prendre connaissance des systèmes d'information se trouvant derrière le pare-feu[15].

I.3.1 Autres définition de services web

Nous pouvons citer plusieurs autres définitions de services web, par exemple, selon W3C, un service web est un composant logiciel identifié par une URL, dont les interfaces publiques et les liaisons sont définies et décrites à l'aide de XML [17]. Il couvre tous les détails nécessaires pour interagir avec le service, y compris les formats de message (qui détaillent les opérations), les protocoles de transport et l'emplacement [18].

Selon[19] :

« Un service web est une agrégation de fonctionnalités publiées pour être utilisées. Il utilise internet comme conduit pour réaliser une tâche. Il est semblable à un processus métier virtuel qui définit des interactions au niveau application »

IBM[20]donne la définition suivante des services web:

« Les services web sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus complexes. Une fois qu'un service web est déployé, d'autres applications peuvent le découvrir et l'invoquer ».

Autrement dit, les services web sont des applications auto descriptives, modulaires et faiblement couplées fournissant un modèle simple de programmation et de déploiement d'applications. Ils reposent principalement sur des technologies basées sur XML pour la structure et le contenu de messages échangés entre services (SOAP), pour la description des services (WSDL) et pour la découverte des services (UDDI).

I.3.2 Les caractéristiques d'un service web

I.3.2.1Basé sur le web

Les services web sont basés sur les protocoles et les langages du web, en particulier HTTP et XML.

I.3.2.2Autodécrit (*Self-described*), Autonome (*self-contained*)

Le cadre des services web contient en lui-même toutes les informations nécessaires à l'utilisation des applications, sous la forme de trois fonctions : trouver, décrire et exécuter.

I.3.2.3 Modulaire (*Modular)*

Les services web fonctionnent de manière modulaire et non pas intégrée. Cela signifie qu'au lieu d'intégrer dans une seule application globale toutes les fonctionnalités, on crée (ou on récupère) plusieurs applications spécifiques qu'on fait interopérer entre elles, et qui remplissent chacune une de ces fonctionnalités. Une fonctionnalité développée sous forme de services web peut dorénavant être réutilisée et recombinée à une suite d'autres fonctionnalités pour composer une nouvelle application.

I.3.3 L'architecture des services web

Les services web reprennent la plupart des idées et des principes du web (HTTP, XML), et les appliquent à des interactions entre machines. Comme pour le World Wide Web, les services Web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Les technologies utilisées par les services Web sont HTTP, WSDL, RES, XML-RPC, SOAP et UDDI.

- **REST:**

REST(Representational State Transfer) est une architecture de services Web. Élaborée en l'an 2000 par Roy Fiedling, l'un des créateurs du protocole HTTP, du serveur Apache HTTPd et d'autres travaux fondamentaux, REST est une manière de construire une application pour les systèmes distribués comme le World Wide Web [21].

REST constitue un style architectural et un mode de communication fréquemment utilisé dans le développement de services Web. Le recours à REST est souvent privilégié par rapport au style SOAP, plus lourd, car REST ne consomme pas autant de bande passante, ce qui rend son utilisation plus pratique sur Internet[21].

- **XML-RPC:**

XML-RPC est un protocole simple utilisant XML pour effectuer des messages RPC (appels à distance dans les réseaux d'ordinateurs). Les requêtes sont écrites en XML et envoyées via HTTP POST. Les requêtes sont intégrées dans le corps de la réponse HTTP. XML-RPC est indépendant de la plate-forme, ce qui lui permet de communiquer avec diverses applications. Par exemple, un client Java peut parler de XML-RPC à un PerlServer[22].

- **SOAP**

SOAP (Simple object Access Protocol) est un protocole standard de communication. C'est

l'épine dorsale du système d'interopérabilité. SOAP est un protocole décrit en XML et standardisé par le W3C. Il se présente comme une enveloppe pouvant être signée et pouvant contenir des données ou des pièces jointes. Il circule sur le protocole HTTP et permet d'effectuer des appels de méthodes à distance[23].

- **WSDL**

WSDL (Services web Description Language) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. WSDL est basé sur XML et permet de décrire de façon précise les détails concernant le service Web tels que les protocoles, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie et les exceptions pouvant être envoyées[23].

L'architecture standard d'un service Web est organisée en plusieurs couches. Chacune d'elles répond à des préoccupations fonctionnelles différentes telles que la publication, la description, la messagerie et le transport. Comme illustré sur la Figure I-5 [24].

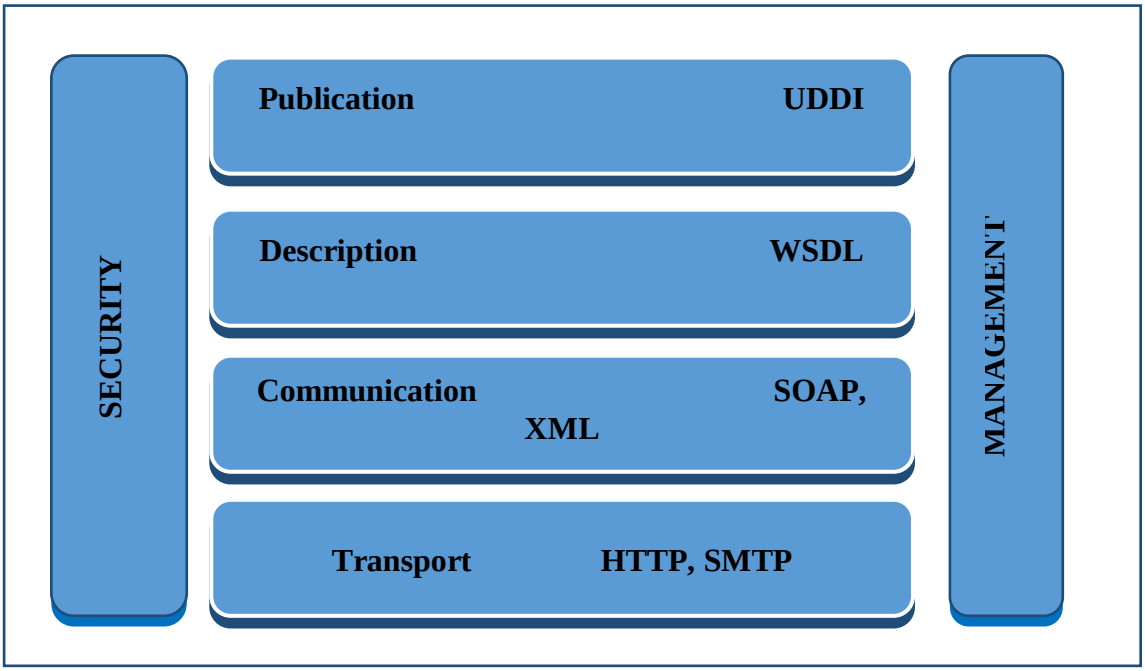

Figure I-5 Architecture du service Web [22]

Les différentes couches de l'architecture d'un service web s'interfacent avec des standards, comme suit :

- La couche de publication : repose sur le protocole UDDI (Universal Description, Discovery and Integration), qui assure le regroupement, le stockage et la diffusion des descriptions des services Web.

- La couche description: est prise en charge par le langage WSDL (Web Service Description Language),qui décrit les fonctionnalités fournies par le service Web, les messages reçus et envoyés pour chaque fonctionnalité, ainsi que le protocole utilisé pour la communication[23].
- La couche communication : la couche de communication des messages propose différents mécanismes liés à l'acheminement des messages (format de communication des messages, adressage, routage...etc.). Cette couche utilise des protocoles reposants sur le langage XML, car sa syntaxe unique résout les conflits syntaxiques lors de l'encodage des données. Actuellement, SOAP (Simple Object Access Protocol) est le protocole le plus utilisé pour cette couche [15].
- La couche transport : le protocole le plus utilisé dans cette couche est l'HTTP (Hyper Text Transfert Protocol). Cependant, d'autres protocoles peuvent être utilisés, tels que le SMTP (Simple Mail Transfer Protocol) ou le FTP (File Transfer Protocol), permettant ainsi aux services web de rester indépendants du mode de transport utilisé[15].

I.4 La composition des services web

La composition de services web est le processus qui consiste à combiner plusieurs services pour satisfaire un besoin complexe de l'utilisateur[26]. Les approches de composition de services web, proposées dans la littérature (voir par exemple [27], [28], [29], [30], [31], [32]), sont diverses mais nous pouvons distinguer deux grandes principales classesles approches manuelles et les approches automatisées[33].

Le processus de composition des services web consiste à créer un service composite offrant une nouvelle fonctionnalité, à partir des services web existants plus simples, par le processus de découverte dynamique, d'intégration et d'exécution de ces services dans un ordre bien défini afin de satisfaire un besoin bien déterminé qui ne pouvait pas être réalisé par un service élémentaire[34].

Autrement dit, la composition de services web est la plus importante fonctionnalité assurée par une architecture SOA. Celle-ci offre un environnement homogène pour la composition dans la mesure où toutes les parties de la composition sont des services idéalement décrits de la même façon et communiquant par les mêmes standards d'échange de messages[33].

I.4.1 Catégories de composition de servicesweb

La classification de la composition de services peut se faire via plusieurs aspects notamment le type de contrôle des services, le degré de l'automatisation, et la gestion des services[35], [36] (Figure I-7).

I.4.1.1 Type de contrôle

Selon le type de contrôle des services, la composition du service peut être réalisée en utilisant l'un des deux paradigmes suivants : l'orchestration de services et la chorégraphie de services [37].

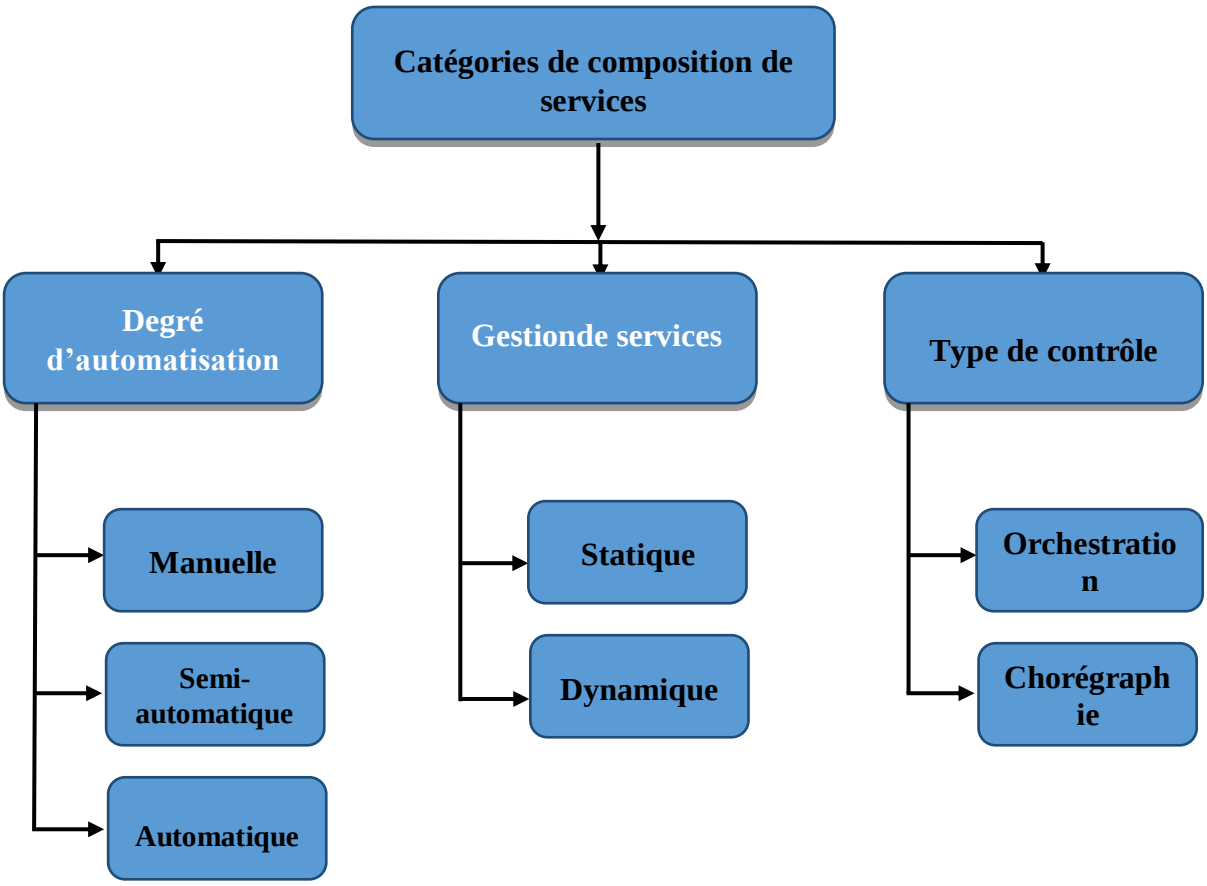

Figure I-6 Classification des méthodes de composition de services

a)Orchestration : L'orchestration de services représente un processus opérationnel centralisé unique (l'orchestrateur) qui coordonne l'interaction entre les différents services. L'orchestrateur est chargé d'invoquer et de combiner les services. Les relations entre tous les services participants sont décrits par un point final unique, c'est-à-dire le service composite. [33].

L'orchestration comprend la gestion des transactions entre les différents services. L'orchestration utilise une approche centralisée pour la composition des services (voir la figure I-7).

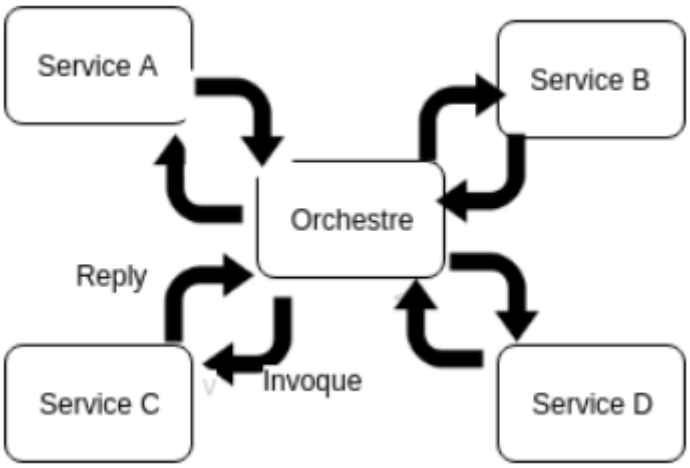

Figure I-7 Orchestration de services [31]

b)Chorégraphie : la chorégraphie de service est une description globale des services participants, qui est définie par l'échange de messages, les règles d'interaction et les accords entre deux ou plusieurs points terminaux. Elle permet à chaque partie impliquée de décrire son rôle dans l'interaction. La chorégraphie utilise une approche décentralisée et collaborative pour la composition des services (voir la figure I-8) [33].

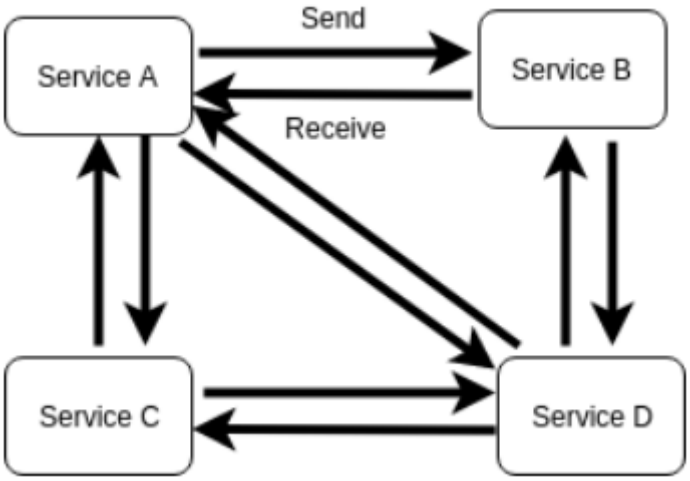

Figure I-8 Chorégraphie de services [31]

En d'autres termes, une chorégraphie décrit les interactions de collaboration entre plusieurs services, alors que l'orchestration représente un contrôle centralisé du point de vue d'une partie. Cela signifie qu'une chorégraphie diffère d'une orchestration en ce qui concerne l'endroit où doit résider la logique qui contrôle les interactions entre les services impliqués [33].

Les figures I.7 et I.8 illustrent ces approches à un niveau supérieur.

I.4.1.2 Gestion des services

Selon la gestion des services, et plus précisément, en fonction du moment au cours duquel les services web sont composés, la composition des services peut être classée selon les deux catégories : la composition statique et la composition dynamique[38].

Plus précisément, la principale différence entre ces deux catégories statique et dynamique dépend du temps pendant lequel un service web concret est intégré dans la composition. Dans la première catégorie, les services requis pour la composition sont choisis, liés, puis déployés au moment de la conception ; tandis que dans la deuxième catégorie la liste des services est sélectionnée dynamiquement lors de la réception d'une requête.

a) La composition statique : Les solutions de composition statiques, sont généralement choisies lorsque les services impliqués dans la composition et les exigences de composition sont statiques, c'est-à-dire qu'ils ne changent pas ou changent rarement, et aussi lorsque les services sont disponibles en permanence. Ainsi, les solutions de compositions statiques ne peuvent pas être utilisées dans un environnement flexible où ils ne sont pas adaptables aux changements tels que l'intégration de nouveaux services par les fournisseurs ou la mise à jour des informations concernant les services disponibles.

b)La composition dynamique : selon Osman et al[39], une composition de services est dite dynamique si les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur. Dans ce type de composition qu'on appelle aussi composition réactive ou on-line, la sélection de services s'effectue « à la volée », suite à l'arrivée d'une requête ; les services sélectionnés sont ensuite reliés pour obtenir un service composite. L'établissement de liaisons entre les services sélectionnés pour la composition dépend des services au moment de l'exécution.

I.4.1.3 Degré d'automatisation

En fonction du niveau d'automatisation, les approches actuelles de composition des services Web peuvent être regroupées en trois catégories : manuelle, automatisée et semi- automatisée.

a) La composition manuelle : dans la composition manuelle, un concepteur humain (administrateur ou fournisseur de services) procède à la composition des services en reliant manuellement les services web. Les travaux de composition manuelle requièrent la définition d'un flux de contrôle (un Workflow abstrait) et d'un flux de données manuellement, soit directement, soit par l'intermédiaire d'outils en utilisant un langage de service Web standard, tel que BPEL (Business Process Execution Language). Ensuite, il relie manuellement les services web aux actions du processus abstrait[40].

Ce type de composition, en plus de nécessiter une intervention humaine constante, prend du temps surtout si le nombre de services est important, et les résultats sont sujets à des erreurs, et il n'y a aucune garantie qu'il réponde à l'exigence de l'utilisateur.

b) La composition automatique : L'automatisation du processus de composition, concerne principalement l'automatisation de création de schéma de composition spécifiant le flux de contrôle et de données entre les différentes actions et l'affectation de services concrets à ces actions ou directement entre les différents services concrets sans passer par un schéma abstrait. Le degré d'automatisation du processus de composition est une dimension importante de toute stratégie de composition de services [33].

c) La composition semi-automatisée : c'est un type de composition où les utilisateurs sont assistés pendant le processus de composition. Dans ce type de composition l'automatisation peut concerner plusieurs aspects, par exemple, l'utilisation d'interfaces graphiques pour aider l'utilisateur à construire le schéma de composition sous la forme d'un Workflow, ou la suggestion de services concrets pour découvrir et sélectionner les services à composer[33]. Cependant, le rôle de l'utilisateur reste central dans la définition du schéma de composition ou dans le choix des services concrets à utiliser.

I.4.2 Langages de composition de services Web

I.4.2.1 BPEL4WS

(Business Process Execution Language for Services web): c'est un standard d'orchestration métiers qui permettent de décrire vos processus métiers sous la forme de services Web, et de spécifier comment ils sont interconnectés afin d'accomplir des tâches particulières.

C'est issu de la fusion de deux langages : WSFL – Web Service Flow Language [41] d'IBM et XLANG[42] de Microsoft. BPEL4WS combine les caractéristiques d'un langage de processus structuré par bloc (XLANG) avec ceux d'un langage de processus basé sur les processus métier(WSFL).

I.4.2.2 WS-CDL

WS-CDL (Web Service Choreography Description Language) est un langage issu des efforts de standardisation du groupe de travail du W3C portant sur la chorégraphie de services Web (Services web Choreography Working Group). L'objectif de ce langage est de décrire les relations entre les services Web lors d'une composition de type chorégraphie[43].

I.4.2.3 OWL-S

OWL-S est un langage de mise en œuvre de services Web sémantiques qui permet la description, la découverte, l'invocation et la composition de type chorégraphie des services

Web. Nous consacrons cette section à la description de leur composition de services Web sémantiques élémentaires[44].

I.5 Conclusion

La technologie des services web permet à des applications de dialoguer à distance via Internet, indépendamment des plates-formes et des langages sur lesquels elles reposent, en s'appuyant sur des protocoles standards.

Dans ce chapitre, nous avons présenté plusieurs éléments de la littérature liés aux services web et à la composition des services web. Dans un premier temps, nous avons étudié l'architecture orientée services (SOA), suivi par l'étude de services web. Dans un deuxième temps, nous avons étudié les différentes solutions ayant proposé des approches pour la composition de services web en citant les différents types de processus de composition dans le but est de préparer le lecteur à bien saisir nos contribution.

Le chapitre suivant abordera le concept de l'optimisation multi-objectifs et les méta-heuristiques.

Chapitre II :

Optimisation multi-objectif et Meta-heuristiques

II Optimisation multi-objectif et Meta-heuristiques

II.1 Introduction

Au cours des dernières décennies, de nombreux algorithmes d'optimisation, y compris exacts et algorithmes approchés, ont été proposés pour résoudre des problèmes d'optimisation.Dans la classe des algorithmes d'optimisation exacte, la conception et la mise en œuvre des algorithmes sont généralement basés sur des méthodes telles que la programmation dynamique, les méthodes de backtracking et branch-and-bound [45]. Cependant, ces algorithmes ont une bonne performance dans de nombreux problèmes [36, 44, 38, 39, 40] mais ils ne sont pas efficaces pour résoudre des problèmes d'optimisation combinatoires et hautement non linéaires à plus grande échelle.

En raison du fait que l'espace de recherche augmente de façon exponentielle avec la taille du problèmeet la recherche exhaustive n'est pas pratique dans ces problèmes. Les méthodes approximatives comme les algorithmes gloutons(Greedy Algorithm) nécessitent généralement de faire plusieurshypothèses qui pourraient ne pas être faciles à valider dans de nombreuses situations [45]. Par conséquent,un ensemble d'algorithmes plus adaptables et flexibles sont nécessaires pour surmonter ceslimites. Sur la base de cette motivation, plusieurs algorithmes généralement inspirés dephénomènes naturels ont été proposés dans la littérature. Parmi eux, certainsdes algorithmes de recherche méta-heuristiques avec un cadre basé sur la population ont montré des capacités satisfaisantes pour traiter des problèmes d'optimisation de grande dimension.

II.2 L’optimisation multi-objectif

L'optimisation multi-objectif (appelée aussi programmation multi-objectif ou optimisation multi-critères) est une branche de l'optimisation mathématique traitant spécifiquement des problèmes d'optimisation ayant plusieurs fonctions objectifs. Elle se distingue de l'optimisation multidisciplinaire par le fait que les objectifs à optimiser portent ici sur un seul problème.

Les problèmes multi-objectifs ont un intérêt grandissant dans l'industrie où les responsables sont contraints de tenter d'optimiser des objectifs contradictoires. Ces problèmes peuvent être NP-difficiles. Leur résolution en des temps raisonnables devient nécessaire et alimente une partie des chercheurs travaillant dans la recherche opérationnelle.

II.2.1 Définition

Dans sa forme la plus générale, un problème d'optimisation multi-objectif consiste à trouver dans un ensemble de solutions admissibles, un sous-ensemble de solutions optimisant(minimisant ou maximisant) ses objectifs. Le cas de la maximisation peut être traité comme une minimisation après inversion des signes de l'expression à maximiser.

Formellement, étant donnés un ensemble de solutions admissibles $X \subseteq R^n$ et $f: R^n \rightarrow R^p$, les p fonctions, le problème d'optimisation multi-objectif consiste à déterminer :.

$$P: \min f(x), x \in X$$

Autrement dit, un problème d'optimisation multi-objectif peut être formulé, d'une façon générale, selon leséquations suivantes :[50]
Minimiser (ou maximiser) :.

$$f_i(X), i = 1, 2, \dots, m$$

Soumis aux contraintes :[50].

$$g_i(X) \geq 0, j = 1,2, \dots, q$$

$$h_k(X) = 0, k = 1,2, \dots, p$$

où m est le nombre de fonctions objectif, $X = [x_1, x_2, \dots, x_n]$ est un vecteur de n variables de décision dont chaque variable x est définie dans les limites supérieure X_i^Uet inferieure X_i^L. Les expressions $g_i(X)\ et\ h_k(X)$sont respectivement des contraintes d'inégalités et d'égalités[50].

Les fonctions objectif du problème d'optimisation forment un espace multidimensionnel appeléespace des fonctions objectif, en plus du traditionnel espace des variables de décision.Le schéma de la Figure II-1 illustre les deux espaces où, pour chaque solution$X = (x_1, x_2, \dots, x_n)$dans l'espacedes variables de décision, il existe un point dans l'espace des fonctions objectif tel que : $F(X) = (f_1(X), f_2(X), \dots, f_n(X))$[50].

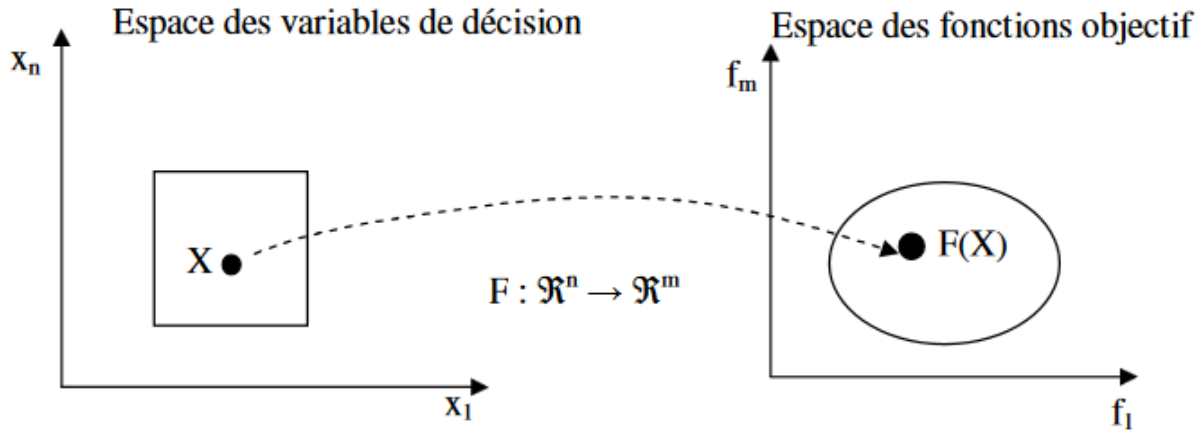

Figure II-1: Représentation d'un problème multi-objectif.[51]

Dans le contexte de l'optimisation multi-objectif, on vise en général :

a)à trouver l'ensemble dessolutions Pareto optimales, c'est-à-dire celles qui couvrent tout le front de Pareto;

b) à s'assurerque les solutions soient suffisamment différentes les unes des autres et qu'elles ne soient pasbiaisées en favorisant un objectif particulier. Les méthodes traditionnelles d'optimisationprésentent beaucoup de lacunes en résolvant un problème multi-objectif à cause des raisons principales suivantes:

- Elles sont incapables de trouver l'ensemble des solutions en une seule exécution de l'algorithme d'optimisation, qui doit alors être exécuté plusieurs fois pour trouver autant de solutions Pareto optimales recherchées;
- L'ensemble des solutions trouvées en suivant cette procédure ne garantit pas que les solutions trouvées soient différentes les unes des autres;
- Elles sont incapables de traiter des problèmes ayant plusieurs solutions optimales (par exemple la méthode d'agrégation des fonctions objectifs).

II.2.5 Ensembles de solutions

L'ensemble des solutions efficaces peut être partitionné en deux ensembles. Le premier contient les solutions dites supportées dont le point image se situe sur la fermeture convexe de la région admissible ; nous pouvons les trouver en résolvant des combinaisons linéaires des objectifs. Le second ensemble contient les solutions non supportées plus complexes à trouver. Elles ne sont pas sur la fermeture convexe, mais ne sont pas dominées pour autant.

Deux solutions admissibles peuvent correspondre à un même point réalisable efficace. Il convient alors de définir l'ensemble de solutions complet minimal qui ne contient qu'une seule

solution pour chaque point non dominé dans l'espace des objectifs et l'ensemble des solutions complet maximal contenant toutes les solutions équivalentes pour chaque point non dominé dans l'espace des objectifs. L'intérêt du premier est de restreindre l'ensemble construit et donc le coût de recherche tout en atteignant tous les points efficaces. L'ensemble maximal permettra à une chaîne de production par exemple de disposer de plusieurs solutions de secours pour un plan de production validé.

La composition du service web peut être modélisée comme une optimisation problème. Ce dernier peut être abordé comme un problème d'optimisation à objectif unique quand nous avons un objectif à optimiser tandis qu'une variante de problème d'optimisation multi-objectif peut être considérée lorsque le problème a plusieurs fonctions objectifs à optimiser.

Un problème d'optimisation de base à objectif unique peut être formulé comme suit [52] :

$$\min f(x), x \in S$$

f est une fonction scalaire et S est l'ensemble des contraintes qui peuvent être définies comme :

$$S = x \in \mathrm{R}^m : h(x) = 0, g(x) \geq 0$$

Une optimisation multi-objectif peut être décrite en termes mathématiques comme suit :

$$\min[f_1(\mathrm{x}); f_2(\mathrm{x}); \ldots; f_n(\mathrm{x})], \mathrm{x} \in S$$

Où n > 1 et S est l'ensemble des contraintes.[52]

L'espace dans lequel l'objectif appartient au vecteur est appelé l'espace objectif, et l'image de l'ensemble des possibles sous f est appelé l'ensemble atteint.

Un tel ensemble sera noté dans la suite par : [52].

$$\mathrm{C} = \{\mathrm{y} \in \mathrm{R}^m : \mathrm{y} = f(x), x \in S\}$$

Les problèmes multi-objectif ont en général un ensemble de solutionsoptimales dont les valeurs des fonctions sont en fait les meilleurs compromis possibles dansl'espace des fonctions objectif. Il faut donc utiliser une autre définition de la "meilleure solution"afin de déterminer exactement quelle solution peut être considérée meilleure par rapport à une autre.

II.2.2 Optimalité des solutions

Pour un problème mon-objectif, la comparaison de deux solutions lors de la recherche de l'optimum est triviale. La définition d'optimalité est à redéfinir dans un contexte multi-objectif ; en effet deux solutions peuvent être incomparables. Par exemple, un problème bi-objectifs ne proposant que deux points réalisables (1,2)et (2,1)d'antécédents distincts ne permet pas d'extraire une solution optimale unique. Il est ainsi nécessaire de laisser à un décideur le verdict final quant au choix de la solution retenue.

Deux définitions de l'optimalité sont envisagées : l'optimalité lexicographique, où un ordre sur les objectifs est fixé et l'optimalité de Pareto, basée sur les notions d'efficacité et de non-dominance, où aucun objectif n'est plus important qu'un autre.

II.2.3 Optimalité lexicographique

Lorsqu'un ordre de priorité ou de préférences est connu sur les objectifs, le problème est résolu d'une scalarisation par somme pondéré ou en optimisant un à un les objectifs dans l'ordre donné. L'idée est alors d'améliorer l'objectif envisagé sans perdre en qualité sur les objectifs préalablement minimisés.

II.2.7 Définition de la relation de dominance

Une solution $x_1$est dite domine une autre solution x_2, et nous écrivons $x_1$domine x_2, si les deux conditions suivantes sont vraies :

1- La solution x_1 n'est pas pire que x_2 dans tous les objectifs.

2- La solution x_1 est strictement meilleure quex_2 dans au moins un objectif.

Pour mieux comprendre de"dominance de Pareto". Soit deux vecteurs U et V dans l'espace des fonctions objectif où unproblème de minimisation est considéré. On dit que le vecteur $U = (u_1, u_1, \dots, u_m)$domine levecteur$V = (v_1, \dots, v_m)$, si et seulement si toutes les composantes de U sont inférieures ouégales à celles correspondantes dans V, et au moins une composante de U est strictementinférieure à celle correspondante dans V. Le principe de la dominance de Pareto est illustré à la Figure II-2.

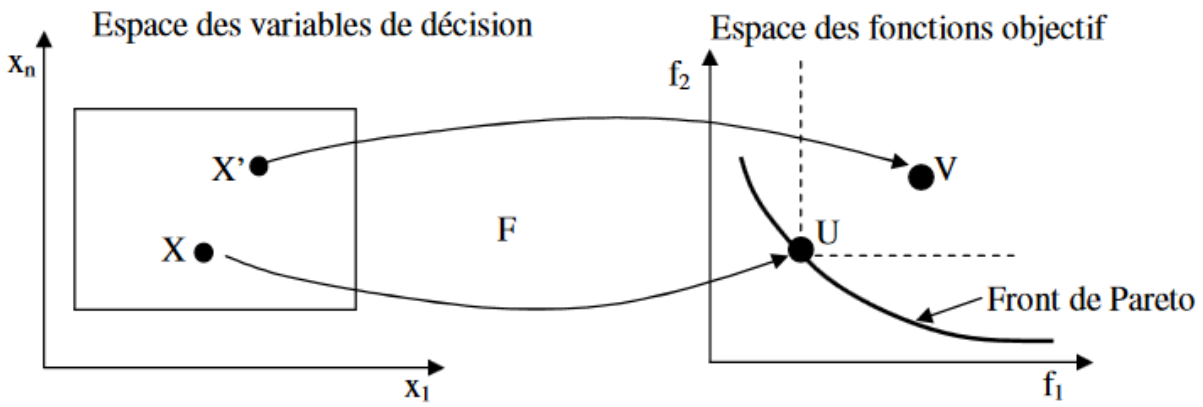

Figure II-2: Dominance de Pareto et optimalité de Pareto

II.2.4 OptimalitédePareto

S'il est impossible de connaître a priori un ordre préférence sur les objectifs, de nombreuses solutions deviennent incomparables entre elles. La première référence à traiter d'objectifs conflictuels est fréquemment attribuée à Vilfredo Pareto en 1896 [53]. Il définit l'efficacité comme l'impossibilité d'améliorer un objectif de la solution sans dégrader l'intérêt d'un autre objectif. L'importance fondamentale de l'efficacité est basée sur ce constat qu'une solution s'avère inefficace s'il existe une solution qui lui est préférée. Un tel ne peut pas représenter une alternative valide pour un décideur[53].

Le concept de "l'optimalité de Pareto" est ainsi utilisé pour établir unehiérarchie entre les solutions d'un problème multi-objectif en vue de déterminer si une solutionappartient réellement à l'ensemble des meilleurs compromis[53].

Figure II-3: Solutions dominées et non dominées dans l'espace objectif[54].

Par ailleurs, une solution est dite Pareto-optimale (optimale de Pareto), si aucune des fonctions objectifs ne peut être améliorée sans dégrader certaines des autres valeurs objectives.

L'ensemble des solutions optimales est appelé"ensemble Pareto optimal", et l'ensemble des valeurs des fonctions objectif correspondantes dansl'espace des fonctions objectif est appelé "front de Pareto". Selon les problèmes à traiter, le frontde Pareto peut avoir une configuration très complexe (e.g., continuité, discontinuité, convexité,disjonction etc.).
La notion d'optimalité de Pareto peut être décrite comme suit : Considérons un vecteur $x^* \in S$. Cet ensemble est dit optimal de Pareto pour un problème multi-objectif si tous les autres vecteurs $x \in S$ ont une valeur supérieure pour au moins un des fonctions objectifs f_i avec $i = \{1, \ldots, n\}$ ont la même valeur pour toutes les fonctions objectifs. [4]
Le concept de solution prédominant dans la définition de solutions pour les problèmes d'optimisation multi-objectifs est celui d'optimalité de Pareto.

Autrement dit, considérons un point x* dans l'espace de conception réalisable S est Pareto optimal si et seulement s'il n'existe pas d'autre point x dans l'ensemble S tel que: $f(x) \leq f(x^*)$avec au moins une$f_i(x) > f_i(x^*)$ (voir la Figure II-3).

Notez que les inégalités entre les vecteurs s'appliquent à chaque composante de chaque vecteur ; par exemple : $f(x) \leq f(x^*)$implique : $f_1 \leq f_1^*, f_2 \leq f_2^*$, etc. L'ensemble de tous les points optimaux de Pareto est appelé l'ensemble optimal de Pareto, et ce terme peut faire référence à des points dans l'espace de conception ou à des points dans l'espace des critères. La définition ci-dessus signifie que pour que x^*soit appelé le point optimal de Pareto, aucun autre point n'existe dans l'espace de conception réalisable S qui améliore au moins une fonction objectif tout en gardant les autres inchangées.
Dans notre travail, nous nous concentrerons sur la recherche de solutions Pareto-optimales en utilisant la relation de dominance.

II.3 Les méta-heuristiques :

Les heuristiques [55] sont des méthodes utilisées pour trouver une bonne (presque) solutions à un coût de calcul raisonnable sans garantir la faisabilité ou optimalité. En d'autres termes, les méta-heuristiques sont un ensemble de stratégies intelligentes pour améliorer l'efficacité des procédures heuristiques. Laporte et Osman [56]ont défini une méta-heuristique comme : « Un processus de génération itératif qui guide un subordonné heuristique en combinant intelligemment différents concepts pour explorer et exploiter l'espace de recherche, des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions quasi-optimales.

Une méta-heuristique d'après Voss et al. [57], est : « un processus maître itératif qui guide et modifie les opérations des heuristiques subordonnées pour produire efficacement des solutions de haute qualité. Il peut manipuler une solution unique complète (ou incomplète) ou un ensemble de solutions par itération. Les heuristiques subordonnées peuvent être des procédures de haut (ou bas) niveau, ou une simple recherche locale, ou juste une méthode de construction.
La majorité de ces algorithmes ont un comportement stochastique et imitent des processus biologiques ou processus physiques.
Différentes catégories ont été considérées pour classer les algorithmes méta-heuristiques jusqu'à présents. Dans la section suivante, nous allons décrire leur classification selon différents points de vue.

II.3.1 Nomenclature & Objectif

Les mots « méta » et « heuristique » sont grecs, où « méta » signifie « niveau supérieur » ou « au-delà » et heuristique signifie « trouver », « savoir », « orienter une enquête » ou « à découvrir » [58].

Le but d'une méta-heuristique est de résoudre un problème d'optimisation donné : elle cherche un objet mathématique (une permutation, un vecteur, etc.) minimisant (ou maximisant) une fonction objectif, qui décrit la qualité d'une solution au problème.

L'ensemble des solutions possibles forme l'espace de recherche. L'espace de recherche est au minimum borné, mais peut être également limité par un ensemble de contraintes.

Les méta-heuristiques manipulent une ou plusieurs solutions, à la recherche de l'optimum, la meilleure solution au problème. Les itérations successives doivent permettre de passer d'une solution de mauvaise qualité à la solution optimale. L'algorithme s'arrête après avoir atteint un critère d'arrêt, consistant généralement en l'atteinte du temps d'exécution imparti ou en une précision demandée.
Une solution ou un ensemble de solutions est parfois appelé un état, que la métaheuristique fait évoluer via des transitions ou des mouvements. Si une nouvelle solution est construite à partir d'une solution existante, elle est sa voisine. Le choix du voisinage et de la structure de donnée le représentant peut être crucial.

Lorsqu'une solution est associée à une seule valeur, nous parlons de problème mono-objectif, lorsqu'elle est associée à plusieurs valeurs, de problème multi-objectifs (ou multi-critères). Dans ce dernier cas, nous recherchons un ensemble de solutions non dominées (le « front de Pareto »), solutions parmi lesquelles nous ne pouvons décider si une solution est meilleure qu'une autre, aucune n'étant systématiquement inférieure aux autres sur tous les objectifs.

Dans certains cas, le but recherché est explicitement de trouver un ensemble d'optimums « satisfaisants ». L'algorithme doit alors trouver l'ensemble des solutions de bonne qualité, sans nécessairement se limiter au seul optimum : nous parlons de méthodes multimodales ou multi-objectifs.

II.3.2 Classification des algorithmes méta-heuristiques

Différentes manières basées sur les caractéristiques sélectionnées ont été proposées pour classer les méta-heuristiques comme le montre la Figure II-5 [59]. Cette section résume brièvement les classes les plus importantes, y compris les inspirées de la nature contre les non-inspirées de la nature, basées sur la population par rapport à la recherche à point unique, dynamique par rapport à la fonction objectif statique, voisinage unique contre voisinage à diverses structures, et utilisation de la mémoire par rapport aux méthodes sans mémoire [48,, 50]

Le diagramme de la FigureII-4 tente de présenter où se placent quelques-unes des méthodes les plus connues. Un élément présenté à cheval sur différentes catégories indique que l'algorithme peut être placé dans l'une ou l'autre classe, selon le point de vue adopté.

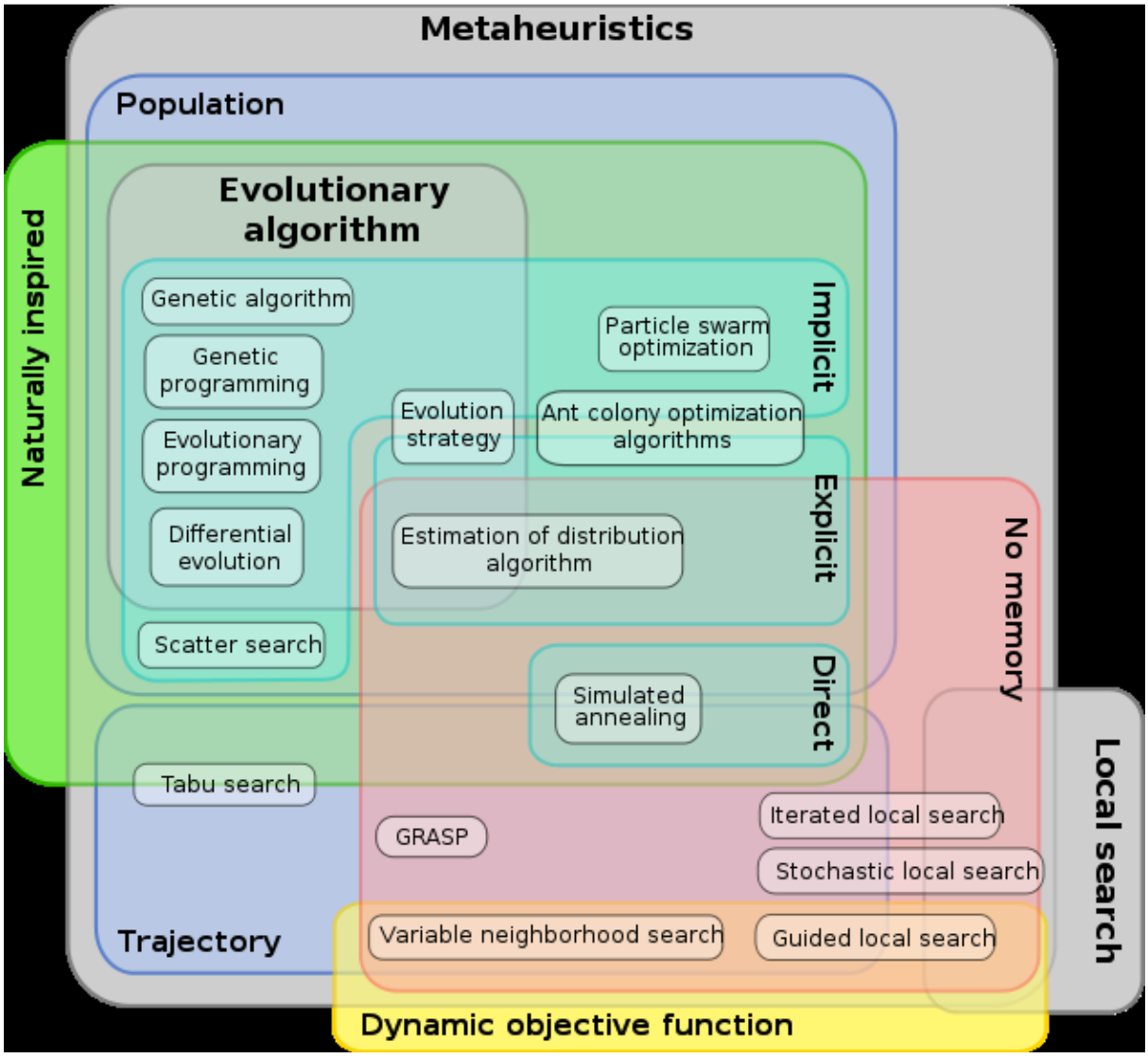

Figure II-4: Classification des algorithmes méta-heuristiques [59]

II.3.2.1 L’inspiration de la nature contre la non-inspiration de la nature

Cette classe est basée sur l'origine de l'algorithme. La majorité des méta-heuristiques sont des algorithmes inspirés de la nature tels que Ant Colony Optimization (ACO), Particle Optimisation des essaims (PSO), algorithmes génétiques (GA) algorithmes mémétiques (MA). Aussi, certains d'entre eux sont des algorithmes non inspirés de la nature comme Iterated Local Search (ILS) [62] et Tabu Recherche (TS) [63].

II.3.2.2 La recherche basée sur la population par rapport à un point unique

Les méta-heuristiques peuvent également être classées selon le nombre de solutions utilisées en même temps. Les méthodes de trajectoire, comme illustré à la Figure II-5, sont les algorithmes travaillent sur la base d'une solution unique à tout moment et englober les méta-heuristiques de recherche locale telles que TS, ILS et Variable Neighborhood Search (VNS) [64].

Les algorithmes basés sur la population effectuent une recherche avec plusieurs points initiaux dans un style parallèle comme les méta-heuristiques basées sur les essaims (swarm).

Les algorithmes basés sur les essaims sont constitués de simples particules ou agents interagissant localement entre eux et avec leur environnement [65]. Chaque particule suit une ou plusieurs règles sans aucune structure centralisée pour contrôler son comportement. Par conséquent, les interactions locales et aléatoires entre les particules conduisent à un comportement global intelligent. Ces algorithmes appliquent deux approches pour obtenir un bon fonctionnement[66]: exploration globale et exploitation locale.

L'exploration est la puissance de l'expansion de l'espace de recherche, alors que l'exploitation est la capacité de trouver la solution optimale autour d'une bonne (quasi-)solution optimale. En raison d'une connaissance insuffisante de l'espace de recherche dans les étapes initiales, plus d'exploration doit être effectuée par l'essaim pour éviter le piégeage dans les optimums locaux. En revanche, plus d'exploitation est requise par laps des étapes, de sorte que l'algorithme est capable de s'ajuster sur des points semi-optimaux. En d'autre, un compromis approprié entre l'exploration et l'exploitation est nécessairepour avoir une recherche efficace[63].

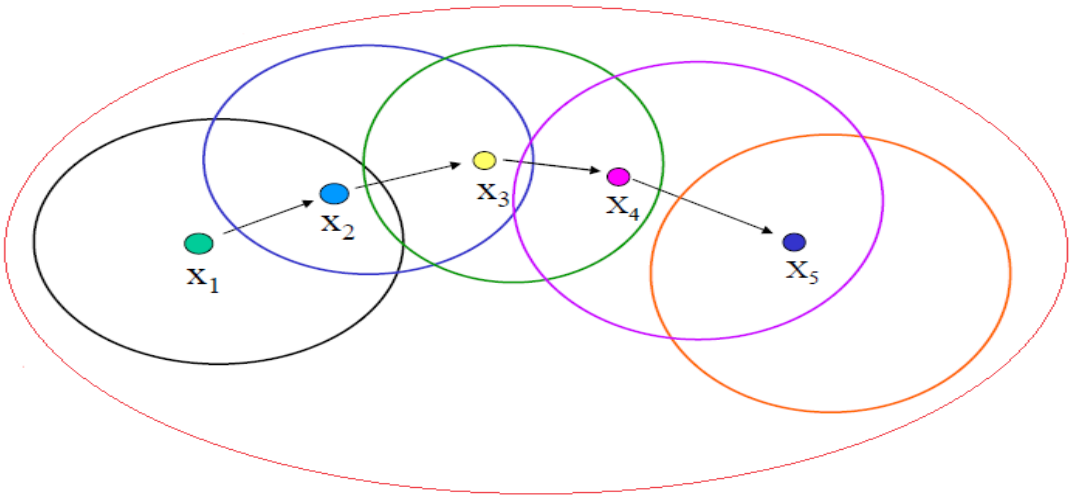

Figure II-5: La méthode à base trajectoire

II.3.2.3Fonction objectif dynamique contre statique

Une autre caractéristique qui peut être employée pour les méta-heuristiques la classification est la manière d'utiliser la fonction objectif. En d'autres termes, le la fonction objectif est conservée « telle quelle » dans la représentation du problème par certains tandis que d'autres, comme la recherche locale guidée (GLS) [67], la modifient pendant la recherche. L'idée derrière cette approche est d'échapper aux optima locaux en changer le paysage de la recherche. Par conséquent, la fonction objectif est modifiée par incorporant les informations recueillies au cours du processus de recherche.

II.3.2.4Nombre de structures de voisinage

La majorité des algorithmes méta-heuristiques appliquent un seul voisinage structure. En d'autres termes, la topologie du paysage de fitness ne change pas dans le cours de l'algorithme tandis que d'autres, comme Variable Neighborhood Search (VNS), emploient un ensemble de structures de voisinage. Cette dernière structure offre la possibilité pour diversifier la recherche en permutant entre différents paysages de fitness [68].

II.3.2.5 Emploi de mémoire

L'utilisation de la mémoire est l'une des caractéristiques les plus importantes pour classer les méta-heuristiques. En d'autres termes, l'utilisation de la mémoire est connue comme l'un des fondamentaux éléments d'une puissante méta-heuristique. Les algorithmes sans mémoire effectuent un processus de Markov, car l'information qui est utilisée pour déterminer l'action suivante est l'état actuel du processus de recherche. Il existe plusieurs façons d'utiliser la mémoire[68].

En outre, l'utilisation de la mémoire à court terme est généralement différente de la mémoire à long terme. La première garde généralement une trace des mouvements récemment effectués, des solutions visitées ou, en général, décisions prises. La seconde est généralement une accumulation de paramètres synthétiques à propos de la recherche.
Dans les chapitres suivants, nous allons présenter nos trois contributions en commençant par une recherche locale stochastique pour le Problème de composition des services, puis une approche de recherche à voisinage variable, et à la fin, une approche mémétique[68].

II.3.2.6Implicite, explicite, directe

En considérant les méta-heuristiques comme des méthodes itératives utilisant un échantillonnage de la fonction objectif comme base d'apprentissage (définition plus particulièrement adaptée aux méta-heuristiques à populations) apparaît le problème du choix de l'échantillonnage.
Dans la très grande majorité des cas, cet échantillonnage se fait sur une base aléatoire, et peut donc être décrit via une distribution de probabilités. Il existe alors trois classes de méta-heuristiques, selon l'approche utilisée pour manipuler cette distribution.

La première classe est celle des méthodes implicites, où la distribution de probabilité n'est pas connue a priori. C'est le cas par exemple des algorithmes génétiques, où le choix de l'échantillonnage entre deux itérations ne suit pas une loi donnée, mais est fonction de règles locales. Par opposition, nous pouvons donc classer les méthodes explicites, qui utilisent une distribution de probabilité choisie à chaque itération. C'est le cas des algorithmes à estimation de distribution.

Dans cette classification, le recuit simulé occupe une place particulière, puisqu'on peut considérer qu'il échantillonne la fonction objectif en utilisant directement celle-ci comme distribution de probabilité (les meilleures solutions ayant une probabilité plus grande d'être tirées). Il n'est donc ni explicite ni implicite, mais plutôt « direct ».

II.3.2.7Évolutionnaire ou non

On trouve parfois une classification présentant les algorithmes d'optimisations stochastiques comme étant « évolutionnaires » (ou « évolutionnistes ») ou non. L'algorithme sera considéré comme faisant partie de la classe des algorithmes évolutionnaires s'il manipule une population via des opérateurs, selon un algorithme général donné.
Cette façon de présenter les méta-heuristiques dispose d'une nomenclature adaptée : nous parlerons d'opérateurs pour toute action modifiant l'état d'une ou plusieurs solutions. Un

opérateur construisant une nouvelle solution sera dénommé générateur, alors qu'un opérateur modifiant une solution existante sera appelé mutateur.
Dans cette optique, la structure générale des algorithmes évolutionnaires enchaîne des étapes de sélection, de reproduction (ou croisement), de mutation et enfin de remplacement. Chaque étape utilise des opérateurs plus ou moins spécifiques.

II.3.2.8Taxinomie de l'hybridation

Nous parlons d'hybridation quand la méta-heuristique considérée est composée de plusieurs méthodes se répartissant les tâches de recherche. La taxinomie des méta-heuristiques hybrides se sépare en deux parties : une classification hiérarchique et une classification plate. La classification est applicable aux méthodes déterministes aussi bien qu'aux méta-heuristiques [69].

La classification hiérarchique se fonde sur le niveau (bas ou haut) de l'hybridation et sur son application (en relais ou concurrente). Dans une hybridation de bas niveau, une fonction donnée d'une méta-heuristique (par exemple, la mutation dans un algorithme évolutionnaire) est remplacée par une autre méta-heuristique (par exemple une recherche avec tabou). Dans le cas du haut niveau, le fonctionnement interne « normal » des méta-heuristiques n'est pas modifié. Dans une hybridation en relais, les méta-heuristiques sont lancées les unes après les autres, chacune prenant en entrée la sortie produite par la précédente. Dans la concurrence (ou coévolution), chaque algorithme utilise une série d'agents coopérants ensembles.

Cette première classification dégage quatre classes générales :

- bas niveau et relais (abrégé LRH en anglais),
- bas niveau et coévolution (abrégé LCH),
- haut niveau et relais (HRH),
- haut niveau et coévolution (HCH).

La seconde partie dégage plusieurs critères, pouvant caractériser les hybridations :

- si l'hybridation se fait entre plusieurs instances d'une même méta-heuristique, elle est homogène, sinon, elle est hétérogène ;
- si les méthodes recherchent dans tout l'espace de recherche, nous parlerons d'hybridation globale, si elles se limitent à des sous-parties de l'espace, d'hybridation partielle ;

- si les algorithmes mis en jeu travaillent tous à résoudre le même problème, nous parlerons d'approche générale, s'ils sont lancés sur des problèmes différents, d'hybridation spécialisée.

Ces différentes catégories peuvent être combinées, la classification hiérarchique étant la plus générale.

II.4Les méta-heuristiques utilisées dans notre travail de thèse

Dans cette partie, nous allons présenter certaines méta-heuristiques que nous avons utilisées dans notre travail de thèse : la méthode de recherche locale, l'algorithme génétique, l'algorithme de recherche locale stochastique & l'algorithme mémétique.

II.4.1 La méthode de recherche locale

Le principe des méthodes de recherche locale (ou méthodes d'amélioration itérative)est inspirée des méthodes d'optimisation continue. Ces méthodes consistent à déterminer itérativement la solution d'une fonction continue en utilisant des outils comme les dérivées partielles ou les gradients, suivant que la fonction soit ou non dérivable[1].

La description de ces méthodes à partir d'une solution de départ X_0, à engendrer une suite (_nie) de solutions (X_n), déterminées de proche en proche, c'est-à-dire itérativement(X_{i+1} étant déterminée à partir de X_i). Le choix de la solutionX_{i+1} se faitdans un ensemble _localement proche_ de la solution $X_i$et de manière à avoir unesolution X_{i+1}''plus intéressante'' au sens de la recherche locale. Les deux expressionsmises entre guillemets dans la phrase précédente se formalisent à partir de la notion devoisinage et de recherche dans un voisinage[67].

La construction essentielle de ces méthodes repose en effet sur la déterminationd'un bon voisinage. Ensuite, suivant la façon de choisir une solution dans le voisinage, on obtient différentes méthodes de recherche locale : méthode tabou, descente ''pure'',descente stochastique, recuit simulé,...

II.4.2 La méthode de recherche locale stochastique :

Les méthodes stochastiques, contrairement à la plupart des méthodes déterministes,ne nécessitent ni point de départ, ni à la connaissance du gradient de la fonction objectifpour atteindre la solution optimale. Elles s'appuient sur des mécanismes de transitionprobabilistes et aléatoire qui explorent efficacement l'espace de recherche et convergentvers l'optimum

global. Leur nature aléatoire implique que plusieurs exécutionssuccessives de ces méthodes conduisent à des résultats différents pour une mêmeinitialisation du problème d'optimisation.

II.4.2.1 Structure des méthodes SLS :

Sur la base des éléments définis précédemment, un algorithme SLS fonctionne comme l'illustre la Figure II-6. Lorsqu'il est appliqué à des problèmes d'optimisation, dont la définition comprend une fonction objectif qui spécifie la qualité des solutions, un algorithme SLS nécessité de garder une trace de la meilleure solution rencontrée pendant le processus de recherche. Ce dernier est généralement arrête dès qu'une solution pour l'instance du problème donnée est trouvée.

ALGORITHMEII.1Méthode de recherche locale stochastique [59].
1 : Détermine l'état initial de la recherche
2 : **tant que**le critère de terminaison n'est pas satisfait **faire**
3: Effectuer une étape de recherche
4 : Si nécessaire, mettre à jour la solution actuelle.
5 : **fin tant que**
6 : **Renvoie**S

L'étape de recherche consiste à exécuter les opérations de voisinage en basant sur la solution courante et un paramètre probabiliste dont l'objectif est de trouver une solution candidate qui sera prise comme nouvelle solution courante. Pour mettre à jour la solution courante, il existe deux moyens possibles, soit nous choisissons aléatoirement une solution voisine, soit nous sélectionnons la voisine qui donne la meilleure valeur de la fonction objectif.

II.4.2.2 Concepts & définitions

Le concept de la recherche locale stochastique, tel que défini ci-dessus, fournit un cadre unificateur pour de nombreux types d'algorithmes. En particulier, il capture des algorithmes constructifs, dont l'espace de recherche contiennent des solutions partielles d'un problème donné, c'est à dire, des solutions candidates qui peuvent être étendues en ajoutant des composants de la solution (comme les arêtes dans le cas du TSP). Par conséquent, les algorithmes qui sont fortement basés sur les procédures de recherche de manière constructive, comme le GRASP ou d'optimisation de colonie de fourmis, entrent dans le cadre SLS général. De même, les algorithmes basés sur la population, qui opèrent sur des ensembles de solutions

candidates, se conforment par la définition générale de SLS en tenant compte des positions de recherche se composent d'ensembles de solutions candidates individuelles[1].

II.4.2.3L'amélioration itérative

L'amélioration itérative est la méthode SLS la plus élémentaires. Elle est basée sur l'idée d'améliorer itérativement une solution candidate du problème posé relativement à une fonction d'évaluation. Plus précisément, la recherche est démarré à partird'une position (solution) initiale, et à chaque étape de recherche, la solution actuellement candidatenotée s est remplacée par une solution candidate voisine s'avec une valeur de la fonction d'évaluation[1].

II.4.2.4Méthodes SLS simple

Les méthodes décrites dans ce chapitre sont dites "simples" dans le sens où elles sont généralement fondées sur une seule relation de voisinage fixe.

a) Algorithme d'amélioration itérative RII :

Les algorithmes RII (Randomised Iterative Improvement) constituent une versionles plus simples pour permettre à l'aggravation des mesures de recherche et d'aller parfois à un choix au hasard vers une position voisine de recherche. En RII, cette idée est mise en œuvre par commutation probabiliste entre deux types d'étapes de recherche dans une même structure de voisinage : d'étapes d'amélioration itérative et d'étapes de recherches aléatoires non-informées de mesures, dans lequel un voisin est choisi uniformément au hasard. Plus précisément, à chaque étape de recherche, une étape de marche aléatoire uniforme est effectuée avec une probabilité **wp**, et une démarche d'amélioration itérative est réalisée autrement (c'est à dire avec une probabilité 1 - wp). Le paramètre wp est appelé probabilité de recherche ou un paramètre de bruit[1].

Grâce à ce mécanisme de longues séquences d'étapes de recherche aléatoire peuvent être effectuées arbitrairement où la probabilité de r étapes consécutives de recherche aléatoire estp_r.

Une extension d'un algorithme d'amélioration itérative dans un algorithme RII exige habituellement que quelques lignes de code et ne possède qu'un seul paramètre. En dépit de leur simplicité conceptuelle, les algorithmes RII peuvent être efficaces. Par exemple, ils ont fourni la base pour GSAT avec la marche aléatoire, un algorithme bien connu pour le problème de satisfiabilité propositionnelle (SAT) [70]. Il existe cependantpeu d'algorithmes RII, peut-être parce que d'autres algorithmes SLS plus complexes atteignent souvent de meilleures performances.

b) Algorithme probabiliste d'amélioration itérative (PII) :

Les étapes de recherche aléatoire, telle que pratiquée dans RII, permettent de conduire à unedétérioration (cas de minimisation) significative de la valeur de la fonction d'évaluation. Une alternative intéressante consiste à fonder l'acceptation d'une aggravation de l'étape de recherche sur le changement de fonction d'évaluation qu'elle a causé, ce qui constitue l'idée clé derrière l'amélioration itérative probabiliste (PII). A chaque étape, l'algorithme PII sélectionne une solution candidate voisine selon une fonction P(g, S), qui détermine une distribution de probabilité sur le voisinage de S en tenant compte de la fonction d'évaluation g[1].

Dans la pratique, des échantillons de cette distribution de probabilités peuvent être prises comme suit: d'abord, un voisin nommé s0 de la position de recherche actuellenommé s est sélectionné au hasard, puis une décision est prise si s0 est acceptée comme la nouvelle position de recherche. En minimisant g, cette décision est dans de nombreux cas sur la base des faits suivants, bien connus par : la condition Metropolis:

1 **si** $g(s') < g(s)$

Exp $((g(s) - g(s'))/T)$ **sinon**

Où T est un paramètre qui détermine le degré d'aggravation des mesures de recherche qui sontsusceptibles d'être acceptées. La condition de Metropolis est fréquemment utilisée dans les algorithmes de recuit simulé (voir ci-dessous) et dans ce contexte le paramètre T est appelé température [69, 70].

II.4.2.5Les méthodes SLS Hybrides

a) Recherche locale itérative (ILS)

L'idée clé derrièrecette méthode SLS est d'échapper de l'optimum localactuel de la fonction d'évaluation donnée par le moyen d'un mécanisme de perturbation.Essentiellement, trois composants sont au cœur de tout algorithme de recherche locale itératif (ILS) [17].

- Uneprocédure filiale de recherche locale est utilisée de façon efficace pour trouver l'optimum local, ce qui est généralement basée sur un algorithme d'amélioration itérative ou d'uneméthode SLS simple.
- La procédure de perturbation qui introduit une modification à une solutioncandidate donnée afin de permettre au processus de recherche de s'échapper de l'optimum local.

- Un critère d'acceptation est utilisé pour décider si la recherche doit être nouvellement poursuivie à partir d'un optimum localtrouvé.

Sur la base de ces composants, l'algorithme ILS fonctionne comme décrit dans la Figure. II-7.

ALGORITHMEII.2Méthode deRecherche locale réitérée (IlS) [1].
1 : Détermine une solution candidate initiale s
2 : Effectue la recherche filiale locale sur s
3 : **tant que**le critère de terminaison n'est pas satisfait **faire**
4 : r := s
5 : Effectue la perturbation sur s
6 : Exécute la filiale de recherche locale sur s
7 : En se basant sur le critère d'acceptation :
8 : Maintenir s ou revenir à s := r
9 : **fin tant que**

Lors de l'utilisation d'un algorithme d'amélioration itérative comme une procédure filiale de recherche locale, les ILS peuvent être considérés comme une démarche aléatoire biaisée dans l'espace des optimums locaux atteint par cette procédure. La procédure de perturbation devrait introduire une modification qui ne peut être immédiatement annulée par la procédure de recherche locale, afin de parvenir à une diversification effective de la recherche. Le critère d'acceptation détermine le degré d'intensification de recherche. Par exemple, si seulement l'amélioration des solutions candidates qui sont acceptées, l'ILS exécute la recherche de la première-amélioration aléatoire dans l'espace de l'optimum local. En résumé, la procédure de perturbation et le critère d'acceptation détermine l'équilibre entre l'intensification et de diversification de recherche.

Typiquement, l'attraction principale de l'ILS est le fait qu'elle est en général très facile pourfaire des extensions des réalisations existantes d'un algorithme de recherche local simple dans un algorithme ILS [1].

b) Algorithmes Itératif Greedy (IG) :

Les méthodes de construction Greedysont au cœur de nombreux algorithmes d'approximation bien connus. Ils fournissent également la base pour les algorithmes itératifsGreedy, qui peuvent être considérés comme une variante de l'ILS dans lequel la recherche locale et les phases de perturbation sont remplacées par les phases de construction et de destruction. A partir d'une solution candidate complète, IG alterne entre des phases de destruction, durant

lesquelles certains composants de la solution sont retirés de la solution candidate actuels s (par exemple, au hasard ou par l'utilisation d'une heuristique, en fonction de leur impact sur la fonction d'évaluation), et les phases de construction, au cours desquelles les composants des solutions sont ajoutés heuristiquement, jusqu'à ce qu'une nouvelle solution complète **s'** candidat ait été obtenue. Comme dans ILS, un critère d'acceptation est utilisé pour décider s'il convient de poursuivre la recherche de **s'**ou **s**.

On note que la solution candidate initiale en IG peut être générée par une méthode de construction qui peut être différente de celle utilisée dans les phases de construction ultérieures. En outre, une procédure de recherche locale perturbatrice peut être utilisée pour améliorer encore les solutions candidates complètesconsidérées durant la recherche. Dans ce cas, IG peut être considéré comme une variante de l'ILS dans laquelle on combine une phase destruction et une phase de construction des formes de la procédure de perturbation.

Les méthodes IG constituent une approche prometteuse pour résoudre les problèmes pour lesquels les bons algorithmes constructifs existent[1].

c) Procédures de recherche randomisée adaptative Greedy (GRASP):

Une combinaison de la recherche locale constructive et perturbatrice constitue la base pour le GRASP [38, 39], qui fonctionne comme suit: En utilisant une procédure randomisée de construction, une solution candidate complète est générée et qui est ensuite améliorée en utilisant une procédure perturbatrice de recherche locale. Ce processus à deux phases est répété jusqu'à ce qu'un critère de terminaison soit satisfait.

La procédure de construction dans le GRASP ajoute itérativement des composants de la solution qui sont choisis au hasard à partir d'une liste restreintede candidats. Les éléments de cette liste sont déterminés en utilisant une fonction heuristique, dont la valeur d'une composante donnée peut dépendre de composants de lasolution déjà présente dans la solution candidate courante partielle (c'est l'aspectadaptatif du processus de recherche). GRASP a été appliquée aux plusieurs problèmes combinatoires. Pour une description détaillée des différentes extensions de GRASP,voir [73].

II.4.3 L'algorithme génétique

L'Algorithme Génétique est un algorithme d'optimisation qui s'inspire du processus d'évolution des êtres vivants. Il fut développé par le scientifique américain John Henry Holland dans les années 1970.

En recherche opérationnelle, l'Algorithme Génétique est une méta-heuristique de la grande famille des algorithmes d'évolution (*Evolutionary Algorithms*) qui offrent l'avantage de fournir des solutions de très grande qualité en un temps raisonnable; l'inconvénient est qu'il n'y a aucune garantie que la solution soit l'optimum global.

Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données[74]. Pour l'utiliser, on doit disposer des cinq éléments suivants :

1. Un principe de codage de l'élément de population. Cette étape associe à chacun des pointsde l'espace d'état une structure de données. Elle se place généralement après une phase demodélisation mathématique du problème traité. Le choix du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très employés à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs, pour l'optimisation de problèmes à variables continues.
2. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures.Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.
3. Une fonction à optimiser. Celle-ci prend ses valeurs dans R+ et est appelée fitness ou fonction d'évaluation de l'individu. Celle-ci est utilisée pour sélectionner et reproduire les meilleurs individus de la population.
4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état.
5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation [74].

II.4.4 L'algorithme mémétique

Un algorithme mémétique est une hybridation entre un algorithme génétique et une méthode de recherche locale. En effet, la puissance des algorithmes génétiques provientdu fait qu'ils

sont capables de balayer de manière globale l'espace des solutions contrairement aux méthodes de descente ou aux heuristiques qui explorent une petite zone de cet espace [75].
Un inconvénient d'un algorithme génétique est que lesopérateurs standards de croisement et de mutation nepermettent pas d'intensifier suffisamment la recherche [76].L'opérateur de mutation apporte une légère modification àl'individu. Son rôle est de favoriser la diversification desindividus alors que la sélection se charge de conserver lesmeilleurs. C'est pourquoi les algorithmes génétiques sontsouvent hybridés avec des méthodes de recherche locale[77]. C'est le cas des algorithmes mémétiques de Moscato[78].
Le principe général des algorithmes mémétiques est semblable à celui des algorithmes génétiques à la différence près qu'un opérateur de recherche locale est ajouté après celui de mutation. Ce dernier permet d'ajouter une intensification à la diversification apportée par le principe de l'algorithme génétique. Un algorithme génétique a besoin pour son fonctionnement d'une population initiale. Celle-ci peut-être construite de manière aléatoire ou de façon gloutonne. L'algorithme itère différentes étapes ensuite jusqu'à un critère d'arrêt propre à chaque problème/utilisateur :

- sélection de deux parents (individus) dans la population
- application d'un opérateur de croisement entre les deux parents afin de générer un nouvel enfant (individu)
- application d'un opérateur de recherche locale à l'enfant
- mise à jour de la population avec l'enfant (selon la stratégie garder les meilleures solutions, un échantillon large, etc.).

Le schéma récapulatif de l'algorithme mémétique peut être décrit comme suit :

- Population de base générée aléatoirement ou avec une initialisation de type gloutonne
- Évaluation de chaque individu
- Sélection d'un sous-groupe parmi la population
- Croisement de deux parents pour créer un nouveau fils
    - Application d'une recherche locale aux nouveaux enfants
    - Mutation à appliquer sur l'enfant
    - Mis à jour de la population en gardant les meilleurs enfants, et supprimant les plus mauvais individus

Plus de détail sur ces algorithmes sera exposé dans les chapitres suivants.

II.5Conclusion

Dans ce chapitre, nous avons vu le concept d'optimisation multi-objectifs ainsi que une aperçue sur les concepts de méta-heuristiques, puis les classifications. Il existe un très grand nombre d'autres méta-heuristiques. La recherche dans le domaine étant très active, il est impossible de produire une liste exhaustive des différentes méta-heuristiques d'optimisation. La littérature spécialisée montre un grand nombre de variantes et d'hybridations entre méthodes, particulièrement dans le cas des algorithmes évolutionnaires.

Les spécialistes du domaine s'accordent cependant à tenter de réduire l'emploi de métaphores prétextant amener à de « nouveaux » algorithmes[79]. Les raisons invoquées sont l'absence d'utilité des métaphores, le fait qu'elles masquent l'absence de nouveauté ou une validation expérimentale non rigoureuse.

Dans le chapitre suivant, nous allons présenter la première contribution de la recherche locale stochastique.

Chapitre III

Contribution 1 : une recherche locale stochastique

hybride pour le Problème de composition des services web

Chapitre III : Contribution 1 : une recherche locale stochastique hybride pour le Problème de composition des services

III.1 Introduction

La composition optimale des services web est un processus complexe et tâche difficile.C'est le processus d'une combinaison entre plusieurs services sur la base de règles de composition pour répondre à une demande de l'utilisateur qui ne peut être satisfaite par un service individuel.

Les règles de composition déterminent l'ordre d'invocation des services et la possibilité ou non d'invoquer un service.Dans ce chapitre, nous présentons la composition des services web comme un multi-objectif problème d'optimisation où plus d'une fonction objectif peuvent être optimisés simultanément.

Nous allons présenter trois approches de composition des services web qui sont :

- La recherche locale stochastique SLS
- L'approche d'algorithme génétique GA
- L'approche hybride SLS-GA

Cette contribution a été publié dans une conférence internationale AIAP 2018 :

Azouz, Y., Boughaci, D (2018), Stochastic local search versus genetic algorithm and solving the optimal web services composition problem, AIAP 2018: Artificial Intelligence and Its

Applications.

Nous présentons en suit l'approche de recherche locale stochastique (SLS).

III.2 La Recherche Locale Stochastique SLS

Le problème de l'optimisation de la composition des services web est une tâche complexe et exigeante.Il consiste à sélectionner certains services élémentaires pour générer de nouvelles combinaisons de services. La meilleure combinaison peut avoir les valeurs optimales de QoS attributs étudiés.Ces attributs doivent être pris en compte lors de la sélection du service composite optimal.

Nous présentons dans cette section une approche SLS multi-objectifs pour le problème de composition des services web.Le SLS proposé traite quatre fonctions objectifs du modèle de QoS présentées dans la section III.3. La méthode SLS multi-objectifs (MO-SLS) vise à rechercher les solutions en prenant en compte simultanément ces quatre critères.

Nous appliquons l'approche MO-SLS pour trouver le plan d'exécution optimal de la composition des services web.Un plan d'exécution des services sélectionnés est représenté par un vecteur binaire. L'ensemble des vecteurs binaires forme une matrice X, où chaque ligne représente un plan d'exécution de la composition des services web.

SLS est une méta-heuristique de recherche locale qui a déjà été étudiée pour plusieurs problèmes d'optimisation, elle a été proposée par Hoos [1][80]. SLS fonctionne sur une solution initiale aléatoire.Ensuite, il effectue un certain nombre d'étapes locales alliant combiné les stratégies de diversification et d'intensification pour trouver une solution optimale :

- **Etape 1 :** La phase de diversification qui consiste à sélectionner un individu voisin au hasard.Cette phase est appliquée avec une probabilité fixe $wp > 0$ et l'intensification phase avec une probabilité $(1 > wp)$.Le *wp* est la probabilité fixé empiriquement.
- **Etape 2 :** La phase d'intensification qui consiste à sélectionner le meilleur individu voisin ayant la meilleure fitness. Le processus SLS est répété jusqu'à un certain nombre d'itérations appelé *max_iter* est atteint.

Dans les deux sous-sections suivantes, nous parlerons de la représentation des solutions et un aperçu sur le mécanisme de voisinage utilisé.

III.2.1 La représentation de la solution : chaque solution candidate est représentée par un chromosome de gènes.Un chromosome est modélisé par un vecteur composé d'un ensemble de variables binaires. Chaque variable binaire représente un service candidat à la composition

des services.Il prend la valeur 1 ou 0, 1 si le service correspondant fera partie du service composite optimal trouvé et 0 sinon.

La Figure III-1 montre une solution de composition de services utilisant 9 services et 3 taches. Chaque tache a 3 services candidats.

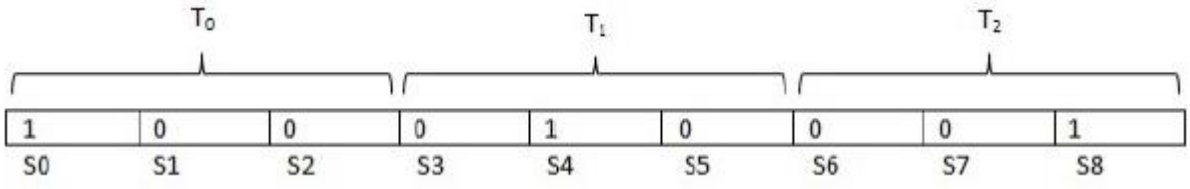

Figure III.1 : Exemple de solution

Chaque service est représenté par une variable binaire et les 3 variables binaires sont regroupées selon une tâche.Pour chaque groupe de 3 services, un seul service sera sélectionné à la composition.Chaque solution correspond à une matrice X. D'après la Figure III-1, nous pouvons dire que le service S_0 est alloué à la tâche T_0, le service S_4 est alloué à la tâche T_1, et enfin, la tâche T_2 est affectée pour entretenir le service S_8.

III2.2 Les solutions de voisinage : la génération des solutions de voisinage sont faites aléatoirement pour chaque solution candidates voisines.Cela se fait par un changement élémentaire de chaque vecteur variable.Un changement élémentaire consiste à changer la valeur 1 par 0 ou 0 par 1, mais une seule opération de changement est autorisée à la fois. Si la valeur de la variable est égale à 0 alors elle est mise à 1. Si la valeur de la variable est égale à 1 alors elle est mise à 0. Chaque tâche Tj est affectée à un service S_{i1}, un changement élémentaire permet d'affecter la tâche Tjà un autre service S_{i2} appartenant au même groupe de service, cela permet de générer une solution candidate voisine.

III.3 L'approche d'Algorithme Génétique GA

Dans cette section, nous présentons un algorithme génétique (GA) pour la composition des services web.Comme vu avec SLS, nous nous occupons du problème d'optimisation multi-objectif de la composition des services avec quatre fonctions objectifs du modèle de QoS qui est présenté dans la section III.4.1 et nous recherchons les solutions prenant en compte ces quatre critères simultanément.

Chaque composition de services a un plan d'exécution, la méthode GA multi-objectifs est utilisée pour sélectionner le plan d'exécution optimal.Pour chaque plan d'exécution d'une composition de services, un chromosome est créé.En accumulant des chromosomes de plans

d'exécution, nous obtenons un ensemble de population de solutions candidates pour une composition optimale des services web.

III3.1 Le processus GA :

L'approche GA multi-objectifs (MO-GA) est un algorithme GA adapté aux problèmes multi-objectifs. Le résultat est un ensemble de solutions optimales appelées solutions Pareto-optimales.

L'approche MO-GA comprend les étapes principales suivantes:

Étape 1. Générer une population initiale : nous générons aléatoirement un ensemble de chromosomes qui représentent la population initiale P de taille N.

Étape 2. Évaluez toutes les solutions de la population P en calculant leur fitness : dans MO-GA, chaque solution a plusieurs valeurs de fitness où chaque fitness représente un critère non fonctionnel.

Étape 3. Calculez une solution non dominée de P et sa liste de solutions non dominées.

Étape 4. Sélectionnez deux parents à l'aide à la sélection du tournoi binaire.Elle consiste à tirer au hasard deux solutions de la population P.

Étape 5. Générer un ensemble des solutions enfants P_enfant en croisant ces deux parents de P réglés en utilisant un croisement à point unique.

Étape 6. Répétez 4 et 5 pour toutes les paires de solutions enfant de P enfant. Étape 7. Exécutez l'opérateur de mutation sur l'ensemble P_enfant en utilisant fonction aléatoire avec probabilité P_r.

Étape 8. Une procédure de correction est exécutée si la solution est irréalisable : cette procédure est exécutée sur chaque nouvelle solution candidate générée alors qu'elle doit respecter toutes contraintes du modèle QoS étudié.

Étape 9. Exécutez une méthode de recherche locale sur P_enfant en tant que perturbation simple pour obtenir une solution de meilleur voisin de l'enfant la solution.

Étape 10. Calculer le meilleur enfant S à partir de P_enfant et sa liste de solutions enfants non dominantes. Après cela, comparez-le avec S_best liste de solutions enfants meilleures et non dominantes pour améliorer la solution actuelle et sa liste de solutions non dominées.

Étape 11. Répétez les étapes 2 à 10 jusqu'à ce que les conditions d'arrêt soient satisfaites, le nombre d'itérations *max_iter*.

Le processus de recherche est répété jusqu'à ce qu'un certain nombre de générations appelées *max_iter* est atteint, le résultat de son exécution est une solution Pareto-optimale localeavec une liste de non-dominés solutions.

III.3.2 Les solutions de voisinage

Les solutions voisines sont générées en modifiant un bit pour chaque génération de voisin. L'opération de voisinage a pour objectif d'améliorer la solution candidate en appliquant une perturbation simple sur un chromosome. Par exemple, si 110011 est le vecteur variable courant, alors les voisins possibles peuvent être comme le montre la figure III.2 :

Figure III-2: Exemple de voisinage

III.3.3 Opérateurs de mutation et croisement

L'opérateur de mutation

Il est appliqué aux solutions individuelles après l'opérateur de croisement où un ou plusieurs gènes sont modifiés de manière aléatoire dans une chaîne avec une faible probabilité de créer un nouveau chromosome.

L'objectif de cet opérateur est de maintenir la diversité de la population et d'augmenter la possibilité de ne pas perdre toute solution potentielle et trouver l'optimum global [81].La mutation se produit au cours de l'évolution selon une probabilité de mutation définissable par l'utilisateur, elle modifie une ou plusieurs valeurs de gêne dans un chromosome.

Exemple de mutation :

Cet exemple nous montre comment appliquer l'opérateur de mutation sur une solution donnée. Nous avons supposé que nous avons 3 tâches (M=3) qui peuvent être exécutées par 15 candidats services (N=15) et où chacun des 5 services est affecté à une tâche.

0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
Bit 1-5					Bit 6-10					Bit 11-15				

Figure III-3: Example of candidate solution

Lorsque nous appliquons l'opérateur de mutation respectivement sur le bit 3, bit 7 et bit 14 nous obtiendrons respectivement ce muté solutions :

Cas du bit 3:

0	0	0	1	0	0	1	0	0	0	0	0	0	1	0
Bit 1-5					Bit 6-10					Bit 11-15				

Figure III-4: Exemple de solution mutée par mutation du bit 3

Cas du bit 7:

0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
Bit 1-5					Bit 6-10					Bit 11-15				

Figure III.5: Exemple de solution mutée par mutation du bit 7

Cas du bit 14:

0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
Bit 1-5					Bit 6-10					Bit 11-15				

Figure III-6: Exemple de solution mutée par mutation du bit 14

L'opérateur de croisement :

L'opérateur de croisement est une technique d'exploration rapide de l'espace de recherche.Il est utilisé dans notre méthode pour deux candidats solutions.Elle consiste à croiser deux parties de cette paire de solutions, la sélection de ces deux parties est basée sur la valeur d'une variable aléatoire appartenant à [0, M] ;M : le nombre de tâches.L'exemple suivant montre l'application de l'opération de croisement sur deux solutions candidates.

Exemple de croisement :

On prend le même cas quand nous avons N = 9 services et M = 3 tâches, nous pennons les deux chromosomes de la Figure III.7, nous obtiendrons trois instances de croisement :

0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
Bit 1-5					Bit 6-10					Bit 11-15				

1	0	0	0	0	0	0	0	1	0	0	1	0	0	0
Bit 1-5					Bit 6-10					Bit 11-15				

Figure III-7 Exemple de solutions candidates pour un croisement

Le croisement par l'échange des premières parties des deux solutions candidates de la Figure III.7, nous obtenons :

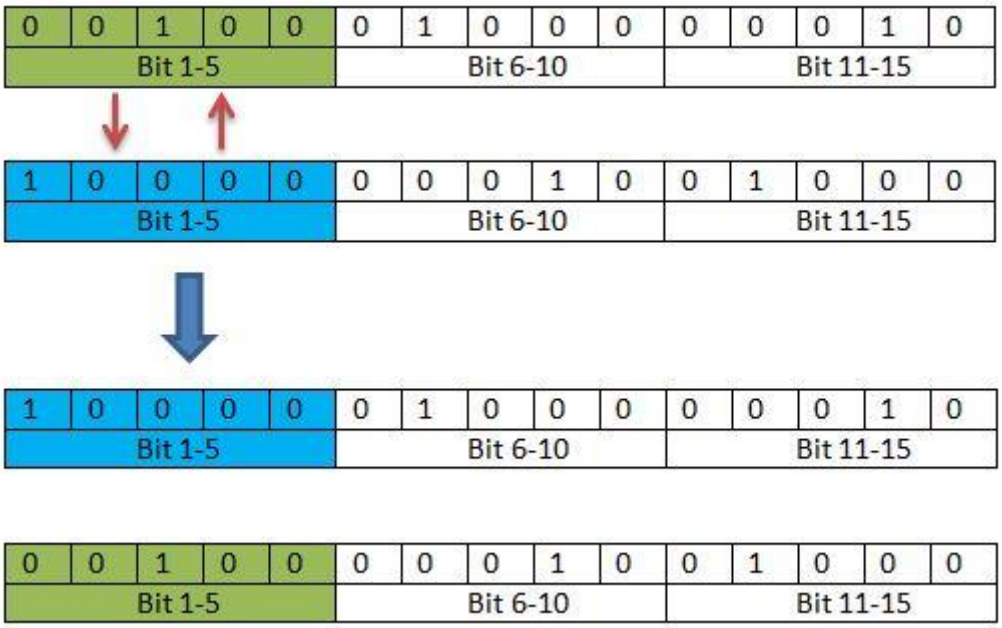

Figure III-8: Exemple de croisement de solutions, cas Rand =0

III.4 Modélisation du problème de la composition de services web

Nous pouvons modéliser le problème de la composition des services web comme un problème d'optimisation combinatoire avec certaines fonctions objectifs qui doivent être optimisées.Dans notre travail, nous utilisons un modèle similaire présenté dans [82].Dans le processus de composition, nous considérons un ensemble de services s_i, i∈ [1…n] où chaque service peut exécuter un ensemble de tâches t_j, j ∈ [1…m].Nous avons un ensemble de n services et un ensemble de m tâches.Nous considérons également qu'un service peut dépendre d'autres services.Le problème est de trouver une composition de services dans laquelle chaque service s_i sera affecté à une tâche t_j.Autrement dit, le problème est de trouver pour chaque tâche t_j le service approprié s_i de l'ensemble des services considérés [82].

Cette combinaison peut être représentée par une matriceX_{ij}où la ligne i représente le service candidat s_i et la colonne j représente la tâche t_jà affecter au service s_i. La représentation matricielle est donnée dans l'éq.1 où X représente les services affectés à une composition.n : est le nombre de services, m : est le nombre de tâches [82].

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & & x_{2m} \\ & \vdots & & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nm} \end{bmatrix} \qquad (1)$$

Comme le montre la matrice X, il existe un ensemble de services candidats. L'objectif est de sélectionner les services appropriés qui seront partie de la composition.Dans l'éq.2, nous donnons si un service sera sélectionné ou non.L'équation 2 permet de vérifier si le service s_i est affecté ou non à la tâche t_j[82].

$$Xij = \begin{cases} 1, \text{si le service i est assigné à la tache j} \\ 0, \text{sinon} \end{cases} \forall\, i \in [1..n], \forall\, j \in [1..m] \quad (2)$$

III.4.1 Modèle de qualité de service QoS

Le but de notre travail est de trouver une composition optimale de services capables d'exécuter efficacement toutes les tâches. Le processus de composition nous permet d'affecter chaque tâche à un service. Nous donnons dans la Figure III.9 un exemple de plan d'exécution où il y a m tâches et chacune a un groupe de n services.

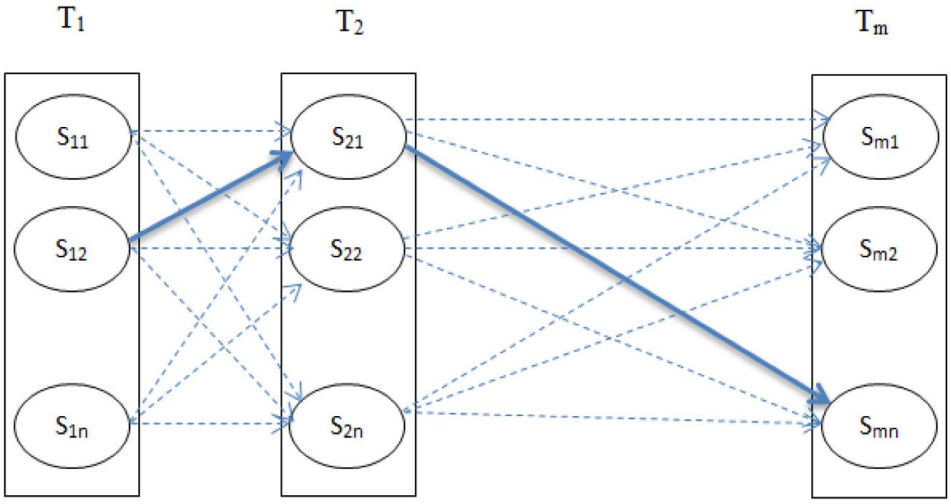

Figure III-9: Exemple de plan d'exécution

Pour chaque tâche, nous devons sélectionner le service qui sera lancé pour exécuter la tâche.Un plan d'exécution correspond à une composition de services. Nous utilisons ici un modèle de qualité de service (QoS) pour estimer la qualité du service offert qui peut être considérée comme un critère pour sélectionner la composition optimale des services.Le modèle de qualité de service utilisé dans notre étude est basé sur quatre critères donnés comme suit :

Le prix d'exécution : le prix d'exécution C_p d'une exécution du plan p est la somme des prix d'exécution du service s_i, C_{ij} comme indiqué dans [82]:

$$Cp = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}$$

La durée d'exécution : cette mesure t_{ij} évalue le temps d'exécution entre la requête et la réponse [16]. Puisque nous considérons que l'exécution des tâches est séquentielle, Le temps peut être exprimé en référeche à la composition : [82].

$$Tp = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}$$

La disponibilité :elle peut être exprimée comme la valeur moyenne pour chaque service de la composition, comme le montre cette équation : [82].

$$Ap = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}}{n}, n \geq 0$$

Laréputation : la réputation d'un plan d'exécution p est la moyenne de la réputation R_p de chaque service dans le plan d'exécution : [82].

$$Rp = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij}$$

III.4.2 Les fonctions objectifs

Nous utilisons les même fonctions objectifs utilisées dans [83].Nous traitons les quatre critères avec la même importance en utilisant une approche d'optimisation multi-objectifs.Notre objectif est pour minimiser les coûts et le temps et pour maximiser la disponibilité et la réputation.Nous avons quatre objectifs à prendre en comptesimultanément.Ces quatre objectifs sont utilisés ensemble, et aucun un poids est donné à l'un d'entre eux [83].

La première est la minimisation des coûts donnée par :

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} p_{ij} x_{ij}$$

p_{ij} : précise que le service s_i est un candidat, parmi d'autres, dédié à la tâchet_j, c'est le moyen d'exprimer la compatibilité entre les services & les taches à réaliser.

Le deuxième objectif est la minimisation du temps donnée par :[50].

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} p_{ij} x_{ij}$$

Le troisième objectif est la disponibilité à maximiser :[50].

$$max \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} p_{ij} x_{ij}$$

Le quatrième objectif est la valeur du service composite à maximiser : [50].

$$max \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} p_{ij} x_{ij}$$

III.4.3 Les contraintes du problème

Pour les contraintes du problème selon le modèle de QoS étudié, il existe deux contraintes principales qui doivent être validées sur chaque solution candidate : La première contrainte est qu'un seul service appartient à une composition pour chaque tâche.Cela signifie que chaque tâche sera affectée à un seul service [83]:

$$\sum_{i=1}^{m} x_{ij} P_{ij} = 1, \qquad \forall \, j \in [1, m]$$

La variable précise s_i le service appartient ou non à la composition, et signifie que le service s_i exécutera la tâche t_j.

La seconde contrainte est le budget utilisateur [83]:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \leq w, \qquad w > 0$$

Pour comparer deux solutions candidates, nous avons utilisé la relation de domination qui est présenté comme suit :

```
Function dominate (V1, V2)
sors ← true
I ← 0
while (I < V1.length and sors == true) do
if Fi(V1) ≥ Fi(V2) then
        I ← I + 1
    else
        sors ← false;
end if
end while
return sors
end function
```

La résolution d'un problème d'optimisation multi-objectif peut donc s'avérer long et fastidieux si desméthodes appropriées ne sont pas mises en œuvre. En milieu industriel où les problèmes

d'optimisation sont très complexes (e.g., multiples objectifs, plusieurs variables et contraintes, non-linéarités), le temps de recherche des solutions optimales devient également un facteur important à prendre en compte. C'est ainsi que des "méta-heuristiques" sont généralement utilisées pour résoudre ces types de problèmes, en recherchant, à défaut de trouver l'optimum global, à se rapprocher aussi près que possible de ce dernier, en faisant un compromis avec le temps de calcul.

Remarque :

Notre objectif dans ce travail de thèse est la résolution méta-heuristique de problème multi-objectifs de composition de services web. Le but principal est de trouver un ensemble de solutions optimales au sens de Pareto. Une solution est dite optimale au sens de Pareto est une situation pour laquelle il n'est pas possible d'améliorer le sort d'un individu sans détériorer celui d'au moins un autre. Dans notre cas, il s'agit d'un ensemble de solutions pseudo-optimales dites aussi solutions approximatives parce que nous utilisons des méta-heuristiques. Ces dernières donnent des solutions approchées et non pas exactes, elles sont capables d'explorer uniquement un espace de recherche et non pas la totalité des solutions contrairement aux méthodes exactes.

III.5 Expérimentation

Nous avons différentes classes de services.Chaque classe contient un ensemble d'instances de service avec la même fonctionnalité.Chaque service web est défini par ses entrées et ses critères QoS. Nous utilisons deux benchmarks : le premier est le jeu de données QWS cité dans [84], [85], le second est généré aléatoirement.La première contient le temps, la disponibilité et la réputation mais ne contient pas le critère de coût.Ainsi, pour notre cas, nous avons proposé d'ajouter de nouveau critère présentant le coût d'exécution du service web.Nous avons étudié plusieurs instances de tailles différentes, les instances sont obtenues en faisant varier le nombre de services (noté N) et le nombre de tâches (noté M).Dans nos expérimentations, nous avons décidé d'évaluer nos approches (SLS, GA et GA-SLS) sur cinq instances N,M=(15,3),(30,3),(30,5),(60,3),(60,5).Nous avons implémenté les solutions en Java version 1.6 sous Windows 7 fonctionnant sur Intel (R) Core (TM) i5-2520M 2,5 GHz et 8 Go de RAM.

Comme utilisé dans [15], nous utilisons 4 fonctions objectifs et 2 contraintes du modèle QoS, nous avons également supposé que pour chaque tâche, il existe un ensemble de services dédiés, et chaque service peut s'exécuter une tâche seulement.Chaque tâche a le même nombre de services.

III.5.1Expérimentation de l'approche SLS

Dans cette section, nous détaillons les résultats obtenus lors de l'application l'approche SLS pour le problème de composition des services web par l'utilisation de types de Benchmark.

III.5.1.1 SLS sur le Benchmark cité dans[84][85]

Nous présentons les résultats numériques obtenus en utilisant le benchmark cité dans [84][85],dans la Figure III-10. Il est clair de voir qu'un compromis de 15 solutions est trouvé à la 5ème itération. Les solutions de compromis respectent toutes les contraintes considérées.

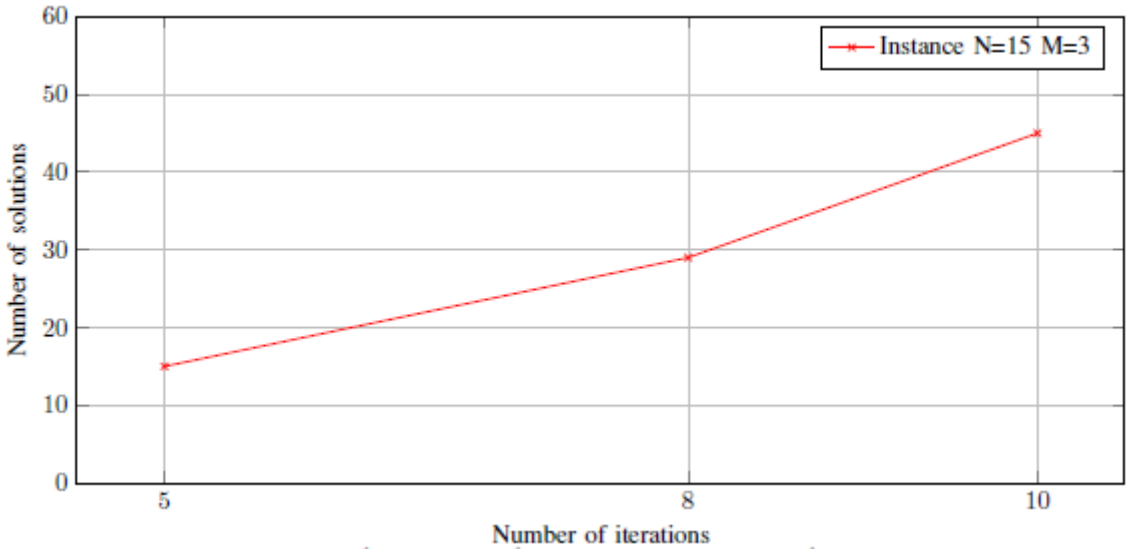

Figure III-10: Ensemble de solutions non dominé par itérations

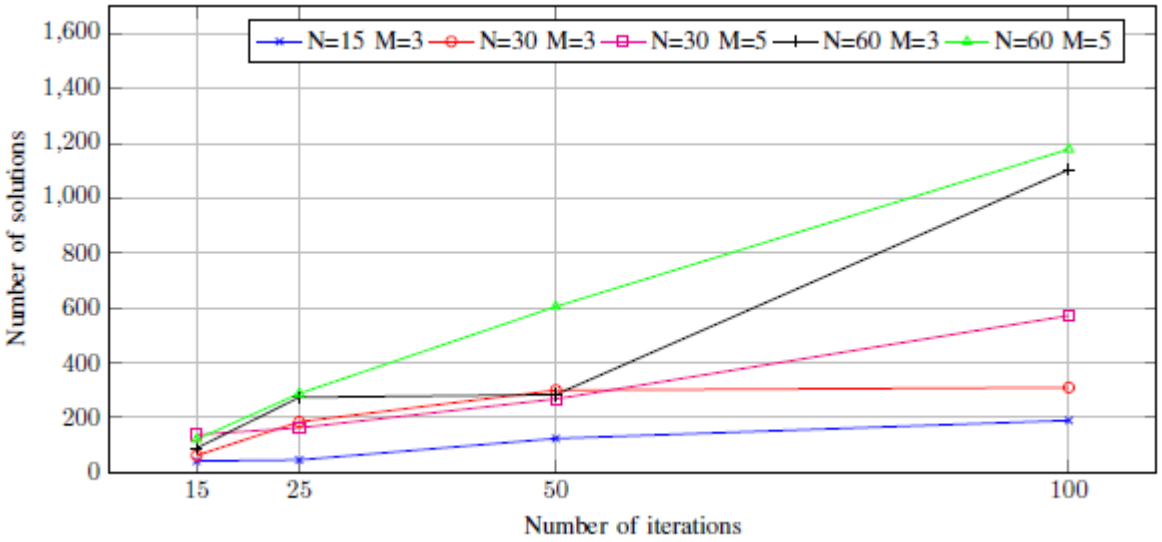

Figure III-11: Solutions non dominées par itération pour cinq instances

L'évolution du modèle basée sur le nombre de solutions distinctes Pareto-optimales trouvées localement est donnée par la Figure III-11. Elle donne les résultats sur cinq instances étudiées.La Figure III-12 affiche le temps écoulé pour chaque exécution d'instance.Par exemple, pour l'instance N=15 M=3, nous remarquons que 42 solutions sont trouvées dans la 15ème itération en 70 millisecondes et 190 solutions sont trouvées dans les 100 itérations en 1536 millisecondes.

Pour l'instance de N=30 et M=3 (30 services et 3 tâches), après l'exécution de 15 itérations nous avons trouvé 62 solutions en 371 millisecondes.La méthode SLS donne 310 solutions non dominées en 2580 secondes à la 100ème exécution itération.

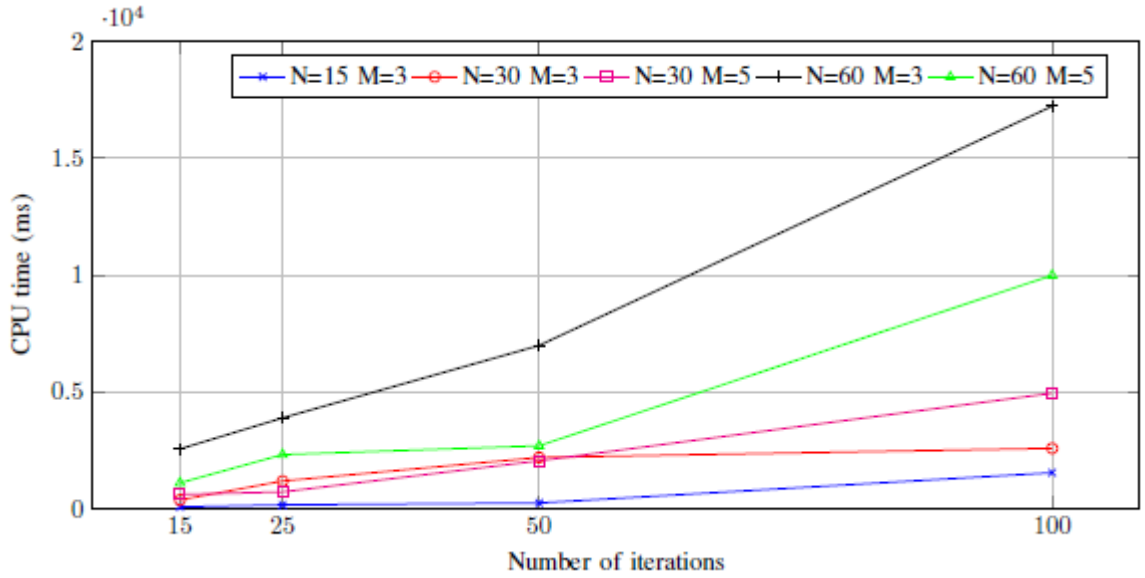

Figure III-12: Temps écoulé par itération pour cinq instances

Pour l'instance de 60 services et trois tâches (N=60 et M=3), 89 solutions distinctes sont trouvées à la 15ème itération. En utilisant 60 services et 5 tâches (N=60 et M=5), cela l'expérience de notre algorithme a montré que nous pouvons trouver jusqu'à 121 solutions distinctes de la 15e itération en environ 910 millisecondes, ce résultat évolue à chaque fois que l'on augmenter le nombre d'itérations.

Nous avons noté que le nombre de solutions optimales trouvées pour les instances N=60 M=3 et N=60 M=5 est supérieure au nombre trouvé pour les autres instances.Ces résultats montrent que SLS réussit à trouver des solutions de compromis.

III.5.1.2 SLS sur le Benchmark généré aléatoirement

Nous présentons dans cette section les résultats trouvés lors de l'application méthode SLS multi-objectifs sur le benchmark aléatoire. D'après la Figure III-13, nous voyons que pour l'instance N=15 M=3, un compromis de 21 solutions est trouvé à la 5ème itération.Les solutions de compromis respectent toutes les contraintes sans aucune violation. La Figure III-14 montre que le nombre de solutions Pareto-optimales trouvées localement est important pour toutes les instances étudiées.Pour l'instance (N,M)=(60,5), le nombre de solutions Pareto-optimales localement trouvées a atteint 1386 à la 100ème itération avec un temps écoulé était 16975 millisecondes comme illustré à la Figure III-15.

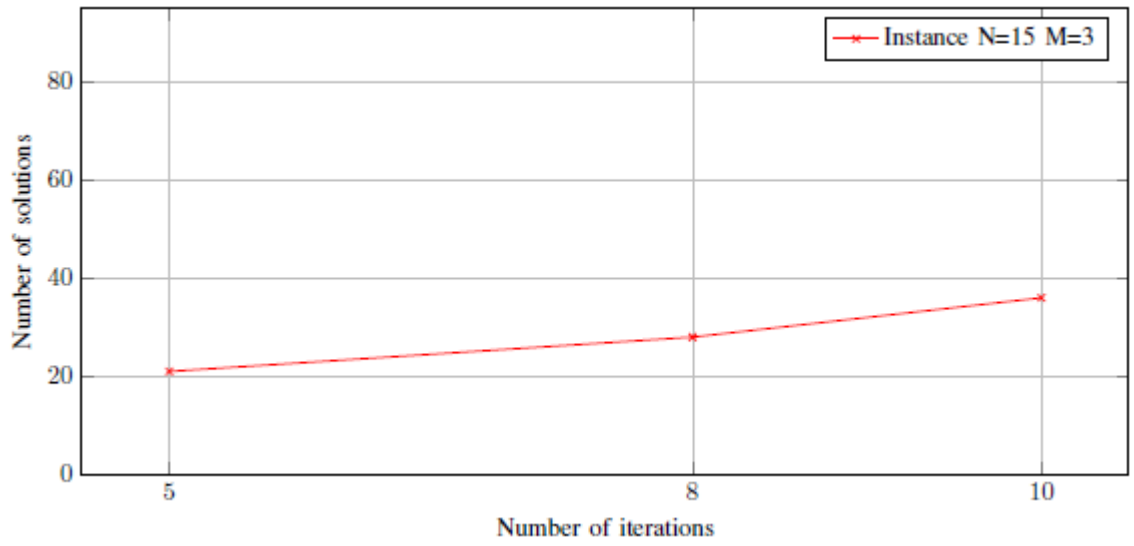

Figure III-13: l'ensemble de solutions non-dominées par itération

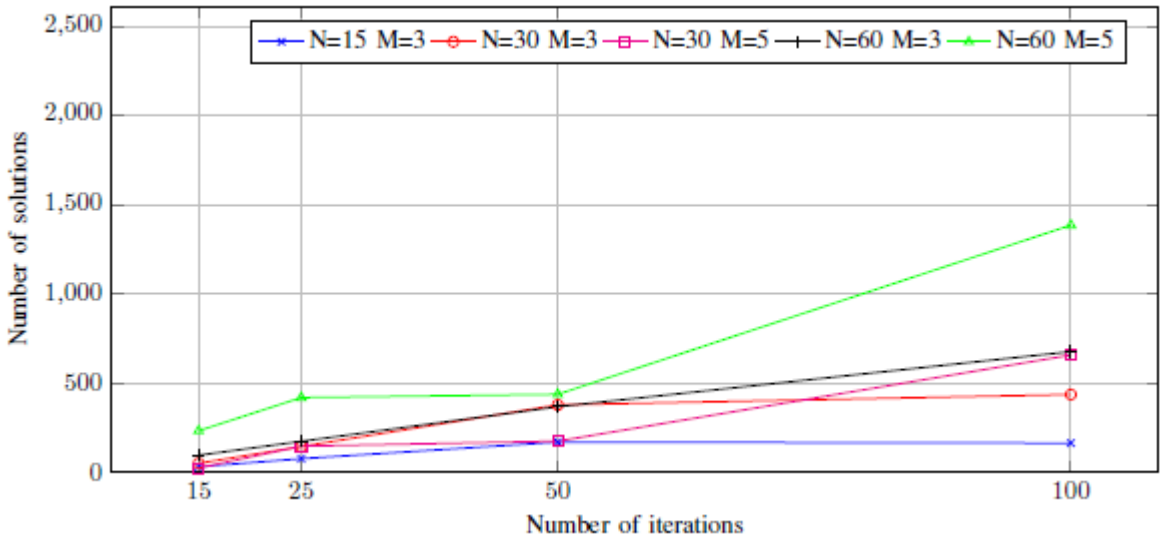

Figure III.14: l'ensemble de solutions non-dominées par itération pour cinq instances

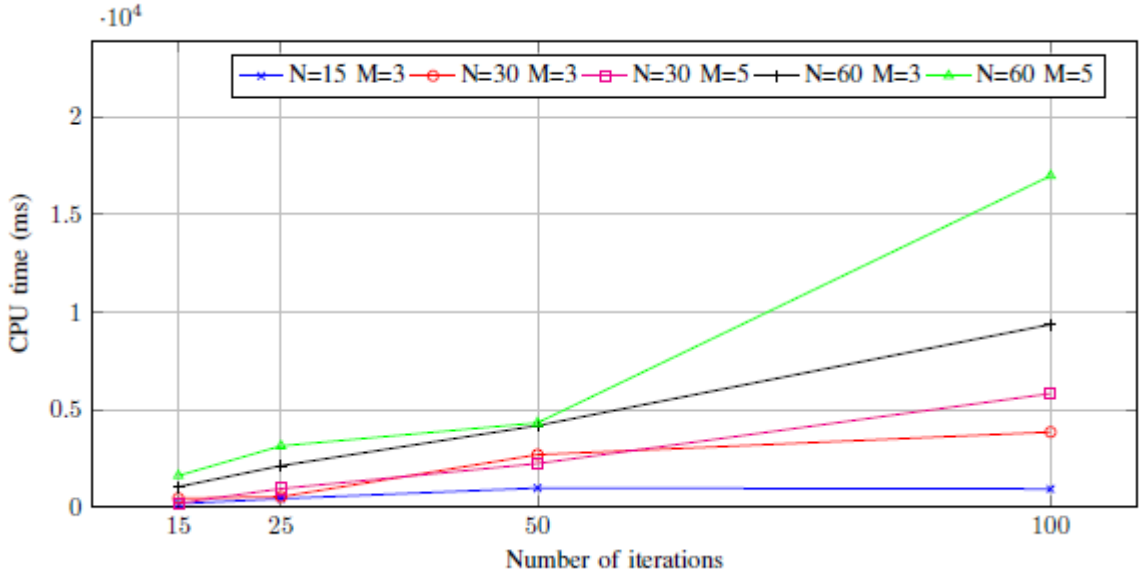

Figure III-15: Temps écoulé par itération pour cinq instances

La méthode SLS est capable de trouver une solution optimale avec un ensemble de solutions Pareto-optimales.Ainsi, l'utilisateur choisira la solution optimale pouvant satisfaire ses besoins.Lorsque l'ensemble de solutions optimales trouvées est très grand, il sera difficile de choisir la bonne solution.Sur ce, nous avons pensé à utiliser une autre méthode dont l'objectif est de trouver un certain nombre de solutions optimales plus réduit.Nous avons proposé d'utiliser l'algorithme génétique multi-objectif (GA).Les résultats obtenus sont présentés dans la section suivante.

III.5.2 Expérimentations de l'approche GA

Lors de l'application d'un algorithme génétique multi-objectif (GA) pour le problème d'optimisation de la composition de services web, un ensemble de solutions optimales est trouvé et chacun représente une composition optimale de services.C'est un ensemble de solutions non dominées, appelé aussi solutions Pareto-optimales.

III.5.2.1 GA en utilisant le Benchmark cité dans [84][85]:

Nous présentons ici les résultats numériques trouvés par GA sur le Benchmark cité dans [84][85]**.** Nous pouvons voir sur la Figure III-16, l'évolution dans notre modèle le nombre de solutions optimales par itération où 45 solutions se sont trouvées à la 5ème itération.Ces solutions ne violent aucune contraintedu modèle QoS.

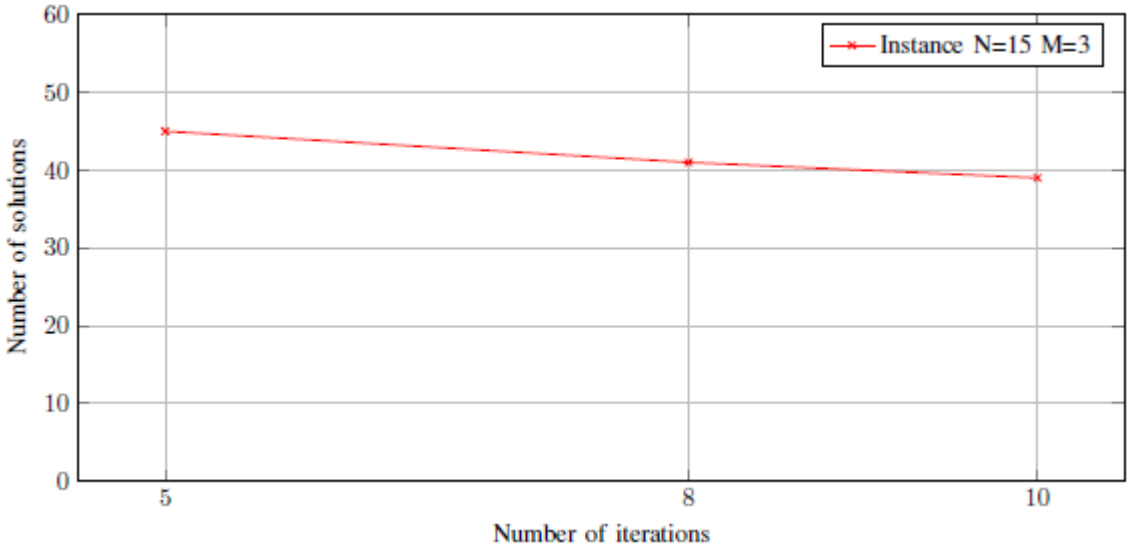

Figure III-16: l'ensemble de solutions par itération

Pour plus de détails sur l'évolution de l'approche MO-GA, nous l'avons testé dans d'autres cinq cas comme présentés dans la Figure III-17. Sur ce, nous montrons cinq diagrammes pour les cinq instances d'exécution en faisant varier le nombre services et le nombre de tâche.

Pour l'instance de N=15 et M=3, la Figure III-17 montre que 46 les solutions sont trouvées dans la 15ème itération en 8813 millisecondes. Nous voyons aussi que 55 solutions sont trouvées à la 100ème itération en 11960 millisecondes.

Pour l'instance de 30 services et 3 tâches (N=30 et M=3), les Figures III-17 et III-18 montrent que 67 solutions distinctes sont trouvées pour l'itération 15 en 12054 millisecondes, par ailleurs 57 solutions distinctes non dominées pour la 100ème itération dans 12306 millisecondes.

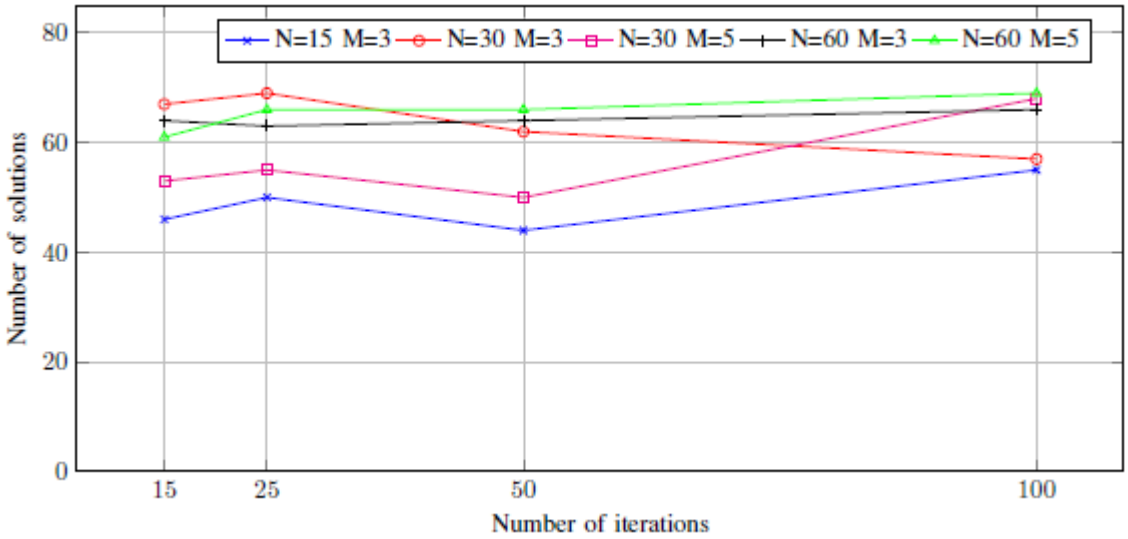

Figure III-17: Ensemble de solutions non-dominées par itération pour cinq instances

En revanche, le cas de 60 services et 3 tâches, comme le montre la Figure III-17, 64 solutions distinctes sont trouvées à la 15ème itération en 27440 millisecondes. Après la 100ème itération d'exécution, 66 solutions optimales sont trouvées dans 16036 millisecondes.

Dans le cas de 60 services et 5 tâches, chaque 12 services sont réservés pour chaque tâche. Un service sera sélectionné pour exécuter une tâche. Nous avons vu que nous pouvions trouver jusqu'à 61 solutions distinctes dans la 15ème itération et 69 solutions dans la 100ème itération en 25925 millisecondes.

Lorsque nous trouvons un ensemble de solutions non dominées, l'utilisateur sélectionner une solution, les autres sont conservées.

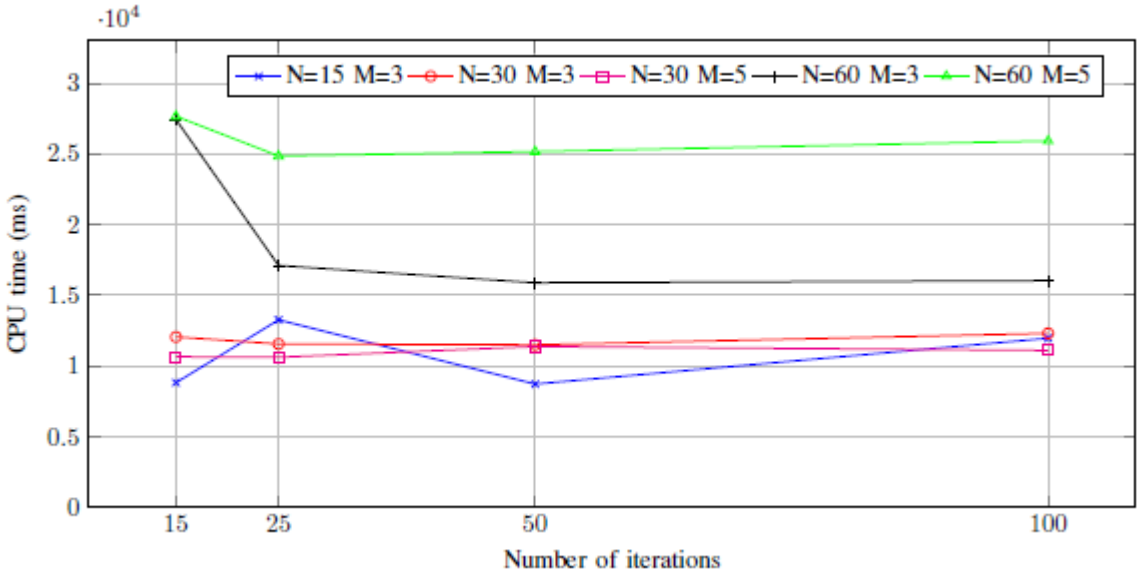

Figure III.18:Temps écoulé par itération pour cinq instances

III.5.2.2 GA en utilisant le Benchmark généré aléatoirement :

En utilisant un benchmark généré aléatoire, un ensemble de solutions non dominées est trouvé et chacune représente une composition optimale.À partir de ces ensembles de solutions optimales trouvées, l'utilisateur peut en sélectionner un. Nous représentons dans cette section les résultats numériques sur un jeu de données aléatoire.Dans la Figure III-19, nous montrons l'évolution des nombre de solutions optimales pour l'instance N=15 et M=3. Des compromis de 36 solutions sont trouvés à la 5ème itération.

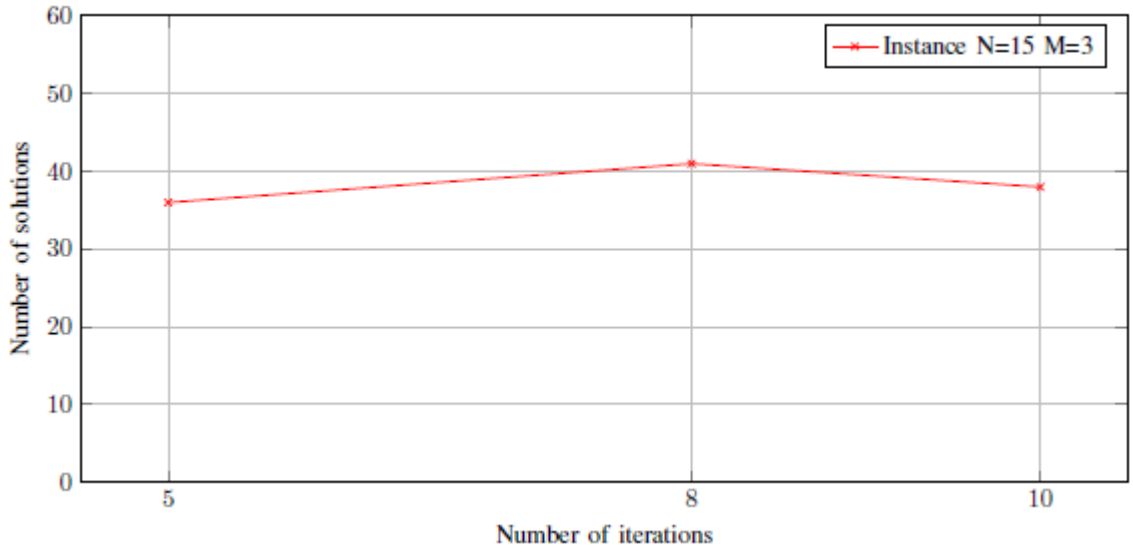

Figure III-19:Ensemble de solutions non dominé par itérations

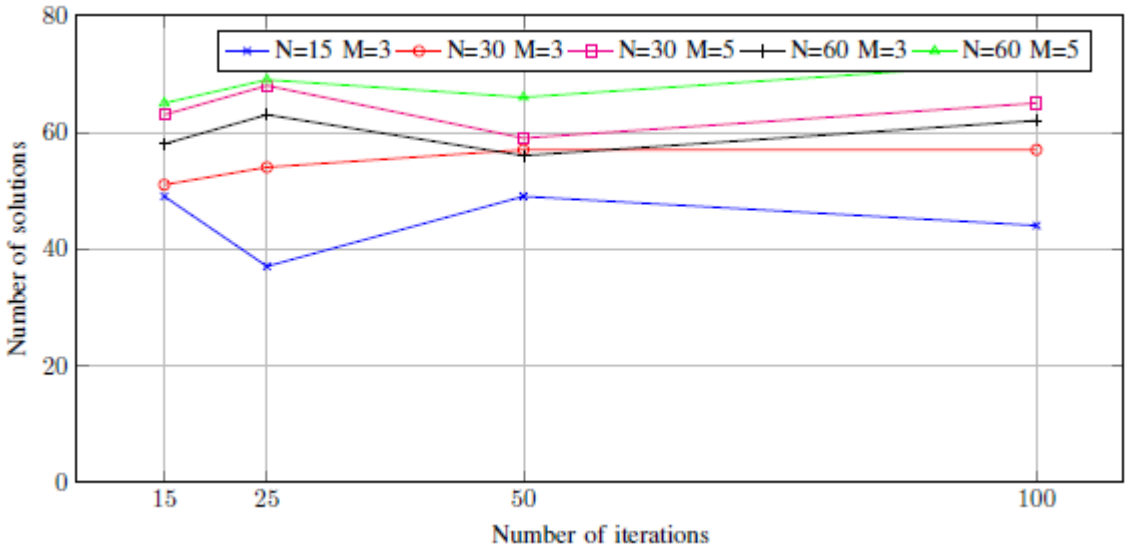

Figure III-20: Ensemble de solutions non dominé par itérations pour cinq instances

D'après la Figure III-20, nous pouvons dire que le nombre de solutions est compris entre 9 et 27, ce nombre est restreint.Sur d'autre part, nous avons noté que le nombre de solutions optimales est approximativement stable pour toutes les instances.

La Figure III-21 montre que le temps écoulé pour chaque exécution d'une instance augmente à chaque fois que le nombre d'itérations augmente.Par exemple par exemple N=15 M=3, le temps écoulés étaient de 10802 millisecondes à la 15ème itération, et 9041 millisecondes à la 100ème itération.

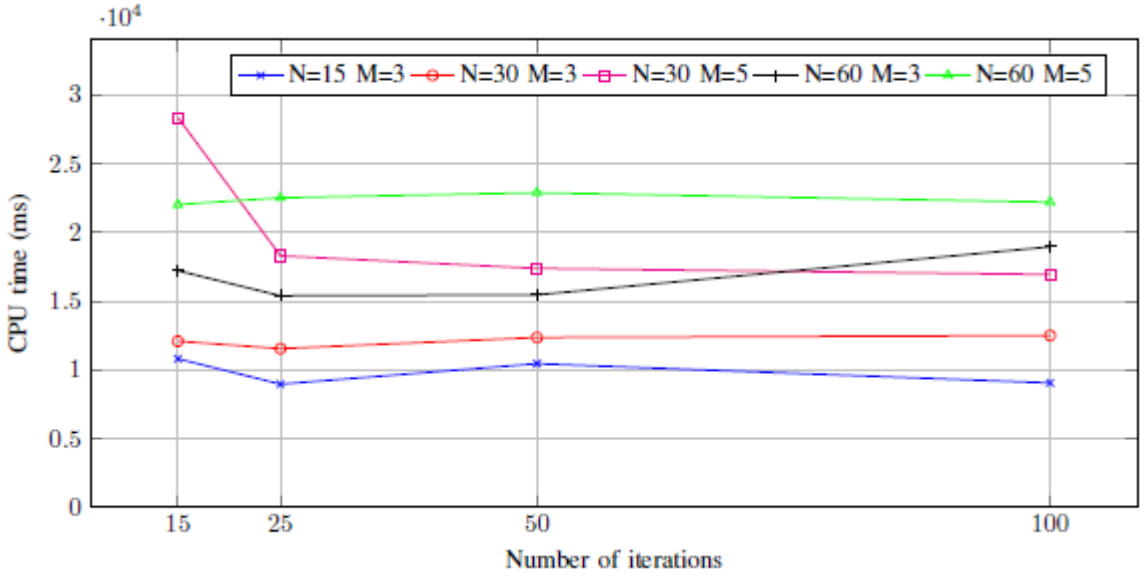

Figure III-21:Temps écoulé par itération pour cinq instances

GA affiche les meilleurs résultats en termes de nombre de solutions de compromis. Dans la section suivante, nous exposons les résultats obtenus par l'application la méthode hybride GS-SLS.

III.5.3 Expérimentations de l'approche GS-SLS

Dans ce qui suit, nous présentons les résultats numériques trouvés de processus GA-SLS sur les deux benchmarks cités dans [84]&[85].

III.5.3.1 GA-SLS en utilisant le Benchmark cité dans [84]&[85]:

Lorsque nous utilisons le benchmark cité dans [84]&[85], nous présentons dans la Figure III-22. La relation entre les solutions non dominées et le numéro d'exécution de l'itération (15, 25, 50, 100...).Ici nous notons que le nombre de solutions optimales est compris entre 60 et 95 concernant les cinq instances différentes.

La Figure III-23 montre le temps d'exécution de l'algorithme GA-SLS pour les numéros d'itération : 15, 25, 50, 100...etc.Nous remarquons que le temps écoulé pris par chaque instance est très important.Par exemple pour l'instance 04, le temps écoulé ne dépasse pas 16505 millisecondes pour 100 itérations d'exécution.

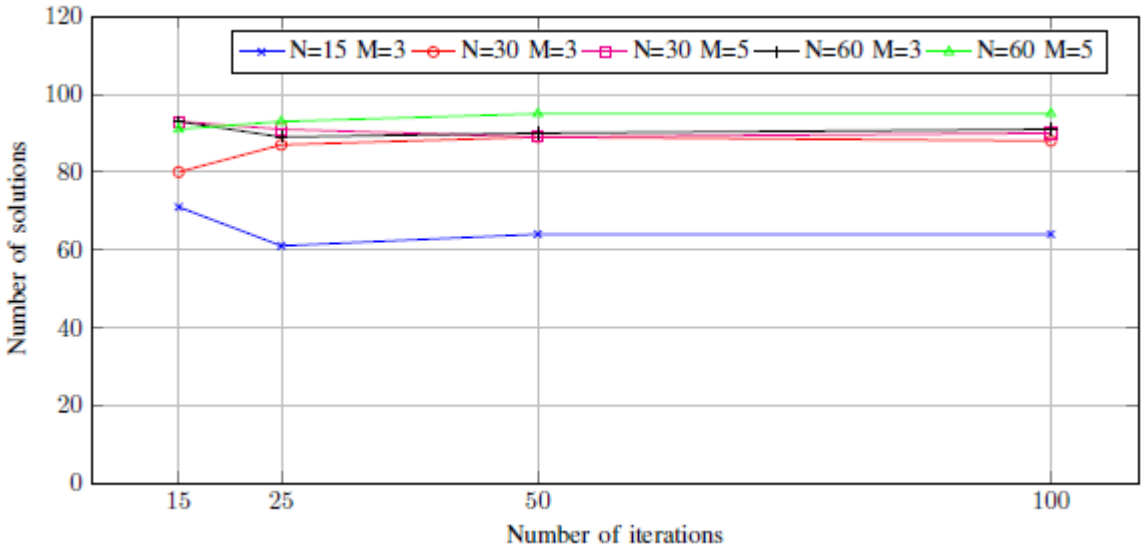

Figure III-22:GA-SLS Ensemble de solutions non-dominées par itérations pour cinq instances

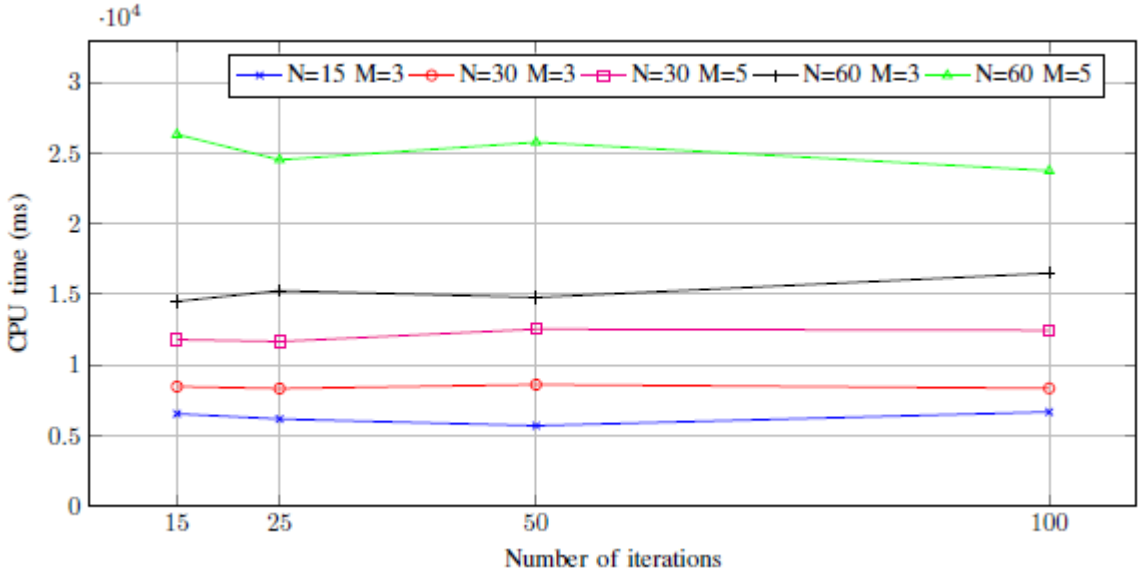

Figure III-23:GA-SLS temps écoulé par itération pour cinq instances

III.5.3.2 GA-SLS en utilisant le Benchmark généré aléatoirement :

Lors de l'application de la méthode GA-SLS sur le benchmark aléatoire, nous montrons sur la Figure III-24 l'évolution du nombre de solutions optimales non dominées par itération (15, 25, 50, 100...).Le nombre des solutions optimales se situe entre 50 et 95 concernant les cinq différentesinstances étudiées.

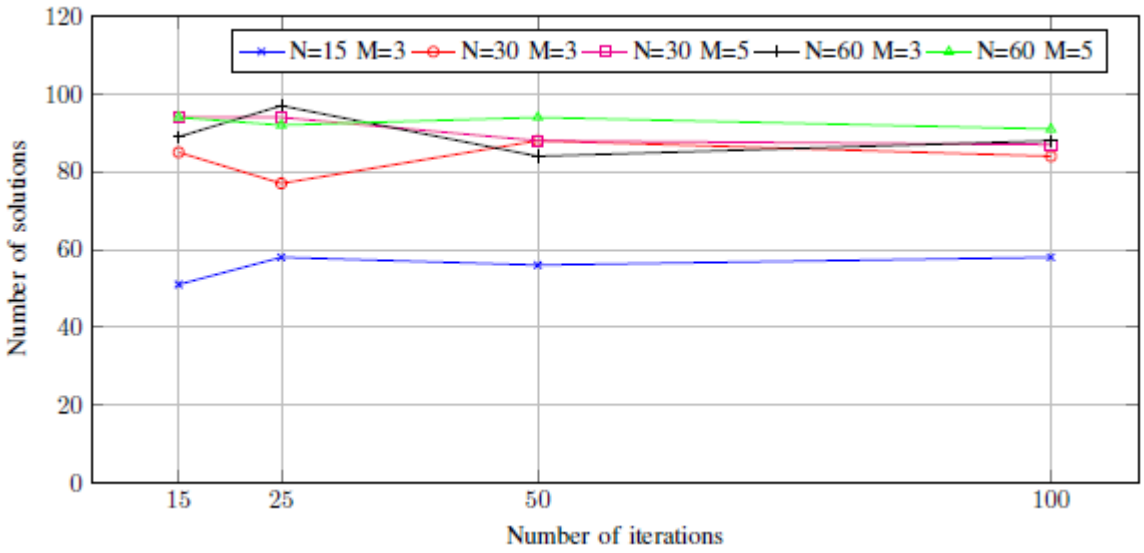

Figure III-24: GA-SLS : Ensemble non-dominées par itération pour cinq instances

La Figure III-25 donne la variation du temps écoulé par itération pour les cinq instances étudiées.Nous pouvons donc dire que le temps de l'exécution de la méthode GA-SLS est très intéressant.La GA-SLS réussit à trouver de bons résultats en peu de temps.

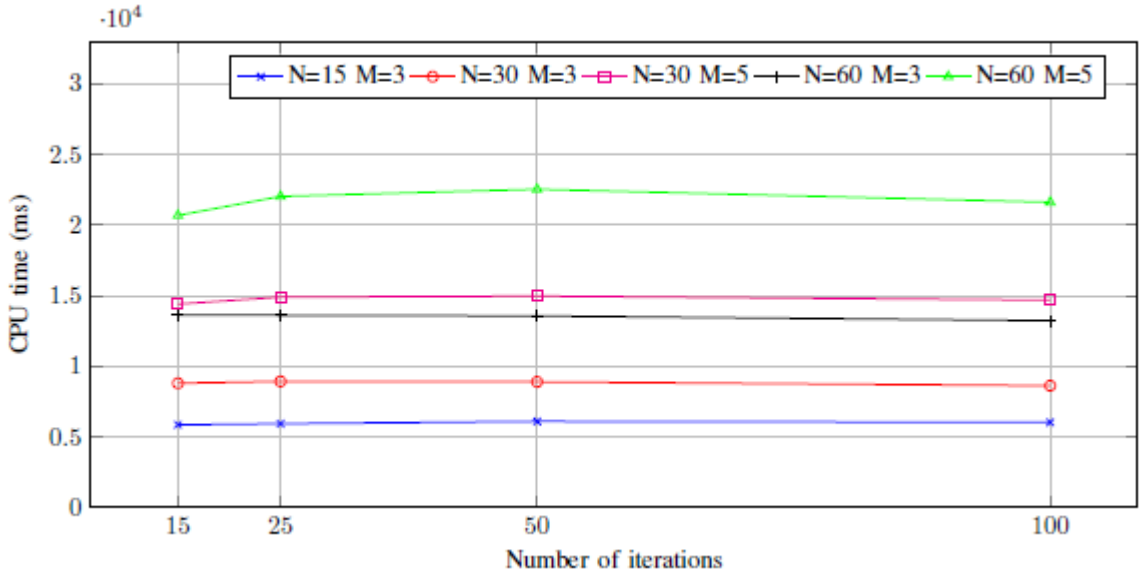

Figure III-25: GA-SLS : Temps écoulé par itération pour cinq instances

III.5.4 Etude comparative & discussion

Nous essayons dans cette section de comparer les approches proposées : SLS, GA et GA-SLS pour la composition de services web en se basant sur les résultats numériques trouvés. D'après les Figures III-11 et III-14, nous pouvons dire que le nombre de solutions optimales trouvées par SLS est grand.Par exemple pour l'instance de N=60 et M=5 du benchmark cité dans

[84]et[85], nous voyons que 1179 solutions sont trouvées dans l'itération 100 et 1386 solutions sont trouvées en utilisant le benchmark aléatoire.

Pour le résultat du GA, nous voyons que le nombre de solutions optimales trouvé est globalement compris entre 40 et 70 en utilisant le benchmark de [84], [85] et 35 et 70 en utilisant la référence aléatoire. Pour GA-SLS, les résultats montrent que le nombre solutions optimales est globalement compris entre 61 et 95 en utilisant le benchmark de [84], [85] et 51 et 95 en utilisant le benchmark aléatoire. Sinon, d'après les Figures III-12 et III-15, nous pouvons dire que le nombre de solutions optimales et le temps consommé par SLS est intéressant.

Par exemple pour l'instance de N=60 M=5, pour le benchmark cité dans [84], [85] nous voyons que 13588 millisecondes se sont écoulées en 100 itérations, cependant 16975 millisecondes sont écoulées pour le benchmark aléatoire pour 100 itérations.

D'après les Figures III-18, III-21, II-23 et III-25, GA-SLS donne des résultats prometteurs.Les algorithmes SLS et GA sont exploités pour obtenir des résultats intéressants sur le nombre de solutions optimales et le temps d'exécution écoulé. L'approche GA-SLS nous permet de trouver des solutions optimales mis en un temps considérablement réduit.Il permet d'élargir l'espace de recherche qui peut nous conduire à trouver les solutions optimales. L'étude comparative entre GA, SLS et GA-SLS permet de dire que l'approche GA-SLS donne un résultat encourageant pour le problème d'optimisation de la composition des services web.

III.6Conclusion

Dans ce chapitre, nous avons proposé trois méta-heuristiques pour résoudre le problème de composition de service web qui sont : la Recherche localeStochastique (SLS), l'algorithme génétique (GA) et l'algorithme hybride GA-SLS qui combine les algorithmes SLS et GA.L'approche GA-SLS proposée nous permet de trouver un résultat apprécié pour le problème d'optimisation de la composition des services qui satisfait les contraintes du modèle QoS.La méthode GA-SLS a été évaluée sur deux types de benchmarks.Les résultats numériques trouvés sont encourageants et démontrent les bénéfices de la méthode GA-SLS.

Dans le chapitre suivant, nous allons présenter l'étude de la méthode de recherche locale hybride multi-objectifs pour la composition optimale des services Web.

Chapitre IV :

Contribution 2: une approche hybride de recherche à voisinage variable pour le problème de composition des services web

IV. Contribution 2: une approche hybride de recherche à voisinage variable pour le Problème de composition des services

IV.1 Introduction

Un grand nombre d'organisations et les entreprises ont adopté des architectures orientées services pour implanter leur cœur de métier et externaliser l'application sur Internet.Différentes plates-formes et plusieurs langues ont été fournis, tels que Universal Description Discovery et Intégration (UDDI) [86], langage de description de services Web (WSDL) [87], protocole d'accès simple aux objets (SOAP) [88] et une partie de l'ontologie DAML-S [89] (ServiceProfile et ServiceGrounding).

La composition des services web est l'une des plus importantes problèmes dans le développement d'architectures orientées services. C'est un domaine actif d'efforts de recherche et de développement pour l'intégration et l'interopérabilité des applications.

Dans ce chapitre, un algorithme multi-objectif de recherche à voisinage variable (VNS) et méthode de recherche locale stochastique (SLS) ont été utilisés dans nos approches proposées pour résoudre le problème des services composition. Le VNS est une méta-heuristique simple mais efficace pour un problème d'optimisation combinatoire et globale [64].

Nous proposons et évaluons l'approche hybride proposée MOVNS-SLS qui combine les deux algorithmes VNS et SLS en introduisant le concept suivant : changement de voisinage dans la recherche, en utilisant systématiquement une méthode de recherche locale stochastique locale et un outil d'optimisation multi-objectifs. L'approche proposée est évaluée sur certains ensembles de données pour mesurer son efficacité.
Nous avons utilisé le même modèle QoS que la première contribution pour la résolution du problème de la composition services web.

IV.2 La Recherche A Voisinage Variable VNS : Un Aperçu

IV.2.1 Problème d'optimisation linéaire

Nos travaux se placent dans le contexte de la résolution d'un problème d'optimisation linéaire. Il s'agit de trouver une solution qui minimise (ou maximise) une fonction linéaire, en satisfaisant un ensemble de contraintes linéaires. Dans toute la suite, nous supposerons (sauf mention contraire) que le problème à résoudre est un problème de minimisation. Un problème P peut être représente par le couple < S, *f*>, où S est l'ensemble des solutions réalisables

(c'est-à-dire satisfaisant l'ensemble des contraintes), et $f : S \rightarrow R$ est la fonction à minimiser (traduisant l'objectif du problème). A chaque solution $x \in S$ est associé un coût $f(x)$. Une solution optimale, généralement notée $x^* \in S$ vérifie l'inégalité suivante :

$\forall x \in S, f(x^*) \leq f(x)$, $f^* = f(x^*)$ est la valeur optimale.

L'ensemble des solutions optimales de P est noté $S^* = \{x \in S \mid f(x) = f^*\}$.

IV.2.2 Notion de voisinage

Les heuristiques et les méta-heuristiques manipulent une ou plusieurs solutions, et mettent souvent en œuvre des stratégies de mouvement d'une solution vers une autre. C'est en particulier le cas avec les techniques basées sur la recherche locale. Cette notion de mouvement d'une solution vers une autre est liée à la définition du voisinage de la solution considérée. Le voisinage d'une solution x correspond à l'ensemble des solutions accessibles depuis x à l'aide d'un mouvement élémentaire. Plus formellement, soit $< S, f >$ l'instance du problème d'optimisation P traité, et soit $d : S \times S \rightarrow R^+$ une fonction qui calcule la distance entre deux solutions de S. Un voisinage d'une solution $x \in S$, est un ensemble $V(x) \subseteq S$ des solutions "proches" de x, c'est-à-dire à une distance inférieure à ε donne :

$$V(x) = \{y \in S \mid d(x,y) \leq \varepsilon, \varepsilon \in R^+\}$$

IV.2.3 Optimum local

Pour un problème d'optimisation P donné, un optimum local associé à un voisinage IV est une solution $x^L \in S$ tel que $f(x^L) \leq f(x)$, $\forall x \in V(x^L)$.

La notion d'optimum local représente une limite dans l'utilisation et l'efficacité des méthodes de recherche locale que nous abordons dans la suite. Cependant, comme nous le verrons un peu plus loin, différentes techniques permettent de sortir de ces optima locaux et de relancer la recherche[90].

IV.2.4 Schéma de base

La recherche à voisinage variable (Variable Neighborhood Search, VNS), introduite dans [64], est une méta-heuristique qui utilise systématiquement plusieurs structures de voisinages (voir Figure IV.1), dans le but d'échapper aux minima locauxet fuir des vallées qui les contiennent.

Le voisinage d'une solution est un sous-ensemble de solutions qui peuvent être obtenues par un certain nombre de transformations de données.

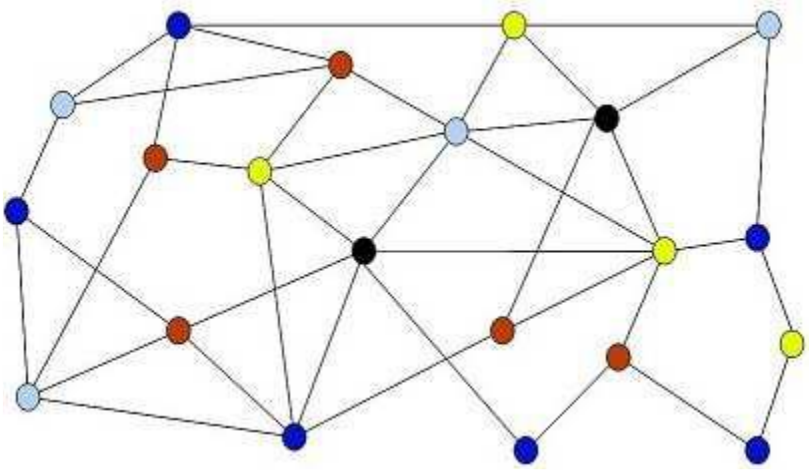

Figure IV-1:structure de voisinage.

Elle comporte trois phases principales [91]:

- une phase stochastique (dite phase de shaking) pour éviter des cycles infinis qui pourraient se produire si une règle déterministe est appliquée,
- une phase de recherche locale qui permet de converger vers un optimum local,
- une phase de changement de voisinage qui permet de diversifier l'exploration de l'espace de solutions afin d'accéder à un plus grand nombre de régions intéressantes.

Soit L = {N1, . . . ,N_{kmax}} une liste finie de structures de voisinage, ordonnée par ordre croissant de taille, où $N_k(s)$ (k = 1 . . . k_{max}), désigne l'ensemble des solutions voisines de s dans N_k. Dans la plupart des méthodes de recherche locale k_{max} = 1.

L'algorithme IV-1 donne le pseudo-code de VNS, avec N_k (k = 1, ... , k_{max}) la structure de voisinage d'index k et t_{max} le temps maximal alloué à la recherche (*TimeOut*). L'algorithme démarre d'une solution initiale s souvent générée aléatoirement. À chaque itération de la boucle des lignes (5-8), une solution s′ est sélectionnée aléatoirement (phase de shaking) dans le voisinage N_k de s (s′ ∈ $N_k(s)$) (ligne 6). Une recherche locale est appliquée à s′ et produit une nouvelle solution s″ (ligne 7). La procédure de changement de voisinage est ensuite appelée avec les paramètres s, s″ et k (ligne 8). Si s′est de meilleure qualité que s (ligne 15), alors s′ devient la solution courante et l'index du voisinage est réinitialisé à 1 (ligne16). Sinon, l'index du voisinage est incrémenté pour passer au voisinage N_{k+1} (ligne 19). L'algorithme s'arrête dès qu'on atteint le *TimeOut*[91].

Algorithme IV-1 :Pseudo-code de la recherche VNS.

```
Fonction VNS(s, kmax, tmax);
2 début
3       répéter
4           k ← 1;
5           répéter
6               s′ ← Shake(s,Nk) /* Phase de Perturbation */ ;
7               s″ ← RechercheLocale(s′) /* Phase d'une recherche locale */ ;
8               NeighbourhoodChange(s, s″, k) /* Changement de voisinage */ ;
9           jusqu'à k = kmax;
10          t ← CpuTime();
11      jusqu'à t > tmax;
12 retourner s ;
13 Procédure NeighbourhoodChange(s, s′, k);
14 début
15      si (f (s′) < f (s)) alors
16          s ← s′ ; k ← 1 /* Effectuer le mouvement et réinitialiser le voisinage */ ;
17      sinon k ← k + 1 /* Prochain voisinage */ ;
```

La recherche VNS a été utilisée avec succès pour résoudre une variété de problèmes d'optimisation. L'un des premiers problèmes traités par cette approche est le problème du voyageur de commerce[64]. De nombreuses variantes de VNS ont été proposées dans la littérature.

Nous passons en revue les plus connues dans les sections suivantes.

IV.2.5 Recherche à voisinage variable réduite

Cette variante de VNS a été proposée et étudiée dans [92]. Elle consiste à explorer les différents voisinages de manière complètement aléatoire. L'algorithme RVNS reprend globalement le même schéma basique d'une recherche VNS à l'exception de la phase d'amélioration par une recherche locale (algorithme 8). A chaque itération, une nouvelle solution est choisie aléatoirement dans le voisinage courant N_k. La nouvelle solution est comparée à la solution courante. Si elle est de meilleure qualité, la solution courante est mise à jour par la nouvelle solution et la recherche continue.

Cette méthode peut être utilisée pour résoudre des problèmes de (très) grande taille pour lesquels l'utilisation d'une recherche locale peut s'avérer très coûteuse lors de la phase

d'amélioration. Les auteurs dans [93] ont proposé une méthode hybride intégrant une recherche VNS réduite dans un algorithme d'essaims de particules. Les auteurs dans[94] ont proposé une approche basée sur la recherche VNS réduite pour résoudre un problème connu sous la terminologie anglaise de multi-level lot sizing problem[91].

IV.2.6 Recherche à voisinage variable

L'exploration et l'évaluation efficace des régions de l'espace de recherche contenant de bonnes solutions(et éventuellement une solution optimale) qui sont très éloignées de la meilleure solution courante est l'un des défis majeurs des méta-heuristiques en général et de la recherche VNS en particulier. La recherche VNS biaisée (connue sous la terminologie anglaise de Skewed VNS) a été proposée notamment pour adresser cette limitation. L'idée est de centrer la recherche autour d'une solution de bonne qualité, qui ne soit pas nécessairement meilleure que la solution courante. Cela est fait en intégrant dans la fonction d'évaluation une distance entre la meilleure solution courante et la solution à évaluer, notamment pour autoriser des mouvements qui ne soient pas nécessairement de meilleure qualité[91].

L'algorithme IV-2 illustre le pseudo-code de SVNS. La différence de cette variante par rapport au schéma basique de VNS est l'étape de mise à jour de la meilleure solution s^*(ligne 9) qui est insérée entre la recherche locale et le changement de voisinage (pour ne pas perdre la meilleure solution). Le changement de la solution courante inclut la fonction de distance entre la nouvelle solution et la solution courante (ligne 17). Cette méthode a été utilisée avec succès dans[95]pour résoudre le problème de k-cardinalité dans un graphe.

Algorithme IV-2 : Pseudo-code de la recherche VNS biaisée.

```
1 Fonction SVNS(s, kmax, tmax);
2 début
3 répéter
4 s∗ ← s;
5 k ← 1;
6 répéter
7 s′ ← Shake(s,Nk ) ;
8 s″ ← RechercheLocale(s′) ;
9 keepBest(s∗, s) ;
10 NeighbourhoodChange(s, s″, k)
11 jusqu'à k = kmax;
12 t ← CpuTime();
```

13 jusqu'à t > tmax;
14 retourner s ;
15 Procédure NeighbourhoodChange(s, s′, k);
16 début
17 si (f (s′) − α × d(s′, s) < f (s)) alors
18 s ← s′ ; k ← 1 ;
19 sinon k ← k + 1

IV.2.7 Recherche à voisinage variable avec décomposition

Les grands voisinages constituent une alternative très intéressante pour diversifier la recherche et assurer la convergence vers un optimum local de (très) bonne qualité. Toutefois, sur des problèmes de grande taille, l'exploration de ces voisinages peut s'avérer très coûteuse en temps et peut ralentir considérablement l'amélioration de la solution courante. Pour privilégier l'exploration de petits voisinages, la recherche à voisinage variable avec décomposition (connue sous la terminologie anglaise de Variable Neighborhood Decomposition Search VNDS)[96]a été proposée, et vise à réduire l'espace de solutions parcouru par la recherche locale comme le montre l'algorithme IV-3.

Algorithme IV-3 : Pseudo-code de la recherche VNS avec décomposition.

1 **Fonction** VNDS(s, k_{max}, t_{max});
2 **début**
3 **répéter**
4 k ← 1;
5 **répéter**
6 s′ ← Shake(s,N_k) ;
7 y ← s′\s ;
8 y′ ← VNS(y, k, t_{max}) ;
9 s″ ← (s′\y) ∪ y′ ;
10 NeighbourhoodChange(s, s″, k)
11 **jusqu'à** k = k_{max};
12 t ← CpuTime();
13 **jusqu'à** t > t_{max};
14 Retourner s ;

Parmi les applications de cette variante, nous pouvons citer le travail original de [96] pour la résolution du problème du P-médiane, ou encore le travail réalisé dans[97], où les auteurs

proposent une heuristique hybride pour la résolution du problème en variables 0-1 mixtes en utilisant le principe de la recherche VNS avec décomposition.

IV.2.8 Recherche à voisinage variable primale-duale

La recherche à voisinage variable primale-duale a été introduite dans[98]dont l'objectif est de donner une évaluation sur la qualité des solutions obtenues. Les méta-heuristiques ne donnent aucune garantie sur l'optimalité de la solution ou une approximation de cette dernière. Généralement, il est difficile de donner avec précision l'écart entre la valeur de la solution finale retournée par l'approche heuristique et la valeur optimale.

Pour cela, les auteurs appliquent, dans une première phase, un algorithme VNS pour obtenir une solution faisable du problème primal. Cette solution sera utilisée dans une deuxième phase pour déduire une solution du problème dual. Enfin, et selon le problème traité, les conditions des écarts de complémentarité peuvent être utilisées pour améliorer les bornes obtenues, sachant qu'il est possible d'utiliser d'autres variantes de VNS pour les deux phases.

Cette variante a été appliquée initialement dans [98] pour résoudre un problème de localisation. L'approche a permis d'obtenir des solutions approchées avec une erreur bornée à 4% sur des instances intéressantes.

IV.2.9 Recherche à formulation variable

La recherche à formulation variable a été introduite par Mladenovic et al. [95], et repose sur l'observation suivante : de nombreux problèmes d'optimisation peuvent être formules différemment, et il n'est pas toujours aisé de savoir si une formulation domine les autres sur toutes les instances du problème traite. L'idée est donc d'utiliser plusieurs formulations disponibles, et de changer de formulation en fonction de l'état de la recherche.

Cette méthode est intéressante en particulier pour les problèmes pour lesquels il est facile de transformer une solution d'une formulation vers une autre, et pour lesquels les techniques de recherche locale se comportent différemment selon les formulations utilisées. Cette approche a également été appliquée récemment par [99] pour résoudre un problème de découpe. La mise en place de cette méthode demande quelques ajustements de sorte à considérer les différentes formulations du problème. Une façon simple de procéder est de suivre la démarche proposée dans [103], qui consiste à ajouter dans chacune des trois étapes principales de la RVIV (la méthode de perturbation, la recherche locale et la fonction de changement de voisinage) l'utilisation de la fonction d'acceptation décrite dans l'algorithme IV.4.

Dans cet algorithme, n_f désigne le nombre de formulations utilisées pour résoudre le problème, alors que la notation $f_i(x)$ est associée à la valeur de la solution x en considérant la

formulation f_i. Cette fonction permet donc de rejeter un mouvement si la solution courante n'est pas améliorée pour au moins une formulation.

Algorithme IV-4 : Fonction d'acceptation de mouvement avec plusieurs formulations.

```
1 Fonction Acceptation(s, s′, nf);
2 début
3     i ← 1;
4     tant que i ≤ nf faire
5         si fi(s′) <fi(s) alors
6             retourner Vrai ;
7         sinon si fi(s′) >fi(s) alors
8             retourner Faux;
9     i ← i + 1;
10 Retourner Faux ;
```

IV.2.10 Recherche à voisinage large

Dans de nombreux problèmes d'optimisation, il est possible d'avoir plusieurs formulations différentes, et il n'est pas toujours aisé de savoir si une formulation domine les autres sur toutes les instances du problème traité.

La recherche à formulation variable a été introduite dans[100], et repose sur l'idée d'utiliser plusieurs formulations disponibles, et de changer de formulation en fonction de l'état de la recherche. Cette méthode est intéressante si la recherche locale utilisée au sein de VNS se comporte différemment sur les différentes formulations et qu'il est facile de transformer une solution d'une formulation vers une autre.

Cette variante a été appliquée dans [99] pour résoudre un problème de découpage. La mise en place de cette méthode demande quelques ajustements de sorte à considérer les différentes formulations du problème. Les auteurs ont ajouté dans chacune des trois étapes principales de VNS (la phase de perturbation, la recherche locale et la fonction de changement de voisinage) l'utilisation de la fonction d'acceptation décrite par le pseudo-code de l'algorithme.

IV.3 Processus général de la méthode de recherche à voisinage variable VNS

Nous appliquons une approche VNS multi-objectifs pour trouver un plan optimal pour la composition des services web.Dans cette résolution de problème, le résultat prévu est un ensemble de solutions.Cette approche VNS comprend trois étapes principales :

1) Génération de la solution initiale : dans cette étape, nous créons une affectation de vecteur binaire qui représente la solution initiale.A partir de là, nous créons une population initiale de solutions candidates.
2) Recherche et évaluation de voisinage : de manière itérative, nous générons l'ensemble de voisinage de la solution courante.Nous avons trois types de structures de voisinage qui sont :
   a) **Voisinage 1 :** "Génération par swapping (échange)" dans cette étape, nous appliquons l'opération d'échange entre les différentes combinaisons de la solution actuelle.
   b) **Voisinage 2 :** "Génération par décalage de deux bits à gauche" : cette génération de voisinage est effectuée par le décalage vers la gauche de deux bits.
   c) **Voisinage 3 :** "Génération par décalage de deux bits à droite" : pour générer l'ensemble des solutions voisines, nous utilisons le décalage vers la droite en deux bits.

L'évaluation des solutions candidates se fait à l'aide de fonctions objectifs.

3) Sélection des meilleures solutions : nous sélectionnons la meilleure solution de la collection actuelle.Ensuite, nous remplaçons l'actuelle solution par la meilleure solution trouvée en appliquant la méthode de recherche locale sur la population actuelle.Si la courante solution est dominée par la meilleure solution, nous revenons au premier voisinage (k=1).Si l'actuelle et la meilleure solution ne sont pas dominées, elles seront ajoutées à une liste de Solutions Pareto optimales et nous passons au voisinage suivant.

Le processus est répété jusqu'à un certain nombre d'itérations appelé "*max_iter*" est atteint.

IV.4 Approche hybride multi-objectifs de recherche à voisinage variable MOVNS-SLS

Nous présentons dans cette section une méta-heuristique d'optimisation multi-objectif pour le problème de composition de service web. Cette méthode est basée à la fois sur les algorithmes VNS et SLS, elle traite simultanément plusieurs fonctions objectifs du modèle QoS pour résoudre le problème d'optimisation de la composition des services.

La nouvelle approche MOVNS-SLS utilise un SLS adapté comme recherche locale pour obtenir la meilleure solution dans chaque espace de voisinage d'une solution courante. Nous notons que SLS est une méta-heuristique de recherche locale déjà étudiée utilisée pour plusieurs problèmes d'optimisation [1]. Elle utilise des décisions aléatoires tout en cherchant des solutions à plusieurs problèmes dans divers domaines [101]. Le SLS fonctionne sur une solution initiale aléatoire. Alors, il effectue un certain nombre d'étapes locales qui combinent les stratégies de diversification et d'intensification pour localiser une bonne solution. La phase

de diversification est appliquée avec une probabilité *Wp > 0* et la phase d'intensification avec une probabilité *(1 -- Wp)*. Le *Wp* est une probabilité fixée empiriquement. Les étapes SLS sont données dans l'algorithme IV-5.

La méthode MOVNS-SLS est esquissée dans l'Algorithme IV-6.

Le paramètre max_iter représente le nombre d'itération d'un programme d'exécution, sol_init est la solution initiale générée par une méthode Random sous la forme d'un vecteur binaire où chaque cellule binaire représente un service (1 : le service fait partie de la composition initiale la solution.0 : il ne fait pas partie).

Algorithme IV-5 : MOSLS

```
1: Entrée: Sol_Curr; k
2: Initialiser Wp;  0 ≤Wp ≤1, max_iter
3: générer une solution initiale aux contraintes Sol_init
4: Solution actuelle: Sol_Curr Sol_init
5: nbr_iter ← 0
6: Tandis que (nbr iter <max iter) faire
7:          Random r = Rand.Nextfloat(0; 1);
8:          si (alors r < Wp) alors
9:              si (k == 1) alors
10:                 Set_Neighb ← Neighborhood_by_swapping();
/*Définissez l'ensemble des solutions voisines en échangeant */
11:                 Sol_Curr ← Random_choice_neighb(Set_Neighb);
12:             fin-si
13:             si (k == 2) alors
14:                 Set_Neighb ← Neighb_by_shift_two_bits_left();
15:                 Sol_Curr ← Random_Choice_Neighb (set_Neighb);
16:             fin-si
17:             si (k == 3) alors
18:                 Set_voisin ← Neighb_by_shift_two_bits_right ();
19:                 Sol Curr ← Random_Choice_Neighb (set_Neighb);
20:             fin-si
21:         else Sol_curr ← Déterministic_choice (set_voisin);
22:         fin-si
23:         si (dominate (Sol_Curr_P, Sol_Curr)) alors
                        Sol_Curr ← Sol_Curr_P
24:         fin-si
```

```
25:         nbr_iter ← nbr_iter + 1
26:    fin-tandis que
27: Retourner  la meilleure solution trouvée: Sol_Curr
```

AlgorithmeIV-6 : MOVNS-SLS

```
1: Initialiser Max_iter
2: Générez une solution initiale aux contraintes Sol_init.
3: Solution actuelle: Sol_Curr ← Sol_init
4: itr_nbr ← 1
5: tandis que (itr nbr <max iter) faire
6:          K ← 1
7:          tandis que (k <k max) faire
8:                  Sol_Curr ← SLS (Sol_Curr; K)
9:                  Si dominate (Sol_Curr_P; Sol Curr) alors
10:                 Sol_Curr ← Sol_Curr /* Changer la résolution actuelle*/
11:                 K ← 1 /*Passage au voisinage initial*/
12:                 Sinon
13:                 K ← K + 1 /* Passage vers le voisinage suivant*/
14:                   Si NO(Dominate(Sol_curr, Sol_curr_p)) alors
15:                   Add (Sol_Curr_P; Liste_Solution_Paretoo).
                      /* Ajouter la solution Sol_Curr_P à la liste de solutions Pareto */
16:                   fin-si
17:                 fin-si
18:         Fin-tandis que
19: itr_nbr ← itr_nbr + 1
20: Fin-tandis que
21: Ajouter (Sol Curr; Liste_Solution_Pareto)
22: Return List_Solution_Paretoo. /* La liste des solutions optimales de Pareto */
```

Dans ce qui suit, nous donnons les résultats numériques trouvés lors de l'application des trois approches pour les services web composition : le MO-VNS basique, le MO-SLS et la nouvelle méthode proposée MOVNS-SLS.

Dans la section suivante, nous allons présenter les résultats numériques des trois approches

IV.Expérimentation Et Résultats

Dans cette section nous présentons les résultats obtenus avec les approches proposées et décrites précédemment. Tous les algorithmes ont été implémentés avec le langage Java.

Les méthodes proposées sont évaluées sur l'ensemble de données du benchmark cité dans [84], [85] et un ensemble de données du benchmark généré aléatoirement. Les services candidats sont classés en plusieurs classes où chaque classe a la même fonctionnalité. Chaque service Web est caractérisé par ses entrées/sorties et ses critères QoS (temps, coût, disponibilité et réputation).Cela a été conclu par une publication dans la conférence international « ISIA 2020 : 4th International Symposiumon Informatics and its Applications » dont le titre est « An effective multi-objective hybrid local search method for the optimal web services composition ».

Considérons que N est le nombre de candidats services et M est le nombre de tâches. Afin d'évaluer l'approche proposée, nous réalisons une série de tests. Nous considérons que nous avons N services et M tâches,$N \in \{15, 30, 60\}$ et $M \in \{3, 5\}$. Chaque tâche a le même nombre de services candidats.

Dans ce qui suit, nous présentons les résultats numériques trouvés lorsqu'on applique les trois approches pour la composition des services web: Le MO-VNS basique, le MO-SLS et la nouvelle méthode hybride proposée MOVNS-SLS.

Pareillement que l'expérimentation de la première contribution, nous avons étudié 5 instances: instance01 = ((n, m) = (15,3)), Instance02 = ((n, m) = (30,3)), instance03 = ((n, m) = (60,3)), instance04 = ((N, m) = (30,5)), instance05 = ((n, m) = (60,5)).

IV.1 Impact du nombre d'itérations sur le nombre de solutions

Pour avoir plus de détails sur l'évolution du nombre des solutions optimales trouvées par itération, nous avons évalué les méthodes proposées sur l'instance 01 (n = 15, m = 3).

Dans la Figure IV-2, nous dessinons trois courbes pour les trois méthodes,nous pouvons faire en sorte que les solutions optimales soient trouvées en $5^{ème}$ itération.

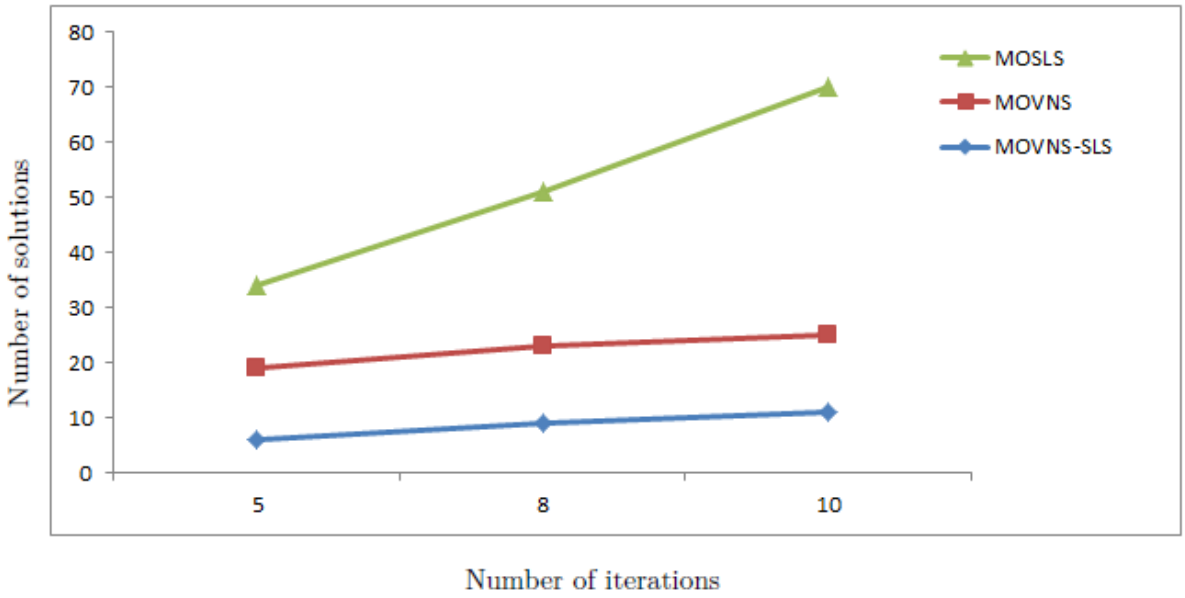

Figure IV-2:solutions non-dominées par itération pour l'instance 01

IV.2 Résultats numériques via le benchmark aléatoire (Random)

Le tableau IV-1 recapitalise les résultats numériques trouvés par les trois méthodes sur les cinq instances étudiées avec le benchmark aléatoire. Il nous montre pour chaque méthode le nombre de solutions trouvées et le temps d'exécution en millisecondes (CPU) consommé par la méthode.

Selon les résultats numériques trouvés en utilisant un benchmark généré par une méthode Random présenté par le Tableau IV-1, nous pouvons voir que pour instance02, MOVNS-SLS a trouvé 14 solutions distinctes dans L'itération 15 en 827 millisecondes. MOVNS-SLS a donné 22 solutions distinctes non dominées pour la 100ème itération en 4282 millisecondes. Cependant MOVNS a donné 14 solutions pour la 100ème itération en 3772 millisecondes et MOSLS a trouvé 438 solutions en 3864 millisecondes.

D'après le tableau IV-1, nous pouvons voir que lorsque nous avons N = 15 services et M=3 tâches, le MOVNS-SLS trouve 20 solutions à la 15ème itération en 90 millisecondes. Nous voyons aussi que 11 solutions ont été trouvées à la 100ème itération en 9939 millisecondes.

Pour l'instance de 30 services et 3 tâches (N=30 et M=3), MOVNS-SLS trouve 16 solutions optimales distinctes dans l'itération 15 en 580 millisecondes. Cependant les solutions de compromis ne violent aucune contrainte du modèle QoS.Pour cette instance, MOVNS-SLS donne 15 solutions optimales distinctes pour la 100ème itération en 4502 millisecondes. Par ailleurs, MOVNS trouve 12 solutions pour la 100ème itération en 10905 millisecondes mais MOSLS a trouvé 166 solutions en 952 millisecondes.

Lorsque nous considérons l'instance 3 (60 services et 3 tâches), MOVNS-SLS trouve 14 solutions en 1566 millisecondes à la 15ème itération. Nous pouvons remarquer que la méthode MOVNS trouve 16 solutions optimales distinctes à la 15ème itération en 873 millisecondes, et à la 100ème itération 14 solutions optimales sont trouvées en 3283 millisecondes. Pour la méthode MOSLS, nous pouvons voir que cette méthode a trouvé 98 solutions dans 1063 solutions et 678 solutions en 9370 millisecondes à la 15ème itération.

Pour l'instance 05 en utilisant 60 services et 5 tâches, chaque tâche peut être exécutée par 12 services et chaque service ne peut exécuter qu'une seule tâche. Nous pouvons constater qu'à la 15ème itération MOVNS-SLS trouve 22 solutions optimales distinctes en 2222millisecondes, et MOVNS trouve 24 solutions optimales en 1229 millisecondes et MOSLS trouve 235 solutions optimales distinctes en 1631millisecondes. Par ailleurs, à la 100ème itération, MOVNS-SLS trouve 25 solutions optimales distinctes en 10081millisecondes, et MOVNS trouve 22 solutions optimales en 5531millisecondes et MOSLS trouve 1386 solutions optimales distinctes en 16975millisecondes.

D'après les résultats numériques, nous pouvons dire que les résultats de MOVNS-SLS sont plus importants & encourageants que ceux de MOVNS et de MOSLS pour les instances étudiées.MOVNS-SLS donne les meilleurs résultats sur la recherche de solutions de compromis.
Nous dessinons les courbes dans les figures IV-3 pour montrer clairement les performances de MOVNS- SLS par rapport à MOVNS et MOSLS.

Instance	Number of Iterations	MOVNS-SLS		MOVNS		MOSLS	
		Nbr sol	cpu time	Nbr sol	cpu time	Nbr sol	cpu time
Instance 01	15	12	1696	13	1561	34	197
	25	10	2976	9	3731	79	454
	50	**13**	**4998**	10	5911	172	999
	100	13	12159	12	10905	166	952
Instance 02	15	14	827	15	1110	53	470
	25	13	1393	14	2144	147	550
	50	**15**	**3116**	14	1997	380	2706
	100	22	4282	14	3772	438	3864
Instance 03	15	18	2322	27	1356	27	219
	25	22	4028	18	2343	149	966
	50	**23**	**7159**	26	3868	177	2253
	100	23	15459	21	6276	659	5832
Instance 04	15	14	1566	16	873	98	1063
	25	15	2417	21	757	176	2140
	50	**10**	**6252**	16	1915	370	4193
	100	13	6466	14	3283	678	9370
Instance 05	15	22	2222	24	1229	235	1631
	25	21	3178	21	2201	421	3153
	50	**22**	**5975**	24	4313	493	4335
	100	25	10081	22	5531	1386	16975

Tableau IV-1:résultats numériques obtenus via le benchmark aléatoire

Dans l'ensemble d'après les résultats analysés du Tableau IV-1, nous pouvons dire que, en utilisant le benchmark aléatoire, les résultats trouvés par MOVNS-SLS sont meilleursque les deux MOVNS et MOSLS pour les instances étudiées.

Nous remarquons dans Figure IV-3 clairement les performances de MOVNS-SLS comparées aux méthodes MOVNS et MOSLS. MOVNS-SLS donne donc de résultats encourageants sur le nombre de solutions optimales distinctes de la composition de services web.

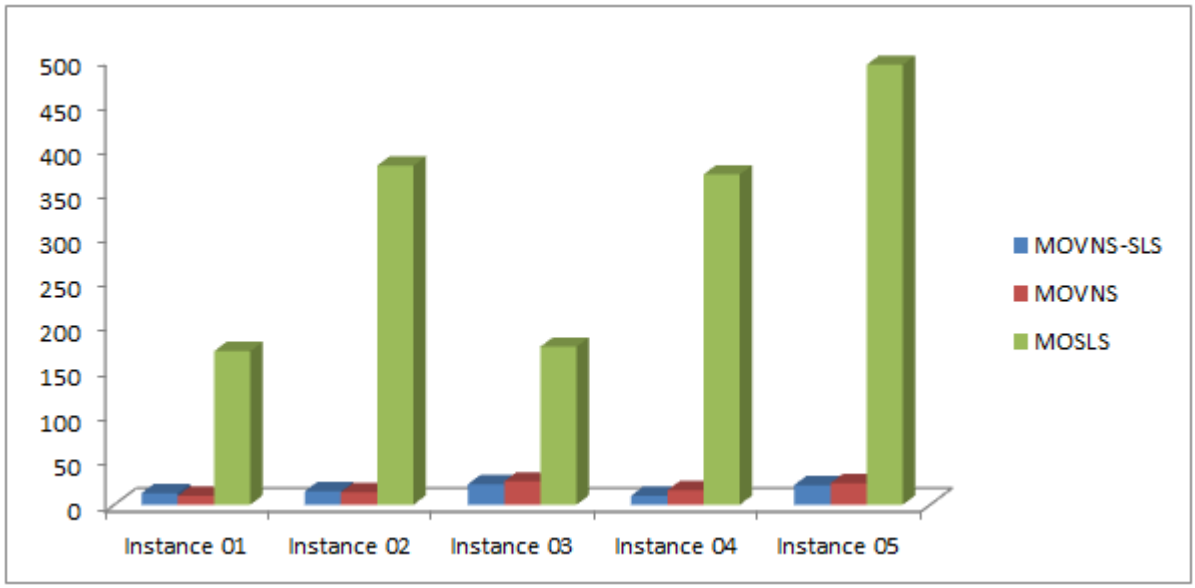

Figure IV-3: Solutions non-dominées pour le dataset Random

IV.3 Résultats numériques sur le benchmark cité dans*[84], [85]*

Dans cette section, nous présentons les résultats numériques obtenus lors de l'application des trois méthodes en utilisant le benchmark cité dans [84], [85].

Le nombre de solutions trouvées par MOVNS-SLS est raisonnable et est en temps utile par rapport à ceux trouvés par les méthodes de base SLS et VNS, comme précisé par les courbes de la figure IV-4.

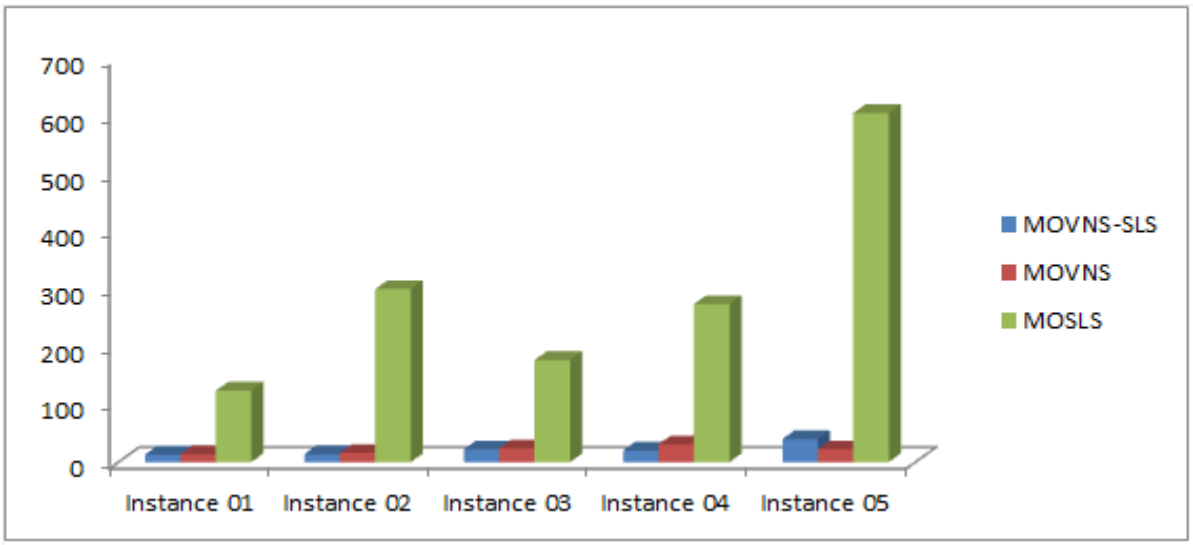

Figure IV-4: solutions non-dominées pour le dataset [80, 81]

L'approche hybride MOVNS-SLS génère un ensemble de solutions où chacune représente une composition optimale. A partir de cet ensemble de solutions optimales, notre approche sélectionne et propose l'un d'entre eux à l'utilisateur, il s'agit d'une solution non-dominée.

D'après le tableau IV-2, nous pouvons voir que les solutions compromis sont trouvées à la $5^{ème}$ itération. En outre, le tableau IV-2 montre la variation du nombre de solutions Pareto optimales distinctes par itération pour les cinq instances étudiées.

Pour l'instance 01 de ce deuxième benchmark, nous voyons dans Tableau IV-2 que MOVNS-SLS trouve 12 solutions dans la $15^{ème}$ itération en 1704 millisecondes tandis que MOVNS a trouvé 20 solutions en 90 millisecondes et MOSLS trouve 42 solutions en 70 millisecondes.

Nous voyons aussi dans la $100^{ème}$ itération, MOVNS-SLS trouve 11 solutions en 11487 millisecondes alors que MOVNS a trouvé 11 solutions en 9939 millisecondes et MOSLS en a trouvé 190 solutions en 1536 millisecondes.

En fait, nous trouvons des solutions non-dominées, puisqu'une fois les solutions sont atteintes, les autres sont maintenues.

Pour l'instance 02 et à l'itération 15, MOVNS-SLS a trouvé 14 solutions distinctes en 1074 millisecondes.MOVNS-SLS a donné 19 solutions non dominées pour la 100ème itération en 5062 millisecondes.Par l'exécution de 100 itérations, MOVNS a trouvé 15 solutions distinctes en 4502 millisecondes mais MOSLS a trouvé 310 solutions distinctes en 2580 millisecondes.Nous pouvons dire que les résultats trouvés par MOVNS-SLS sont encourageants.

Instance	Number of Iterations	MOVNS-SLS		MOVNS		MOSLS	
		Nbr sol	cpu time	Nbr sol	cpu time	Nbr sol	cpu time
Instance 01	15	12	1704	20	90	42	70
	25	13	2565	25	130	46	160
	50	**13**	**5190**	14	2568	124	240
	100	11	11487	11	9939	190	1536
Instance 02	15	14	1074	16	580	62	371
	25	19	1946	16	931	185	1180
	50	**14**	**3124**	16	2093	300	2190
	100	19	5062	15	4502	310	2580
Instance 03	15	22	1407	24	642	138	604
	25	15	1803	22	1073	163	715
	50	**22**	**3200**	24	2343	286	2047
	100	22	5577	24	4721	572	4941
Instance 04	15	14	1506	22	1430	89	1100
	25	16	2858	19	1876	274	2312
	50	**20**	**4332**	22	3577	284	2689
	100	14	5297	24	8990	1104	9991
Instance 05	15	21	2300	22	1051	121	910
	25	24	3496	22	1913	288	1382
	50	**45**	**86755**	31	4027	605	5454
	100	41	15450	33	8553	1179	13588

Tableau IV-2:Résultats numériques obtenus via le benchmark cité dans [84]&[85]

IV.4 Discussion

D'après les résultats numériques donnés dans les deux tableaux IV-1 et IV-2, nous voyons que le nombre de solutions optimales trouvées par MOVNS est compris entre 9 et 35 pour les différentes instances étudiées des benchmarks considérés. A l'exception du cas de l'instance(N=60 et M=5) en utilisant le benchmark cité dans [84]&[85], nous avons obtenu 45 solutions optimales à l'itération 50 et 41 solutions après l'exécution de 100 itérations. Lorsque nous considérons les résultats de MOVNS-SLS représentés dans les deux tableauxIV-1 et IV-2, nous remarquons que le nombre de solutions distinctes non dominées trouvées par MOVNS-SLS est compris entre 10 et 25 pour approximativement les instances étudiées à l'exception de l'instance 05 où 45 solutions non dominées sont trouvées à la 50èmeitérationlorsque nous utilisons le benchmark cité dans [84], [85].

IV.5 Résumé statistique

Pour démontrer les performances de la méthodeMOVNS-SLS proposée, un résumé statistique est donné dans le tableau comparatif IV-3. Pour chaque fonction objectif (coût, temps, disponibilité et réputation), nous donnons le minimum (Min), le maximum (Max), la moyenne (mean), la médiane (median), le premier voisinage (1st Qu) et le troisième voisinage (3rd Qu). Le Tableau IV-3 donne le résumé statistique pour les trois instances de notre expérimentation qui sont : N,M=(15,3), (30, 3), (60,5). D'après les résultats numériques, nous pouvons dire que l'approche proposée MOVNS-SLS est une approche importante et efficacepour l'optimisation de la composition de services web.Elle peut trouver un ensemble de solutions Pareto optimales dans un délai raisonnable.A partir de cet ensemble, l'utilisateur peut choisir la solution optimale qui répond le mieux à ses conditions.

dataset N=15 M=3 number of solutions:13 number of iteratios:50						
Summary	**Min**	**1st. Qu**	**Median**	**Mean**	**3rd Qu**	**Max**
Cost	200,00	220,00	150,00	175,00	185,00	215,00
Time	249,6	278.4	269,12	269,4	278,4	280,5
Avaibility	283.0	283.0	283.0	290.8	300.0	300.0
Reputation	243,90	250,1	253,8	256,8	257,80	262
dataset N=30 M=3 number of solutions:19 number of iteratios:100						
Summary	**Min**	**1st. Qu**	**Median**	**Mean**	**3rd Qu**	**Max**
Cost	95.0	126.2	150.0	147.1	160.0	195.0
Time	272,78	291,91	316,91	319,78	334,91	351,01
Avaibility	283.0	300.0	300.0	298.8	300.0	300.0
Reputation	249,2	251	254,3	257,4	257,5	259,7
dataset N=60 M=5 number of solutions:45 number of iteratios:50						
Summary	**Min**	**1st. Qu**	**Median**	**Mean**	**3rd Qu**	**Max**
Cost	165.0	215.0	250.0	247.1	270.0	345.0
Time	581.8	604.7	630.5	717.6	658.4	1968.5
Avaibility	462.0	483.0	483.0	482.7	483.0	500.0
Reputation	400.3	413.9	423.3	422.7	432.0	446.4

Tableau IV-3 : Résumé statistique sur certains ensembles de données

IV.6 Comparaison supplémentaire

Dans cette section, nous comparons notre méthode proposée (MOVNS-SLS) avec le travail présenté dans [1]. Ce travail [1] présente un modèle basé sur l'optimisation rentable pour résoudre le problème d'optimisation multi-objectifs de la composition de services web.Il est inspiré du comportement des abeilles oùl'algorithme de la colonie d'abeilles artificielles (ABC) est un algorithme évolutionnaire [2]. Comme le montre la figure IV-5, nous pouvons voir que le nombre de solutions Pareto-optimales trouvées par notre approche MOVNS-SLS est compris entre 17 et 22 alors que le nombre de solutions trouvées dans [1] c'est entre 19 et

21 solutions. Nous notons qu'après avoir exécuté 800 itérations, MOVNS-SLS réussit à obtenir 17 solutions mais l'approche de [1] a trouvé 21 solutions. Cette comparaison nous permet d'affirmer la performance de notre approche par rapport aux autres approches envisagées pour la composition des services web.Les résultats numériques de MOVNS-SLS sont prometteurs et intéressants.

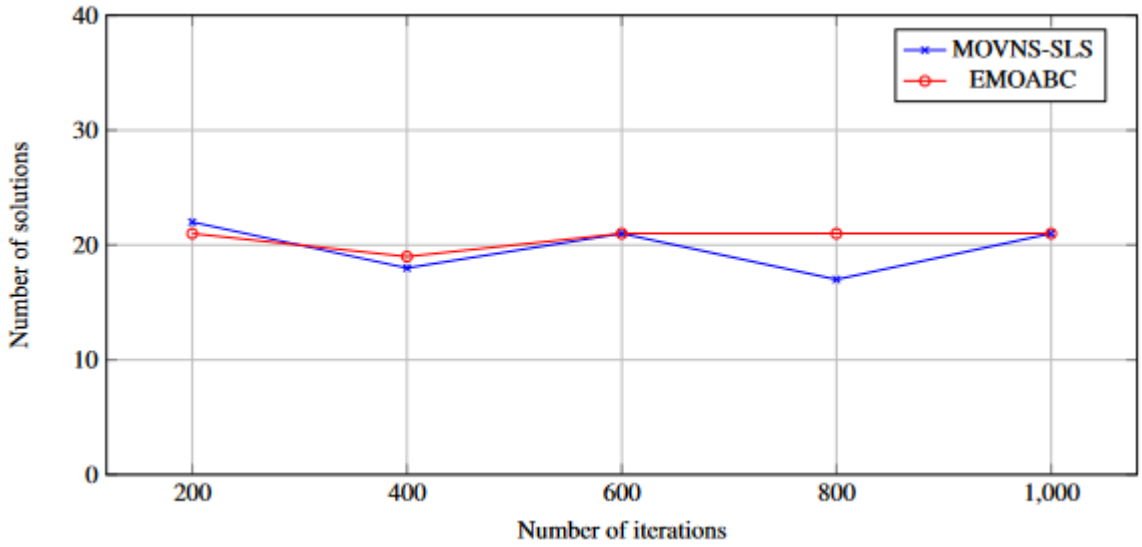

Figure IV-5: ensemble de solutions non dominées par itération pour l'instance N=50 et M=5

IV.7 Conclusion

Dans ce chapitre, nous avons proposé une approche méta-heuristique multi-objectifs pour résoudre le problème de composition des services basés sur le modèle QoS. Cette approche proposée est basée sur les deux algorithmes VNS et SLS, elle est évaluée sur deux ensembles de données et comparée avec l'approche des colonies d'abeilles artificielles (ABC). Les résultats numériques sont prometteurs et démontrent les bénéfices de cette approche.

Dans le chapitre suivant, nous allons proposer une méthode mimétique multi-objectifs pour résoudre le problème de composition de services web.

Chapitre V :

Contribution 3: une approche mémétique pour le Problème de composition des services web

V. Contribution 3: une approche mémétique pour le Problème de composition des services

V.1 Introduction

La technologie des services web est un domaine de recherche actif qui a attiré de nombreux chercheurs au cours des deux dernières décennies.La combinaison de services existants pour en former un nouveau en fonction des exigences données est l'un des avantages les plus prometteurs de la technologie des services Web.Cependant, combiner des services Web existants n'est pas une tâche facile.C'est l'une des difficultés les plus importantes dans le développement d'architectures orientées services.La composition de services web est le problème de création d'un service composite à partir d'un ensemble de services disponibles. Cela peut être réalisé en reliant et en combinant un ensemble de services élémentaires ou atomiques existants.

V.2 Contexte

La composition d'un service web peut être définie comme la combinaison de plusieurs services webdont l'objectif est de réaliser une tâche. Le besoin de composition de services web provient de la nécessité d'atteindre un objectif complexe qui ne peut pas être réalisé par un service autonome ou individuellement.Sur ce, nous avons proposé une contribution où trois approches multi-objectifs pour résoudre ce problème : la méthode de recherche locale MO-LS, l'algorithme génétique MO-GA & l'algorithme mimétique MO-MA.

Dans la section suivante, nous allons proposer une méthode LS multi-objectif pour résoudre le problème de la composition de services web.

V.3 Proposition d'une méthode MO-LS pour le problème de composition de services web

LS est une méthode méta-heuristique qui a déjà été étudiée pour résoudre plusieurs problèmes d'optimisation[1]. LS est une méthode de recherche itérative capable de trouver une solution optimale X pour un problème d'optimisation. Comme avec les méta-heuristiques, LS utilise une certaine fonction objectif f pour évaluer la qualité des solutions [102].

La méthode LS effectue un certain nombre d'étapes de recherche locale en commençant par une solution aléatoire initiale jusqu'à ce que la solution optimale soit atteinte où le but de ce mécanisme est d'optimiser (maximiser ou minimiser) un certain critère parmi un certain nombre de solutions candidates. La variante MO-LS que nous proposons pour résoudre le

problème d'optimisation de la composition de services web prend en compte simultanément les quatre fonctions objectifsprésentées dans la section IV.4.1. Le but est de trouver un ensemble de solutions appelées solutions Pareto-optimalesqui optimisant les quatre objectifs considérés. La méthode LS est utilisée pour trouver les solutions optimales en considérant simultanément plusieurs critères non fonctionnels. Plus précisément, nous proposons l'approche MO-LS pour trouver le plan d'exécution optimal de la composition des services web. Pour chaque plan d'exécution des services sélectionnés, un vecteur est produit. En accumulant les vecteurs de tous les plans d'exécution, nous obtenons une matrice S, où chaque ligne représente le vecteur sélectionné d'un plan d'exécution. Le résultat renvoyé par le MO-LS est un ensemble de vecteurs qui représentent un ensemble de plans d'exécution optimaux.

V.3.1 Le processus MO-LS

L'approche MO-LS est utilisée pour trouver l'ensemble des solutions optimales.Il comprend trois étapes principales :

1. *La génération de la solution initiale* : dans cette étape, nous créons un vecteur qui représente la solution initiale.Sur la base de cette solution, nous allons créer un ensemble de solutions candidates initiales.

2. *Recherche de voisinage et sélection du meilleur voisin* : à chaque itération, on dispose d'un ensemble de solutions voisines de la solution courante, en prenant celle maximisant localement le critère, nous sélectionnons la meilleure solution voisine de cette collection en utilisant la relation de dominance. Le voisin sélectionné remplacera la solution courante si et seulement si elle domine la solution courante.Dans notre cas, la méthode LS sélectionne l'ensemble des solutions non-dominées du problème étudié.

3. *Renvoyer la solution optimale* : la méthode LS renvoie le dernier ensemble actuel de solutions non-dominées qui constituent les solutions Pareto-optimales.

La méthode LS standard est esquissée dans l'algorithme V-1 et l'algorithme V-2.

Algorithme V-1 :Méthode de recherche localeLS
1 : S ← une solution réalisable arbitraire X
2 : **tant que** ($\exists S' \in X$ tel que $f(S_0)<f(S)$) **faire**
3: $S \leftarrow S_0$
4 : **fin tant que**
5 : **Renvoie**S

Algorithme V-2 Méthode de recherche locale multi-objectifs MO-LS

```
1 : S ← une solution réalisable arbitraire dans X.
2 : tant que (Nbr_it < max_itr) faire
3 :       tant que (∃S'∈X) faire
4 :           si (dominate(S', S)) alors
5:                S ← S'
6 :               List_pareto_optim_sol ← vide
7 :           sinon
8 :               si (NOTdominate(S, S')) alors
9 :                   Ajouter la solution S' à List_pareto_optim_sol
10 :              fin si
11 :          fin si
12 :      fin tant que
13 : fin tant que
14 : Ajouter la solution S à List_pareto_optim_sol
15 :Renvoie List_pareto_optim_sol
```

V.3.3 Les solutionsdevoisinage

Pareillement que les deux contributions nous représentons une solution candidate par un chromosome de gènes binaire.Chaque chromosome représente une composition de services.Une variable binaire prend la valeur 1 ou 0, 1 si le service correspondant fera partie de la composition optimale des services et 0 sinon.

L'exemple de la figure V-1 que déjà vu, représente une solution de la composition d'un ensemble de services utilisant 9 services et 3 tâches.Chaque tâche a trois services candidats et chaque service est représenté par une variable binaire. Trois variables binaires sont regroupées pour une tâche,seulement un service sera sélectionné pour une tache dans la composition.Chaque solution correspond à une matrice X telle que présentée dans la section II du chapitre 2.Sur la figure V-1, on peut dire que le service S_2 (la valeur du bit vaut 1) est alloué à la tâche T_0.Le service S_3 est affecté à la tâche T_1.La tâche T_2 est affectée au service S_7.

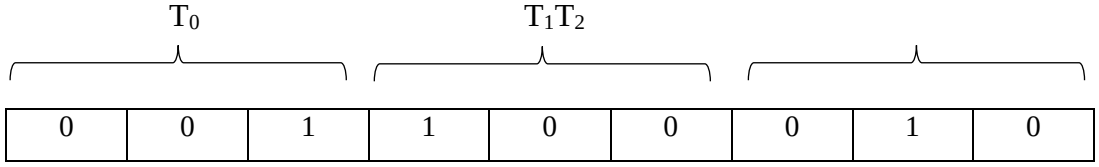

Figure V-1: Exemple de solution

V.4 Proposition d'une méthode MO-GA pour traiter le problème de composition de services web.

Dans cette section nous proposons un GA pour le problème de composition de services Web multi-objectifs.Le but principal de notre méthode notée MO-GA (AG multi-objectifs) est de trouver l'ensemble des solutions optimales.La composition de chaque service a un plan d'exécution.La méthode MO-GA est utilisée pour sélectionner le plan d'exécution optimal.Pour chaque plan d'exécution, un chromosome est créé.En accumulant des chromosomes de plans d'exécution, on obtient une population de solutions candidates pour une composition optimale des services web.

V.4.1 Le processus GA multi-objectifs MO-GA

Comme déjà vu, l'approche GA est utilisée pour sélectionner l'ensemble de solutions Pareto-optimales pour le problème de compositions multi-objectifs de services web. GA est une méta-heuristique inspirée du processus de sélection naturelle qui appartient à la classe plus large des algorithmes évolutionnaires. Le MO-GA démarche comprend les principales étapes suivantes :

• Étape 1. Génération de la population initiale : nous générons aléatoirement un ensemble de chromosomes qui représente la population initiale P de taille N.

• Étape 2. Evaluer toutes les solutions de P en calculant leur fitness : dans MO-GA, chaque solution a plusieurs valeurs de fitness, chaque fitness représente un critère non fonctionnel.

• Étape 3. Calculer une solution non dominée de P et sa liste de solutions non dominées.

• Étape 4. Sélectionnez deux parents à l'aide de la « sélection de tournoi binaire ». Elle consiste à sélectionner au hasard deux solutions de la population P.

• Étape 5. Générer un ensemble de solutions enfants P_child en croisant ces deux parents de l'ensemble P à l'aide d'un croisement à un seul point.

• Étape 6. Répétez 4 et 5 pour toutes les paires de solutions enfants de P_child.

• Étape 7 Exécutez l'opérateur de mutation sur l'ensemble P_child à l'aide d'une fonction aléatoire de probabilité Pr.

• Etape 8. Une procédure de réparation est exécutée si la solution obtenue est irréalisable : cette procédure est exécutée sur chaque nouvelle solution candidate générée alors qu'elle doit respecter toutes les contraintes du modèle de QoS étudié.

• Étape 9. Exécuter une méthode de recherche locale sur P_child comme une simple perturbation pour obtenir une solution meilleure voisine de la solution enfant.

• Étape 10. Calculer le meilleur enfant S à partir de P_child et sa liste de listes de solutions enfants non dominantes.Comparez-le avec S_best et enfant non dominant liste de solutions pour améliorer la solution actuelle et la liste de solutions non dominées.

• Étape 11. Répétez les étapes 2 à 10 jusqu'à ce que les conditions d'arrêt soient satisfaites, le nombre d'itérations max_iter.

Le processus de recherche est répété jusqu'à ce qu'un certain nombre d'itérations appelé max_iter soit atteint, le résultat de son exécution est une liste des solutions Pareto optimales qui sont des solutions non dominées. La méthode MO-GA est esquissée dans l'algorithme V-3.

AlgorithmeV-3 : Algorithme génétique multi-objectifs pour la composition des services web

1 :Entrées : max_itr, pop_size
2 :Sorties : Liste de la solution optimale : S_best, List_optimal_solutions
3 : Générer une population initiale Pinit avec taille = popsize
4 : Solution courante : Scurrent ← Sélectionner la solution non dominée de Pinit
/*Sélection avec la fonction domine*/
5 : List_non_dominate_solutions : liste actuelle des solutions non dominantes de Scurrent
6 : tant que (itération < max_itr) **faire**
7 :**Pour** (chaque paire de P_current) **faire**
8 : (P1,P2) ← sélectionner deux individus parents de Pcurrent
9 : (E1,E2) ← Croisement des deux parents (P1,P2) pour obtenir deux enfants
10 : P_child ← E1,E2
11 :**FinPour**
12 :**Pour** (chaque "ind" individuel de P_child) rand ← Random(0,1)
13 :**si** (rand > Pts) **alors**
14 : ind_mutated = mutate_children(ind) /* Mutation de l'élément "ind" de P_child*/
15 : Remplacer ind par ind_mutated dans P_child
16 : Fin si
17 : FinPour
18 : Pour chaque individu de Pchild **faire**
19 : Scurrent ← Solution ind.
20 : List_Neighb ← calcule le voisinage not_tabo de Scurrent
21 : Best_neighboring ← sélectionne le meilleur voisinage de List_Neighb /* Utilisation de la méthode de recherche locale */.
22 : **si** dominant (Best_neighboring, Scurrent) **alors**
Remplacer la solution Scurrent par Best_neighboring dans Pchild
23 : **fin si**

24 : FinPour

25 : Pcurrent ← Pchild

26 : (S,List_not_domminated_solution_child) ← sélectionner la solution non dominée de Pcurrent/* Utilisant la fonction dominante */

27 : si dominent (S, Sbest) **alors**

28 : Sbest ← S

29 : List_not_dominated_solutions ← List_not_dominated_solutions_child

30 fin si

31 : si dominer (Sbest, S) **alors**

32 : Ajouter la solution Sbest à List_not_domminated_solutions_child

33 : List_not_domminated_solutions ← List_not_domminated_solutions_child

34 : fin si

35 : fin tant que

36 : Ajouter Sbest à List_non_dominate_solutions

37 : Renvoie l'ensemble des solutions optimales trouvées : List_not_dominated_solutions

V.5 Approche memetique multi-objectif MO-MA pour résoudre le problème de composition des services web.

Dans cette section, nous proposons une méthode mémétique multi-objectifs (MO-MA) pour calculer l'ensemble des solutions Pareto-optimales. MO-MA est basé sur l'algorithme génétique (GA) et les algorithmes de recherche locale (LS). MO-MA est le résultat d'une hybridation de deux méthodes précédentes (GA & LS). MA utilise le processus de résolution des algorithmes génétiques avec un opérateur de recherche local mais il n'applique pas l'opérateur de mutation sur les solutions candidates. Le bénéfice tiré de la l'utilisation de cet algorithme est la diversification par la partie génétique et l'intensification par la recherche locale.

L'étape de recherche locale est effectuée sur chaque individu de la population courante. Nous utilisons le même encodage d'une solution que pour les approches LS et GA. La méthode MA multi-objectifs (MO-MA) est basée sur un algorithme mémétique adapté pour résoudre la version multi-objectifs du problème de composition des services web. MA commence par une population initiale, la solution courante est choisie au hasard parmi cet ensemble. Il applique l'opérateur de croisement entre toutes les paires de solutions candidates mais il n'applique pas l'opérateur de mutation sur les solutions candidates. Après l'étape de franchissement, le LS est exécuté sur chaque individu de la population. Les solutions candidates au problème d'optimisation jouant le rôle des individus dans une population et la fonction de fitness

détermine la qualité des solutions. Comme cela a été fait avec LS et GA, nous avons utilisé dans MO-MA plusieurs fonctions objectifs telles que présentées précédemment. MA combine à la fois la recherche locale et le concept évolutionnaire.

Le principe fondamental de la recherche locale est d'exploiter l'interaction entre la diversification et l'intensification là où La diversification pousse à rechercher de nouvelles régions, et l'intensification se concentre davantage sur les régions précédemment jugées bonnes. La phase d'intensification est basée sur l'utilisation d'une condition probabiliste P_child > 0, elle consiste à améliorer la solution candidate en choisissant la meilleur de ses solutions voisines selon les valeurs des fonctions objectifs. L'approche MO-MA est utilisée pour trouver une solution optimale au problème de compositions de services Web. Ce problème est modélisé sous la forme d'un problème d'optimisation multi-objectifs, donc la meilleure solution trouvée est la solution Pareto-optimale, qui fait partie de l'ensemble de solutions optimales non-dominées.

Il y a cinq étapes principales dans notre méthode MO-MA qui peuvent être données comme suit :

• Étape 1. La génération de la population initiale : dans cette étape, nous créons un ensemble de solutions candidates (ensemble de chromosomes) généré aléatoirement qui représente la population initiale P.

• Étape 2. Evaluer toutes les solutions de P en calculant leur fitness : en MA multi-objectifs, chaque solution a plusieurs valeurs de fitness, chaque fitness représente un critère non fonctionnel.

• Étape 3. Calculer une meilleure solution S_best et son ensemble de solutions non dominées à partir de P.

• Étape 4. Sélectionnez aléatoirement deux solutions de la population P, elles seront considérées comme des parents pour appliquer l'opérateur de croisement.

• Étape 5. Générer l'ensemble de solutions enfants P_child en croisant ces deux parents de l'ensemble P en appliquant des opérateurs de croisement.

• Étape 6. Répétez 4 et 5 pour toutes les paires de solutions candidates de P.

• Etape 7. Une procédure de correction est exécutée si la solution obtenue est irréalisable : chaque solution candidate doit respecter toutes les contraintes de la Modèle de qualité de service.

• Étape 8. Exécuter une méthode de recherche locale sur P_child comme une simple perturbation pour obtenir une solution meilleure voisine de la solution enfant.

• Étape 9. Calculez la meilleure solution enfant « S » à partir de l'ensemble P_child et de sa liste de solutions enfants non dominantes, après l'avoir comparée à S_best et liste de solutions filles non dominantes pour améliorer la solution actuelle et sa liste de solutions non dominées.
• Étape 10. Répétez les étapes 2 à 10 jusqu'à ce que les conditions d'arrêt soient satisfaites, le nombre d'itérations max_iter

La méthode proposée MO-MA dans l'algorithme V-4 :

AlgorithmeV-4 : Algorithme mémétique multi-objectif pour la composition de services web

1 : entrées : max_itr, pop_size
2 : sorties : Liste de solution optimale : S_best, List_optimal_solutions.
3 : Générer une population initiale Pinit avec size = pop_size
4 : Solution courante : Scurrent ← Sélectionner la solution non dominée à partir de Pinit
5 : Scurrent/* Sélection à l'aide de la fonction dominate*/.
6 : List_non_dominate_solutions : liste courante des solutions non dominées de Scurrent
7 : tant que (itération < max_itr) **faire**
8 :**Pour** (chaque paire de Pcurrent) **faire**
9 : (P1,P2) ←sélectionner deux individus parents de Pcurrent
10 : (E1,E2) ←Croisement des deux parents (P1,P2) pour obtenir deux enfants
11 : P_child ← E1,E2
12 : **FinPour**
13 :**Pour** chaque individu ind de Pchild**faire**
14 :Scurrent ← ind
15 : List_Neighb ←calcule le voisinage not_tabo de Scurrent
16 :Best_neighboring ←sélectionner le meilleur voisin de List_Neighb
/* Utilisation de la méthode de recherche locale (LS) */
17 : **Si** dominate (Best_neighboring, Scurrent) **alors**
18: Remplacer la solution Scurrent par Best_neighboring dans Pchild
19 :**fin si**
20 : FinPour
21 : Pcurrent ← Penfant
22 : (S,Liste_non_dominated_solution_child)←sélectionner la solution non dominée de Pcurrent /* Utilisation de la fonction dominante */
23 : si dominate (S, Sbest) **alors**
24 : Sbest ← S
25 : List_not_dominated_solutions ← List_not_dominated_solutions_child
26 : fin si
27 : si dominate (Sbest, S) **alors**

28 :Ajouter la solution Sbest à List_not_domminated_solutions_child

29 : List_non_dominated_solutions ← List_non_dominated_solutions_child

30 : fin si

31 : fin tant que

32 : Ajouter Sbest à List_not_dominated_solutions

33 : Renvoie l'ensemble des solutions optimales trouvées : List_not_domminated_solutions

Nous notons que notre algorithme mémétique se compose de quatre composants principaux, qui sont : la génération de la population initiale, la méthode de sélection, le calcul de la fitness des individus et l'étape d'amélioration de la recherche locale.La fonction de sélection calcule la meilleure solution de la population actuelle de solutions.La fonction de recherche locale qui travaille sur l'ensemble des solutions voisines d'une solution courante dont l'objectif est de choisir la meilleure solution de voisinage en utilisant la relation de domination.Pour ces fonctions précédentes, nous utilisons la relation de domination pour comparer les vecteurs de valeurs où chaque vecteur représente une solution.

La plus grande difficulté est de trouver l'ensemble d'éléments non-dominés à partir d'une population de solutions candidates de grande taille.Les vecteurs résultants sont l'ensemble de solutions Pareto-optimal en utilisant les fonctions objectifs.Ainsi, parmi ces derniers, notre algorithme proposé permet de sélectionner un élément.Il est clair que le MA a une complexité linéaire qui est raisonnable dans le temps.

V.6 Expérimentation & Résultats numériques

V.6.1 Expérimentation de l'approche MO-LS

Dans cette section, nous allons présenter les résultats numériques trouvés en utilisant l'approche MO-LS pour résoudre le problème de la composition de services web.

V.6.1.1 Les résultats numériques utilisant l'ensemble de données cité dans *[84][85]*

Dans cette sous-section, nous présentons les résultats obtenus à l'aide de l'ensemble de données cité dans [84]et[85].

D'après la première expérience où dans la Figure V-2 qu'il est clair de voir qu'un compromis de 12 solutions est trouvé à la $5^{ème}$ itération. Cependant la Figure V-3 nous montre l'impact de nombre d'itérations exécutés sur l'évolution du nombre de solutions Pareto-optimales distinctes pour les différentes instances étudiées. Par ailleurs, la Figure V-4nous montre le temps écoulé calculé pour chaque instance. Pour l'instance définie avec N = 15 M = 3, nous

pouvons remarquer que 82 solutions sont trouvées à la 15ème itération en 795 ms.D'autre part, 295 solutions sont trouvées à la 100ème itération en 2989 ms.

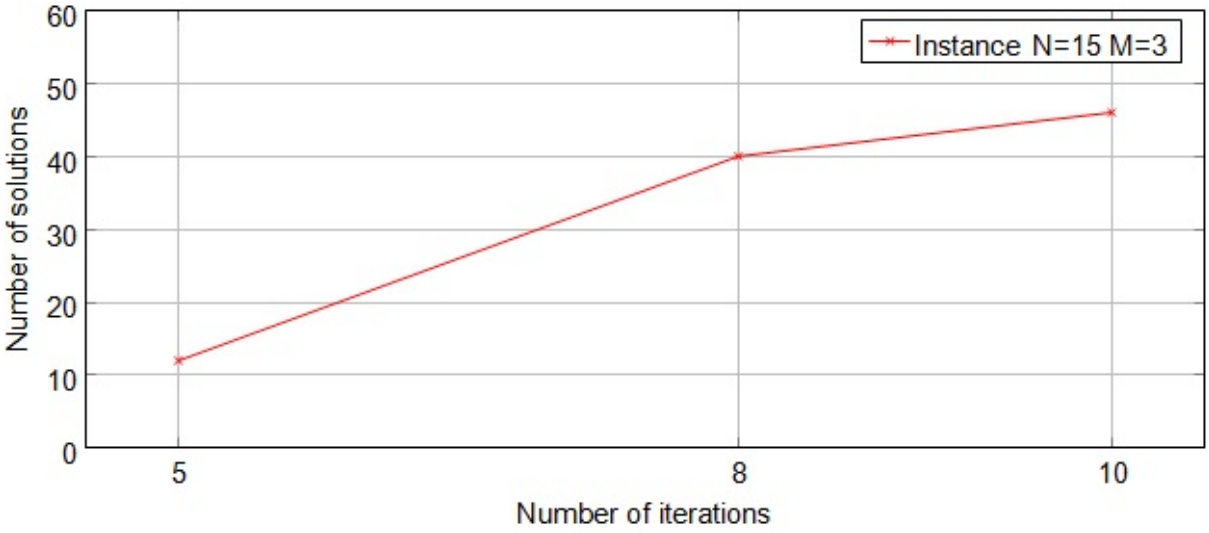

Figure V-2: l'ensemble de solutions non-dominées

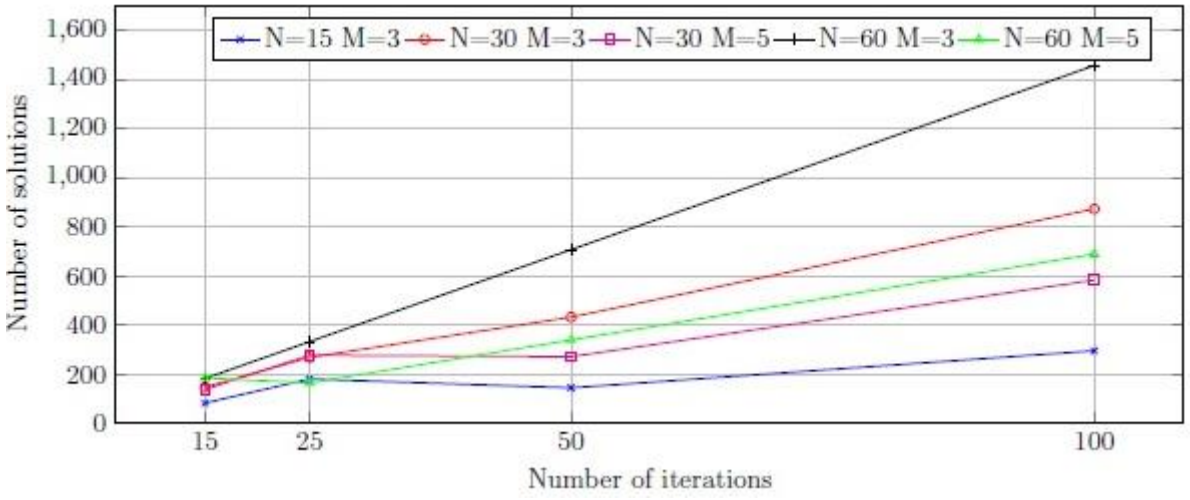

Figure V-3: l'ensemble des solutions pour cinq instances

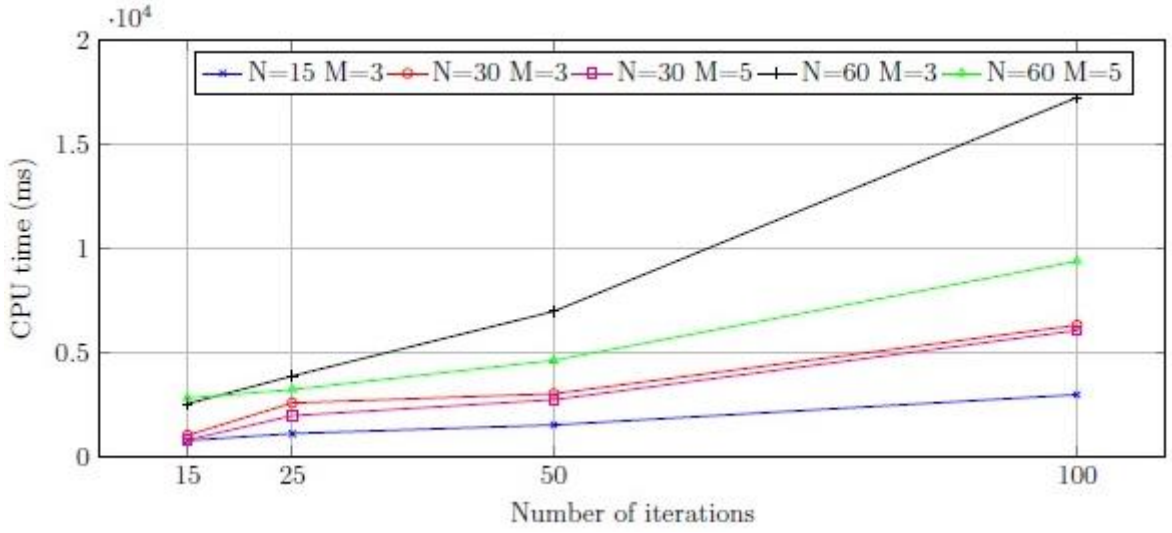

Figure V-4: Le temps écoulé par itération pour les cinq instances

Pour l'instance de N = 30 et M = 3 (30 services et 3 tâches), après 15 itérations.Nous avons trouvé 147 solutions distinctes en 1045ms.Sachant que les solutions de compromis trouvées respectent toutes les contraintes considérées du modèle QoS étudié.La méthode proposée donne à la 100ème itération 873 solutions distinctes non dominées en 6320s. Pour l'instance de 60 services et trois tâches (N = 30 et M = 3), 182 solutions distinctes sont trouvées à la 15ème itération. L'utilisation de 60 services et 5 tâches (N = 30 et M = 5), cette expérimentation de notre algorithme a montré qu'on peut obtenir jusqu'à 184 solutions distinctes dès la 15ème itération en 2823ms environ, ce résultat évolue à chaque fois qu'on augmente le nombre d'itérations.

Nous avons remarqué que le nombre de solutions optimales trouvées pour les instances (N = 60 M = 3) et (N = 60 M = 5) est supérieur au nombre de solutions trouvées pour les autres instances.Ces résultats montrent que LS réussit à trouver des solutions de compromis.

V.6.1.2 Les résultats numériques utilisant le benchmark généré aléatoirement

Nous présentons dans cette sous-section le résultat obtenu lors de l'application de la méthode MO-LS sur les jeux de données générés aléatoirement. Comme le montre la Figure IV-5, pour l'instance définie par N = 15 et M = 3, un compromis de 45 solutions est trouvé à la 5èmeitération. Le compromis des solutions respecte toutes les contraintes du modèle QoS sans aucune violation. Comme nous pouvons le voir sur la figure IV-6, le nombre de solutions Pareto-optimales trouvées est très important pour toutes les instances considérées. Pour l'instance considérée ((N, M) = (60, 5)), le nombre de solutions Pareto-optimales atteint 2212 trouvées à la 100ème itération.

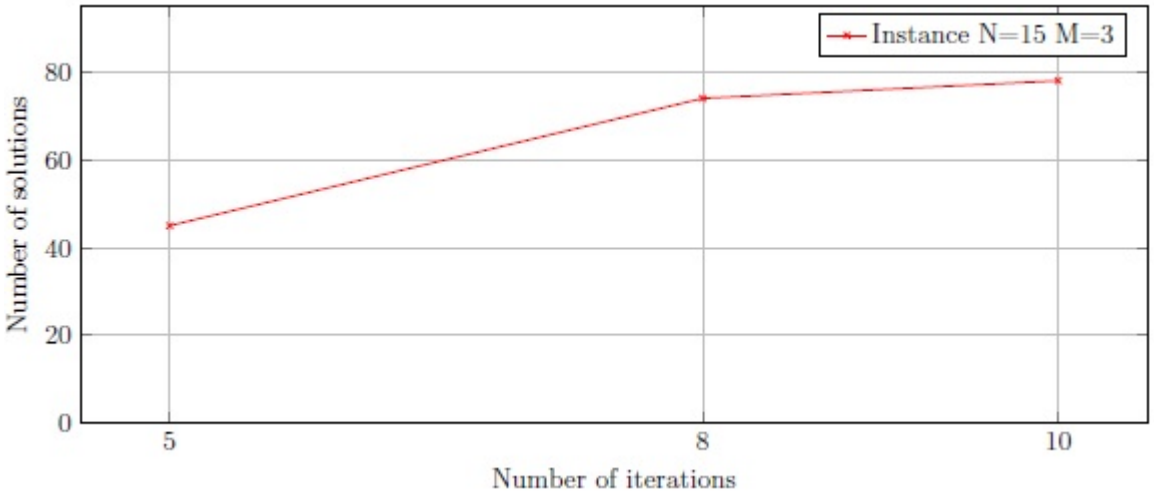

Figure V-5: Ensemble de solutions non dominées par itérations

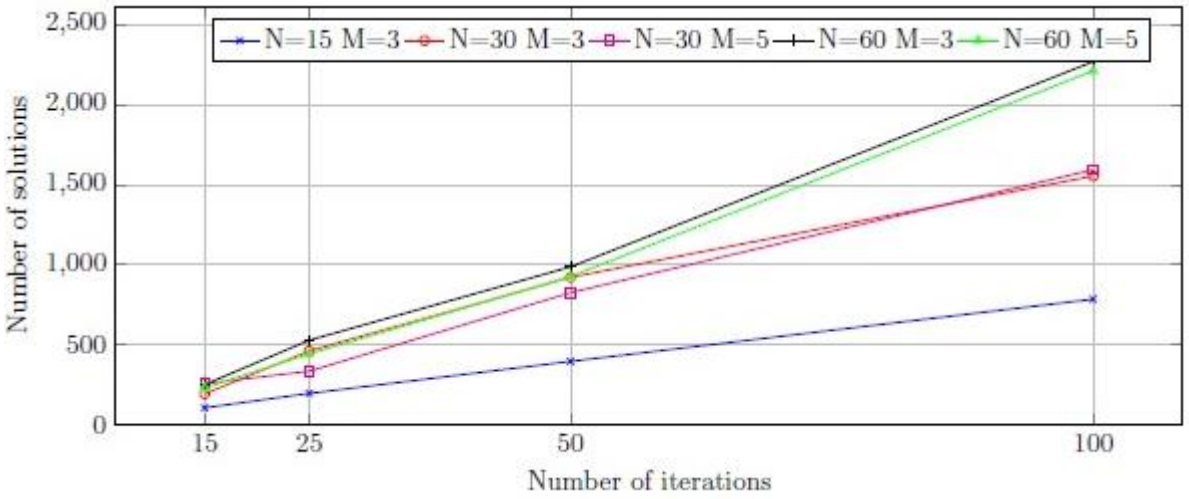

Figure V-6: Ensembles des solutions non-dominées pour cinq instances

Nous avons la même remarque à propos du temps écoulé pour l'exécution des itérations pour instances considérées comme le montre la Figure IV-7. Par exemple pour l'instance ((N, M) = (60, 5)), à la 100ème itération, le temps écoulé est de 20,220 ms.

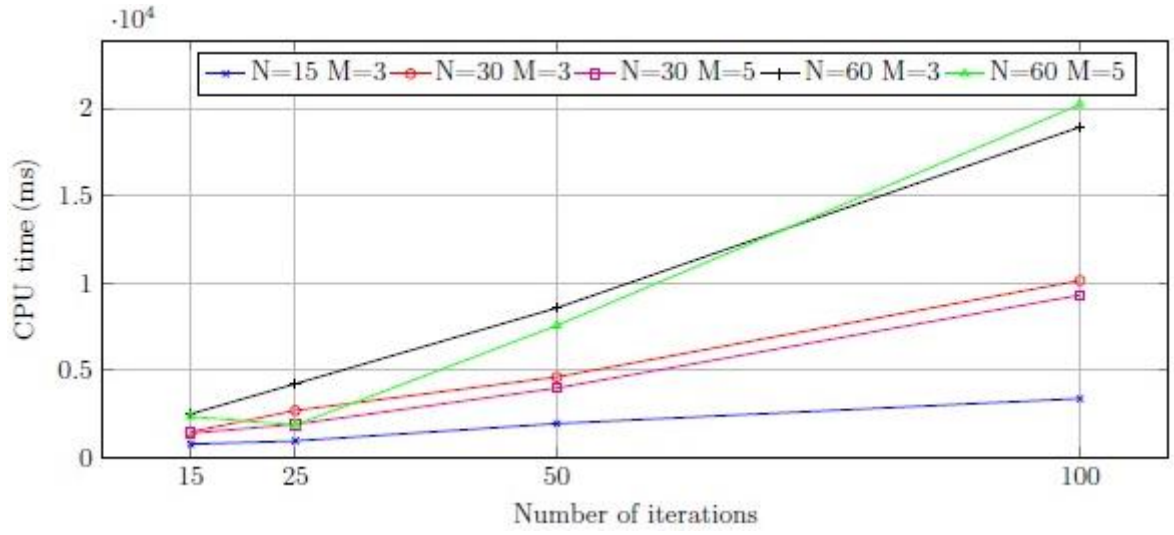

Figure V-7: Temps écoulé pour les cinq instances

Sur ce, nous pouvons dire que la méthode proposée est capable de trouver un bon ensemble de solutions Pareto-optimales. Par conséquent, l'utilisateur peut choisir la solution optimale proposée qui peut satisfaire ses besoins. Mais il n'est pas facile de choisir la solution souhaitée parmi ces nombreuses solutions. Sur ce, nous avons appliqué une autre méthode basée sur l'algorithme GA dont les résultats numérique sont présentés dans la section suivante.

V.6.2 Expérimentation de l'approche MO-GA :

Un ensemble de solutions est trouvé et chacune représente une composition optimale.Ce sont des solutions non dominées appelées aussi solutions optimales Pareto. A partir de cet ensemble de solutions, l'utilisateur peut choisir la solution qui lui convient.

V.6.2.1 Les résultats numériques utilisant l'ensemble de données cité dans *[84][85]*

Dans cette section, nous présenterons les résultats numériques trouvés lors de l'application de GA sur l'ensemble de données cité dans [84], [85]. Dans la Figure IV-8, nous montrons l'évolution de notre modèle en fonction du nombre de solutions optimales trouvées impactée par le nombre d'itérations exécutées. Nous pouvons garantir que 45 solutions environ se sont calculées à la 5ème itération. Pour obtenir plus de détails sur l'évolution du nombre de solutions optimales par itération, nous avons testé notre méthode sur cinq instances. Dans la figure V-9, nous montrons les cinq diagrammes pour les cinq instances.

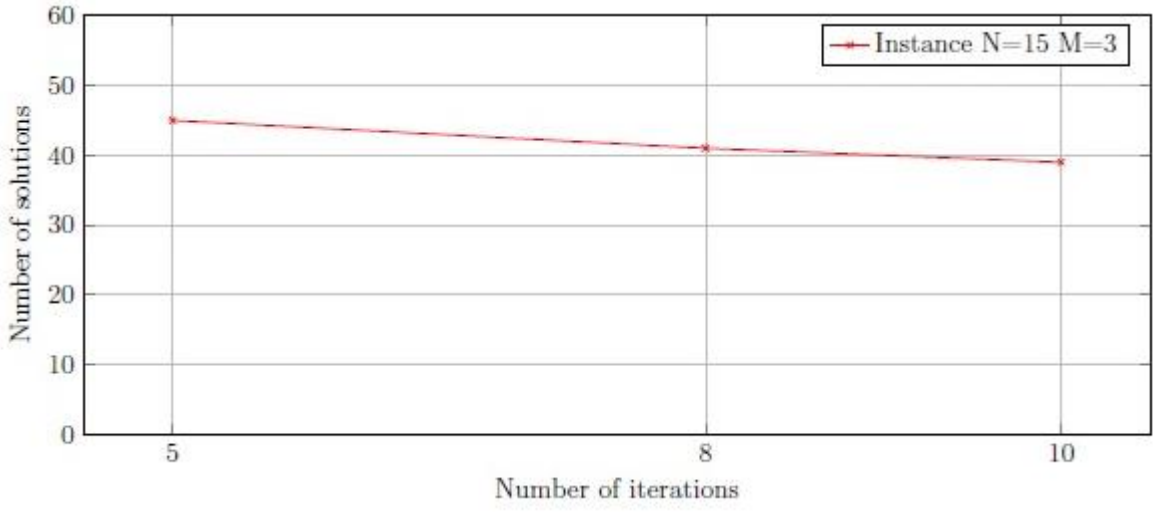

Figure V-8: Ensemble de solutions non-dominées par itération

Concernant l'instance de N = 15 services et M = 3 tâches, nous pouvons voir la Figure V-9 et IV-10 que 46 solutions sont trouvées à la 15ème itération en 8813ms. Nous voyons aussi que 55 solutions sont trouvées à la 100èmeitération en 11960ms.

Pour l'instance de 30 services et 3 tâches (N = 30 et M = 3), nous voyons sur les figures V-9 et IV-10 que 67 solutions distinctes sont trouvées pour la 15ème itération en 12054ms. Le compromis des solutions ne viole aucune contrainte. Pour cette instance, l'algorithme donne 57 solutions distinctes non dominées à la 100ème itération en 12306ms.

Concernant l'instance de 60 services et 3 tâches, comme le montre la Figure IV-9, les solutions optimales ont été trouvées sans violation de contraintes à la 15ème itération où 64 solutions sont localisées en 27440ms. A la 100ème itération, il y a 66 solutions optimales sont trouvées en 16036ms.

En utilisant 60 services et 5 tâches, chaque tâche a 12 services candidats et chaque service ne peut exécuter qu'une seule tâche. Cette expérience de notre algorithme a montré que nous pouvons trouver jusqu'à 61 solutions distinctes à la 15ème itération et 69 solutions environ à la 100ème itération en 25925ms. En fait, via l'approche MO-GA, nous trouvons des solutions optimales non dominées dans un temps raisonnable et en un nombre d'itérations faisables, une fois une solution sera sélectionné par l'utilisateur, les autres solutions sont maintenues.

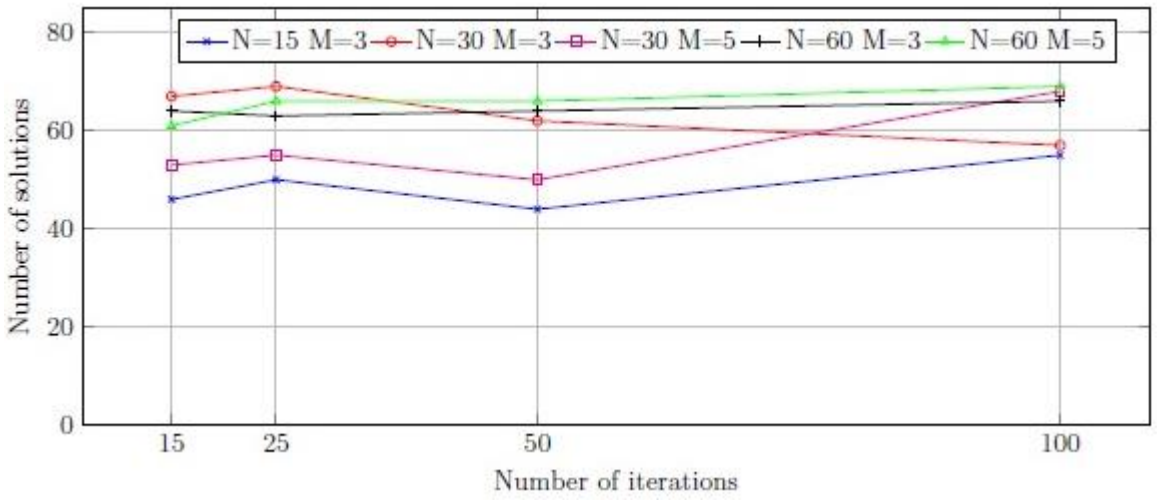

Figure V-9: Ensemble de solutions non-dominées par itération pour cinq itérations.

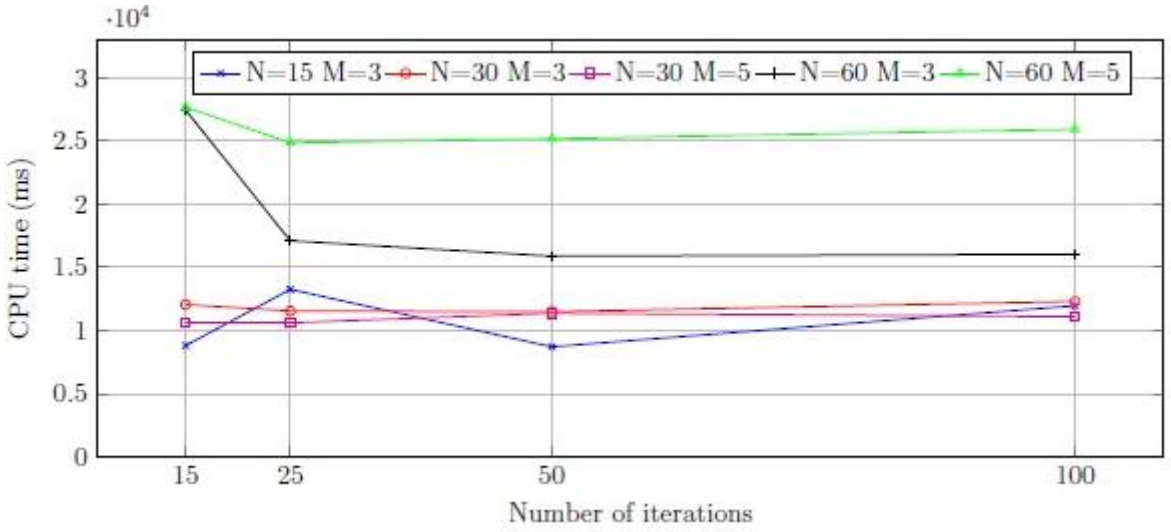

Figure V-10: temps écoulé pour cinq itérations

V.6.2.2 Les résultats numériques utilisant le benchmark généré aléatoirement

Un ensemble de solutions non-dominées est trouvé et chacune représente une composition optimale de services web. À partir de ces solutions optimales trouvées, l'utilisateur peut sélectionner l'une d'entre elles. Nous présentons dans cette section le résultat lors de l'application de MO-GA sur le jeu de données aléatoire. Il est généré par une méthode aléatoire(Random).

Dans la Figure IV-11, nous montrons l'évolution du nombre de solutions optimales pour l'instance (N = 15 et M = 3). Des compromis de 36 solutions sont trouvées à la 5ème itération. D'après la Figure IV-12, nous pouvons dire que le nombre de solutions optimales est compris entre 9 et 27, ce nombre est restreint. D'autre part, nous pouvons noter que le nombre de solutions optimales trouvées est approximativement stable pour toutes les instances étudiées.

La Figure IV-13 montre que le temps écoulé pour chaque instance d'exécution augmente chaque fois que le nombre d'itérations augmente. Par exemple pour l'instance de N = 15 M = 3, le temps écoulé était de 10802ms à la 15ème itération et de 9041ms à la 100ème itération. L'approche MO-GA lorsqu'elle est appliquée à notre problème montre les meilleurs résultats pour trouver des solutions de compromis par rapport les résultats trouvés par l'application de l'algorithme MO-LS.

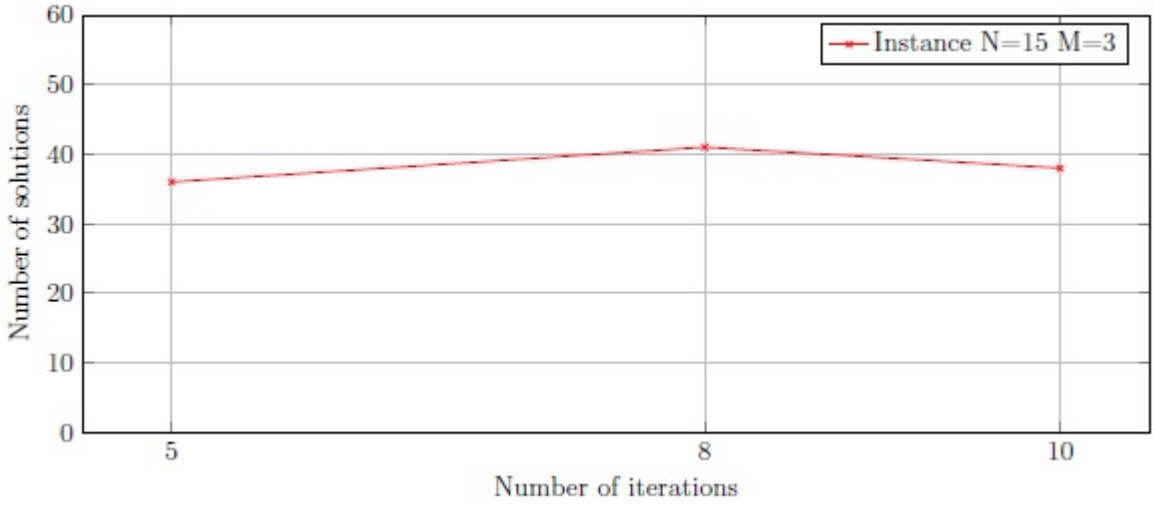

Figure V-11: Ensemble de solutions non-dominées par itération

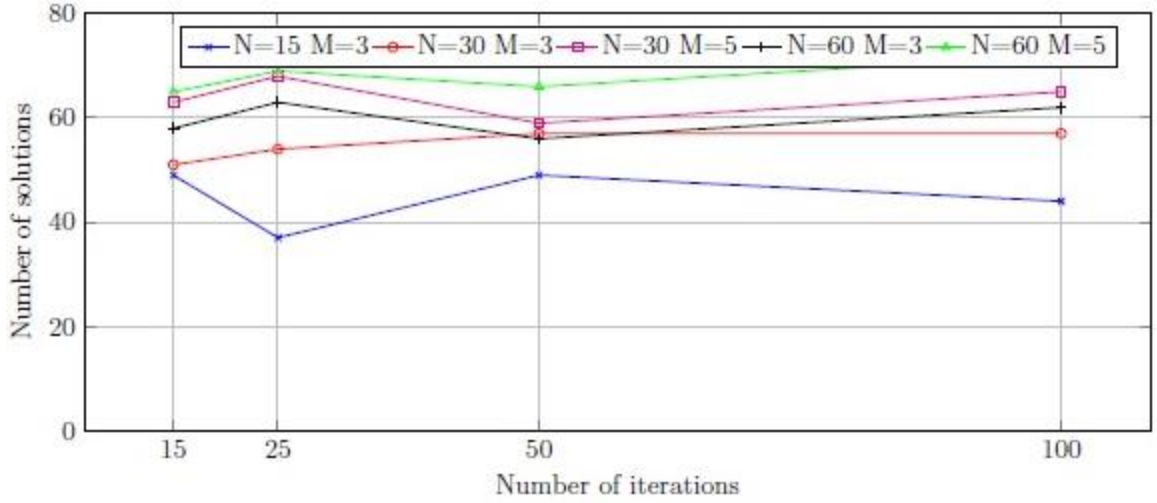

Figure V-12:Ensemble de solutions non-dominées par itération pour cinq instances

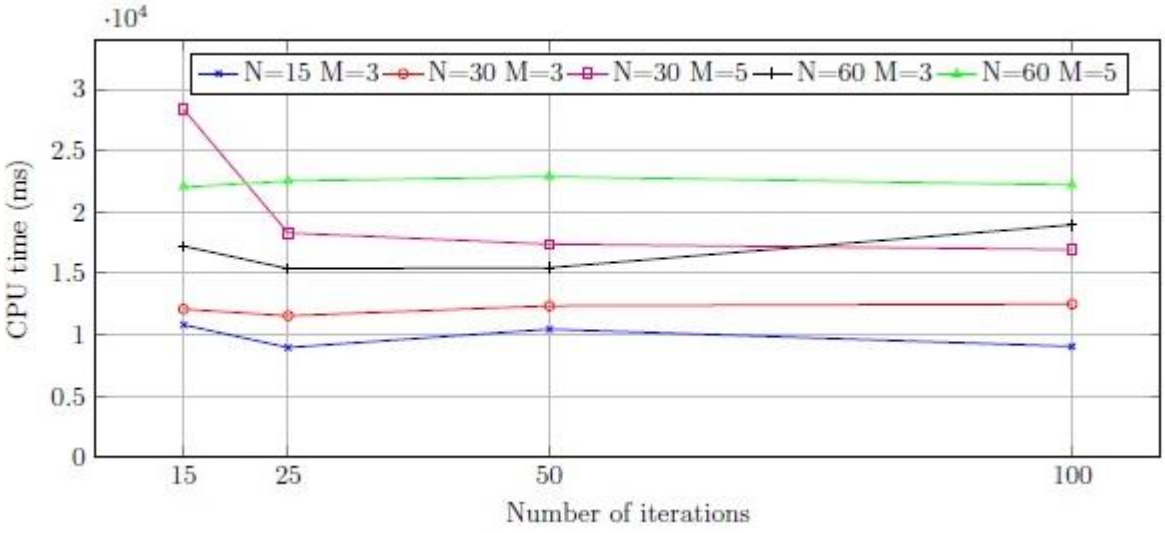

Figure V-13: Temps écoulé par itération pour les cinq itérations

V.6.3 Expérimentation de l'approche MO-MA :

Pour évaluer notre approche MO-MA, nous utiliserons les deux jeux de données déjà utilisés pour les approches MO-LS et MO-GA.On considère les mêmes hypothèses concernant le nombre de services candidats pour chaque tâche.

V.6.3.1 Les résultats numériques utilisant l'ensemble de données cité dans *[84], [85]*

Dans cette sous-section, nous montrons les résultats de l'approche MO-MA obtenus sur l'ensemble de données du benchmark cité dans [84]et [85]. Sur la Figure IV-14, nous remarquons que des compromis de 13 solutions sont trouvées à la 5ème itération. Dans la Figure V-15, nous montrons les résultats des cinq instances de l'évolution du modèle en fonction du nombre de solutions optimales Pareto distinctes trouvées et du nombre d'itérations exécutées. Pour l'instance N = 15 M = 3, nous pouvons remarquer que 12 solutions sont trouvées à la 15ème itération en 1788ms.D'autre part, 14 solutions sont trouvées à la 100ème itération en 8088ms comme le montre la Figure V-16.

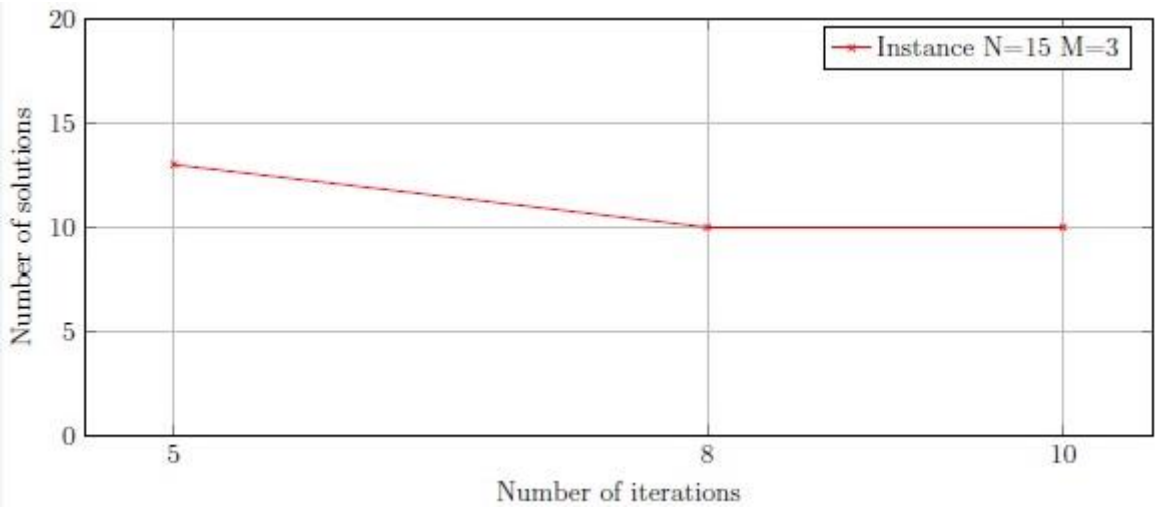

Figure V-14: Résultat de MO-MA : ensemble de solutions non-dominées par itérations

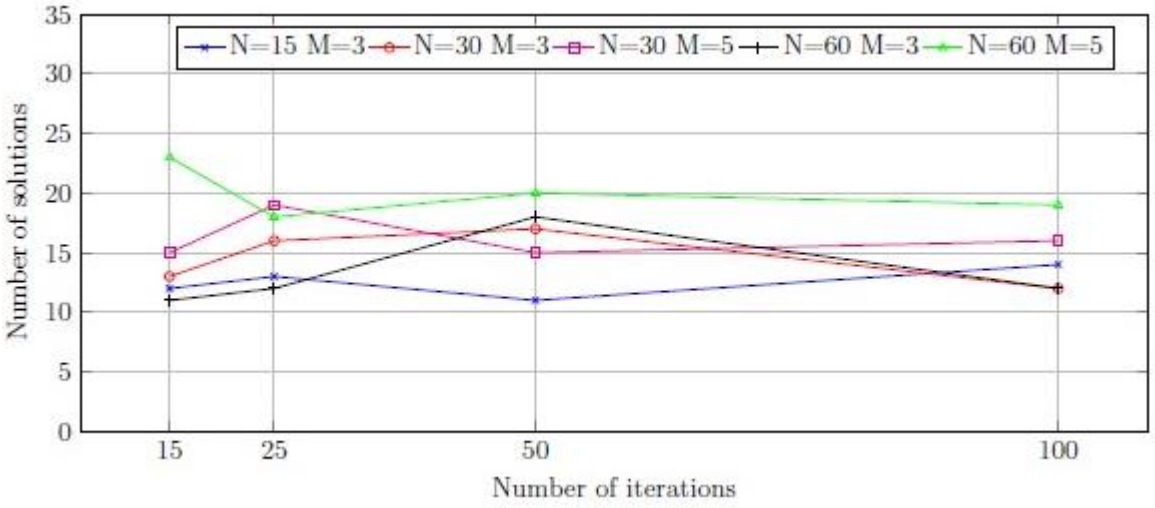

Figure V-15: Résultat de MO-MA : ensemble de solutions non-dominées par itération pour les cinq instances

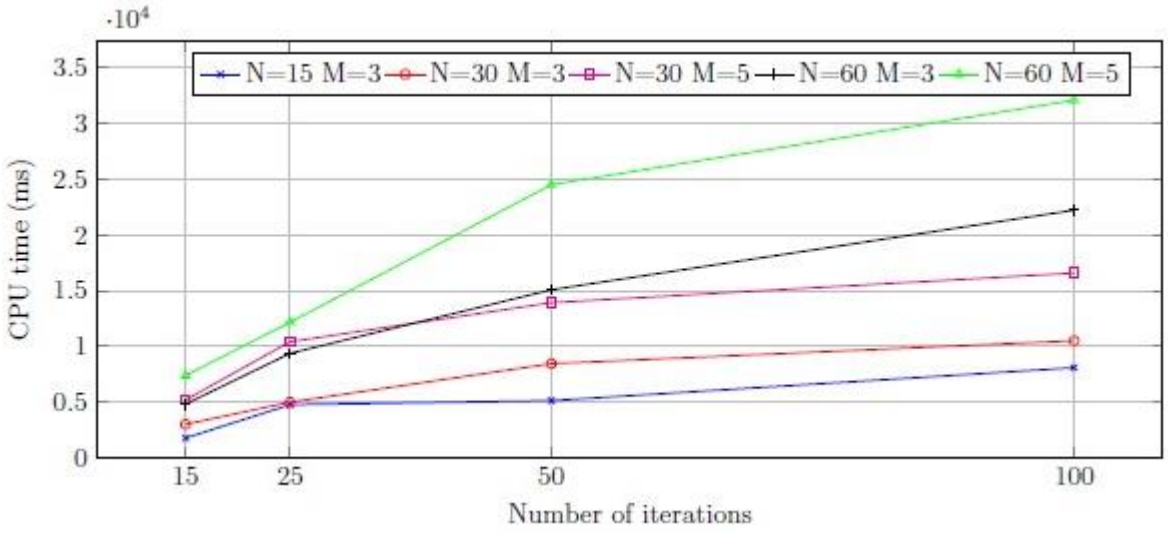

Figure V-16: Résultat de la MO-MA : Temps écoulé par itération pour les cinq instances

Pour l'instance de N = 30 et M = 3 (30 services et 3 tâches), à la 15ème itération, nous avons trouvé 13 solutions distinctes en 3017 ms. Pareillement, le compromis des solutions respecte toutes les contraintes du modèle QoS. La méthode proposée (MO-MA) donne pour 100 itérations 12 solutions distinctes non-dominées dans environ 10s.

Pour l'instance de 30 services et cinq tâches (N = 30 et M = 5), 15 solutions distinctes sont trouvées à la 15ème itération en 5197ms où 16 solutions distinctes sont trouvées à l'itération 100 en 16570ms, ces résultats sont approximativement proches des résultats trouvés pour l'instance de N = 60 et M = 3. En utilisant 60 services et 5 tâches (N = 60 et M = 5), le nombre de solutions distinctes a augmenté, 23 solutions distinctes sont trouvées dans le 15ème itération en 7389ms. Globalement, nous pouvons dire que le nombre de solutions distinctes évolue à chaque fois qu'on augmente le nombre d'itérations, mais généralement, le nombre est limité dans la plage de 11 à 23, nous pouvons conclure que MO-MA réussit à trouver des solutions de compromis, et les résultats obtenus montrent clairement ses performances.

V.6.3.2 Les résultats numériques utilisant le benchmark généré aléatoirement

La première expérience : nous appliquons dans cette section l'approche MO-MA proposée sur le jeu de données aléatoire. Dans la Figure V-17, nous montrons l'évolution du nombre de solutions optimales par numéro d'itération pour l'instance « N = 15, M = 3 ».Un compromis des solutions est trouvé à la 5ème itération. Dans la figure IV-18, nous montrons l'évolution du nombre de solutions optimales Pareto distinctes trouvées pour cinq instances.

Pour l'instance de 15 services et 3 tâches, nous pouvons voir sur les figures V-18 et V-19 que 7 solutions environ sont trouvées à la 15ème itération de 10161ms.Nous voyons aussi que 8 solutions sont trouvées à la 100ème itération en 7350 ms.

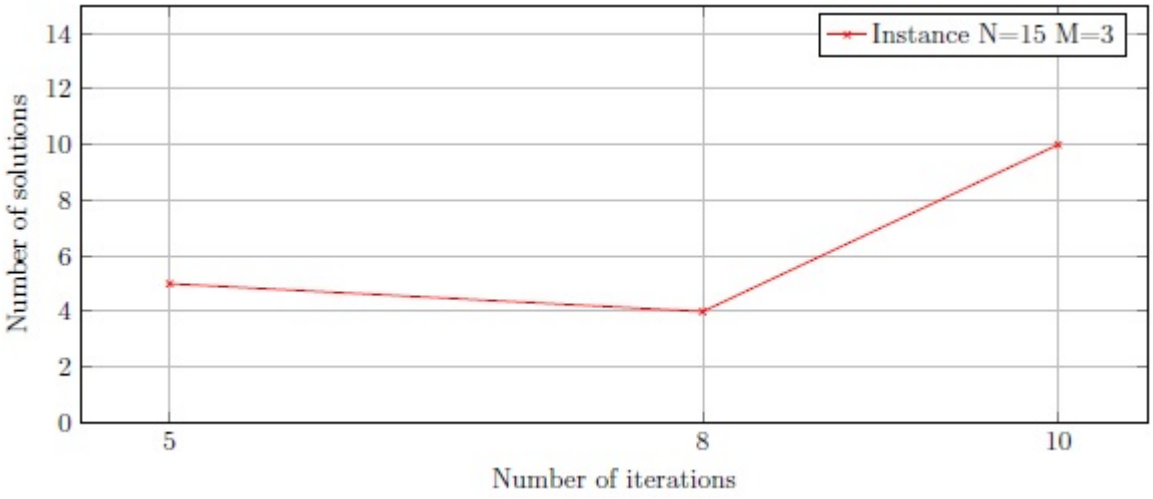

Figure V-17: Ensemble de solutions non-dominées par itération

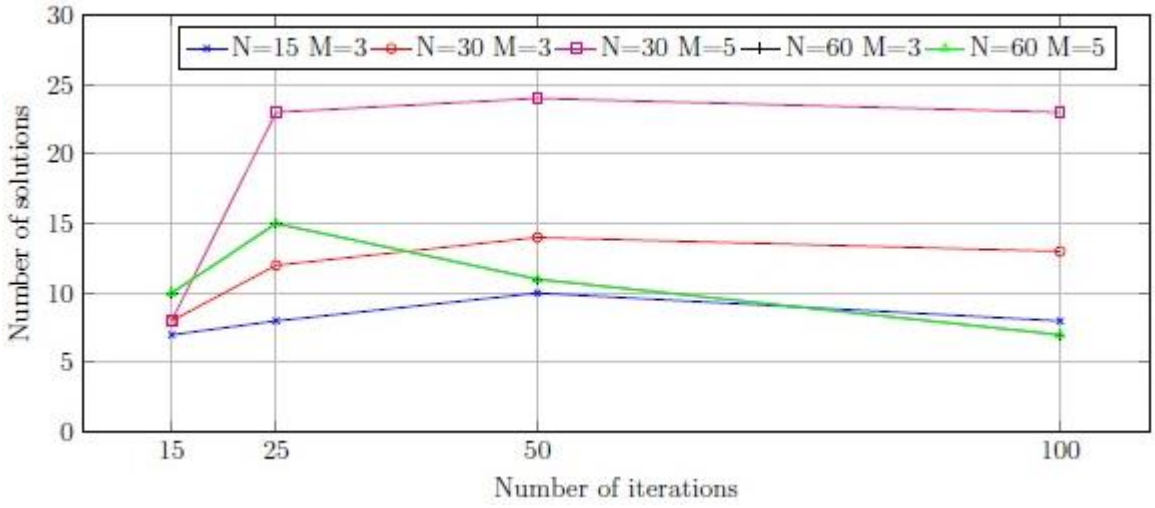

Figure V-18: Ensemble de solutions non-dominées par itération

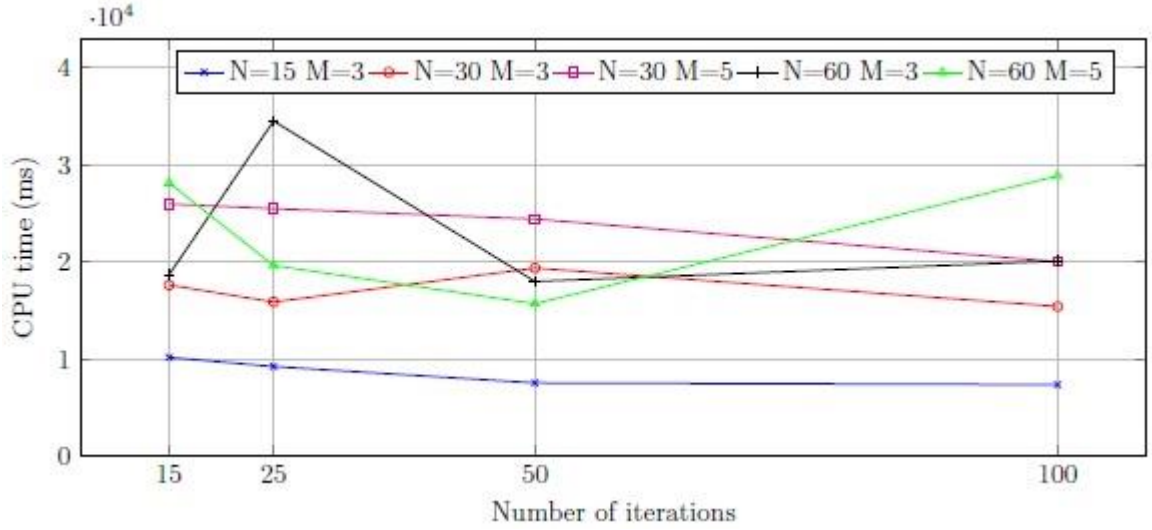

Figure V-19: Temps écoulé par itération pour les cinq instances

En fait, lorsque on trouve des solutions non-dominées, une fois une solution est atteinte est exposée à l'utilisateur, les autres sont maintenues.

Pour l'instance où nous avons 30 services et 3 tâches (N = 30, M = 3), nous pouvons voir sur les figures V-18 et V-19 que 8 solutions distinctes sont trouvées pour 15 itérations en 17627 ms. Pour la 100ème, 13 solutions distinctes non dominées en 15423s environ. Nous avons remarqué que pour cette instance, le nombre de solutions optimales trouvées est approximativement stable, il est compris entre 8 et 14 solutions trouvées. La même observation est notée pour les trois autres instances de (N = 30, M = 3), (N = 60, M = 3) et (N = 60, M = 5). Par contre, nous avons noté sur les figures V-18 et V-19 concernant l'instance (N = 60, M = 3) que le nombre de solutions optimales trouvées en 10 à 15 itérations sont trouvées en 18685 ms et plus de 7 solutions pour les 100 itérations sont obtenues en 20097 ms.

La méthode proposée MO-MA est capable de trouver un ensemble de solutions optimales Pareto avec un nombre limité de solutions. Parmi ces solutions, l'utilisateur peut choisir une solution optimale qui peut satisfaire sa requête.

V.6.4 Une étude comparative entre les trois méthodes proposées

Afin d'effectuer une comparaison équitable, nous avons exécuté les trois méthodes dans les mêmes conditions en donnant le même délai pour chaque méthode.Le délai est fixé à 15 min maximum pour chaque méthode.Le tableau IV.1 donne les résultats numériques pour les trois méthodes MO-LS, MO-GA et MO-MA. Pour chaque méthode, nous donnons le nombre de solutions obtenues (NB solutions), la valeur des quatre fonctions objectifs considérées (coût, temps, disponibilité et notoriété).

Pour chaque fonction, nous donnons la valeur maximale (max), la valeur minimale (min) et la valeur moyenne (moyenne) trouvées par chaque méthode sur chaque instance. Selon les résultats rapportés dans le tableau IV.1, nous pouvons voir que le nombre de solutions optimales obtenues par la méthode MO-MA est beaucoup meilleurs que ceux trouvés par les deux autres méthodes MO-LS et MO-GA.

	MO-LS											
	NB	Cost			Time			Availability			Reputation		
Instances	solutions	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean
Instance 1 ($N = 15$ $M = 3$)	19,855	240	150	196.36	300	283	289.18	446.44	258.6	367.74	300	283	289.18
Instance 2 ($N = 30$ $M = 3$)	20,343	220	150	177.67	425.4	246.4	335.14	300	283	298.42	280.9	253.2	268.07
Instance 3 ($N = 30$ $M = 5$)	18,611	400	285	337.60	645.64	466.64	546.98	500	483	498.21	459	431.2	442.34
Instance 4 ($N = 60$ $M = 3$)	21,238	260	170	201.73	643.2	300	423.02	300	281	296.81	286.1	248.9	268.62
Instance 5 ($N = 60$ $M = 5$)	11,364	140	90	118.90	320	154.09	206.28	200	183	197.31	191.4	170.4	181.23
	MO-GA											
	NB	Cost			Time			Availability			Reputation		
Instances	solutions	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean
Instance 1 ($N = 15$ $M = 3$)	348	295	95	192.98	649.62	249.6	414.48	600	283	370.08	524.80	239.70	328.67
Instance 2 ($N = 30$ $M = 3$)	984	280	90	167.42	601.42	200.4	353.52	483	266	305.06	445.2	234.5	267.05
Instance 3 ($N = 30$ $M = 5$)	895	290	90	170.99	654.5	209	362.38	500	266	313.03	438.7	234.5	274
Instance 4 ($N = 60$ $M = 3$)	1311	260	95	170.98	1901.5	233.5	574.63	400	245	295.58	330.1	204.70	246.33
Instance 5 ($N = 60$ $M = 5$)	1033	425	175	286.46	2204.44	383.11	876.51	586	364	483.78	528.19	303.5	408.99
	MO-MA											
	NB	Cost			Time			Availability			Reputation		
Instances	solutions	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean
Instance 1 ($N = 15$ $M = 3$)	3	210	160	185	305.15	303.4	303.98	300	300	300	264.5	258.3	260.36
Instance 2 ($N = 30$ $M = 3$)	8	240	180	210	306.03	246.4	276.21	300	283	291.5	280.9	259.2	270.05
Instance 3 ($N = 30$ $M = 5$)	8	295	160	227.5	561.09	445.2	503.14	500	483	491.5	440	402.5	421.25
Instance 4 ($N = 60$ $M = 3$)	7	220	125	170.55	432.45	291.45	365.67	300	283	292.44	273.6	262.3	267.52
Instance 5 ($N = 60$ $M = 5$)	3	290	245	264	614.17	490	550.1	500	469	484.2	433.89	424.5	429.95

Tableau V-1: Une étude comparative entre les trois approches proposées

Comme le montre tableau V-1, le nombre de solutions optimales est réduit par rapport à LS et GA.De plus, MO-MA réussit à trouver de bons résultats en termes de coût, de temps, de disponibilité et de points de vue réputation.Nous pouvons conclure que les résultats de l'approche MO-MA sont très compétitifs.

V.6.5 Comparaison supplémentaire

Premièrement, nous essayons de faire une comparaison entre les résultats de chaque méthode proposée (MO-LS, MO-GA et MO-MA) et sa méthode de base (LS, GA, et MA), respectivement. Dans ce travail, nous constatons qu'une seule solution optimale est trouvée par une méthode de base. En revanche, les méthodes multi-objectifs nous donnent plusieurs solutions optimales qui sont souhaitées par l'utilisateur.

Deuxièmement, nous comparons les résultats trouvés en appliquant l'approche MO-MA et ceux trouvés en appliquant MO-LS, MO-GA à résoudre le problème d'optimisation multi-objectifs de la composition des services web.

Troisièmement, nous prenons comme approche comparative les travaux présentés dans[103] où l'algorithme NSGA (Non-dominated Sorting Genetic Algorithm) a été présenté, car cet article est parmi les plus récents et les plus pertinents travaux dans le domaine de l'optimisation multi-objectifs de la composition de services web. Selon les résultats numériques obtenus lors de l'application de l'approche MO-LS, on peut dire que le nombre de solutions optimales est grand.

Par exemple, dans le résultat trouvé en utilisant l'ensemble de données du benchmark cité dans [84], [85], pour l'instance de N = 60 M = 5, nous voyons que 689 solutions sont trouvées dans l'itération 100 et 2269 solutions sont trouvées en utilisant le jeu de données du benchmark aléatoire.

Pour les résultats de MO-GA, nous voyons que le nombre de solutions optimales trouvées est globalement compris entre 40 et 75 en utilisant le jeu de données de [84], [85] et 35 et 70 en utilisant le jeu de données benchmark aléatoire.

Les résultats MO-MA montrent que le nombre de solutions optimales trouvées est globalement compris entre 10 et 25 en utilisant le benchmark cité dans le travail d'Al-Masri et Mahmoud [84]et[85] et 5 et 25 pour le jeu de données aléatoires. Globalement, nous pouvons dire que l'approche MO-MA donne le plus petit nombre de solutions. D'après la figure IV-20, nous pouvons dire que le nombre de solutions optimales trouvées par MO-MA est largement meilleur que le nombre trouvé lorsque en appliquant MO-GA et NSGA. En comparant avec les travaux de [103] où aucune solution n'a été trouvée, le nombre de solutions Pareto-optimales trouvées avec l'approche MO-MA est restreinte et ne dépasse pas 20 alors que dans [103]il a atteint 24 solutions.

L'approche MO-MA permet de trouver l'ensemble des solutions optimales recherchées en un temps considérablement réduit. Pour le temps CPU, nous remarquons qu'en utilisant l'approche MO-MA, le temps écoulé a atteint 32051ms pour trouver une solution Pareto optimale alors qu'il atteint 25925ms en utilisant l'approche MO-GA. En comparant ce résultat avec ceux de [103], nous trouvons le temps d'exécution (TI) de l'algorithme MO-MA est plus petit qu'il s'agit d'un indice reflétant la performance des algorithmes.

Nous avons parlé à titre de comparaison des résultats du MO-MA dans le cas de m = 5 tâches et n = 60 et nous avons estimé les résultats approximativement pour les cas m = 100 tâches et n = 500 services, m = 500 tâches et n = 500 services où nous avons trouvé le TI allant de 17 à

60 ms.Ainsi, on peut dire que les résultats du MO-MA sont prometteurs et très intéressants par rapport aux autres méthodes envisagées. Nous observons que plus le nombre de services augmente, plus de solutions sont trouvées.De plus, à mesure que le nombre de services candidats augmente, le temps écoulé pour trouver les solutions augmente également.

Notre MO-MA proposé est capable d'obtenir des solutions optimales avec un ensemble de solutions Pareto dans un temps de réponse apprécié.D'après l'expérience et l'étude comparative, nous pouvons dire que MO-MA est capable de donner des résultats intéressants et prometteurs pour le problème de la composition des services web.Les résultats montrent une meilleure performance de la méthode MO-MA par rapport aux autres méthodes d'optimisation multi-objectifs pour les deux jeux de données utilisés.

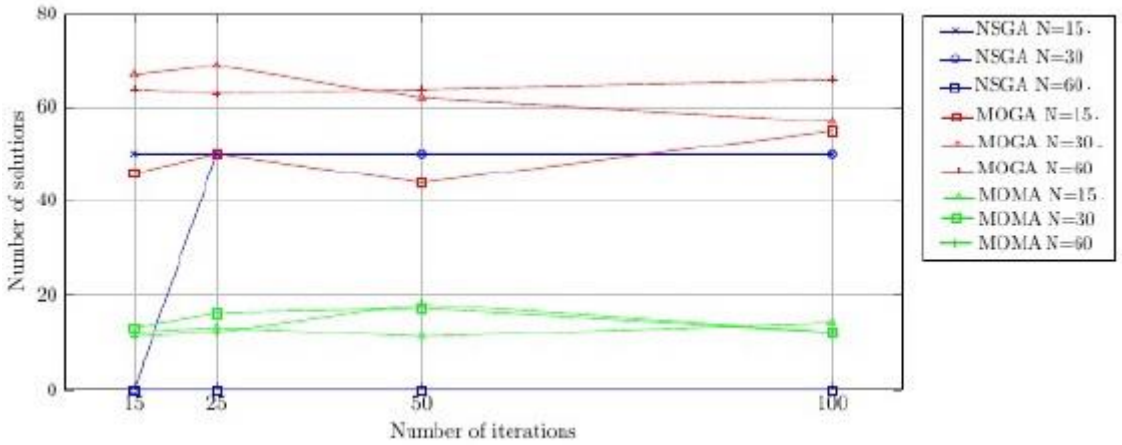

Figure V-20: Ensemble de solutions trouvées par approche

V.6.6 Comparaison supplémentaire avec MO-ACO

Dans cette section, nous comparons notre méthode MO-MA avec la méthode évolutive MO-ACO proposée par [104]. MO-ACO est un algorithme d'optimisation multi-Objectif des colonies de fourmis proposé par Wei et al. Ils ont conçu une approche pour gérer le problème de la composition des services web et où le but est de trouver un bon compromis entre plusieurs objectifs.

Avant de comparer les deux méthodes, nous donnons quelques définitions et notations utiles utilisées par les auteurs dans leurs travaux[104]:

• Le service abstrait (AS) est une fonctionnalité séquentielle correspondant à une tâche.

• Le service concret (CS) est une instanciation d'un service abstrait. Le service concret est obtenu en implémentant la fonctionnalité spécifiée dans le service abstrait.

• Service candidat : est la liste des services candidats avec une fonctionnalité commune.

• Plan abstrait d'exécution (AEP) : étant donné une combinaison de services, l'AEP est un chemin séquentiel du point de départ au point d'arrivée dans un processus fonctionnel. Le graphe fonctionnel est un service connexe. Il illustre toutes les combinaisons de services sélectionnés à partir d'ensembles de services candidats.

Nous précisons que l'auteur dans l'ouvrage [104] a utilisé un modèle basé sur le workflow. Un Workflow est un ensemble de tâches à effectuer dans un ordre précis. Le modèle proposé comprend un ensemble de services candidats pour chaque service abstrait. Les services web appropriés sont sélectionnés pour former le plan d'exécution de la composition du service web. Dans d'autres travaux, une composition de service dynamique basée sur la QoS a été proposée pour trouver la composition de service qui optimise la fonction d'utilité globale pour la QoS. Pour une comparaison équitable, nous avons utilisé les mêmes concepts que ceux de [104]. Nous avons mené une expérience pour évaluer les performances de notre approche par rapport à la méthode MO-ACO. Pour cela, nous avons implémenté quelques plans abstraits d'exécution possibles de services abstraits.

Par exemple, nous donnons les deux plans d'exécution suivants AEP1 et AEP2, avec respectivement 9 et 11 services abstraits (tâches).

AEP1 = {AS1, AS2, AS4, AS7, AS9, AS11, AS12, AS13, AS21} où M = 9.

AEP2 = {AS1, AS2, AS4, AS6, AS8, AS8, AS8, AS8, AS8, AS13, AS21} où M = 11.

A chaque tâche en AEP, un service abstrait ASi est associé. Dans les expériences rapportées dans cette section, nous utilisons le Lp comme métrique pour évaluer les solutions obtenues. Les solutions vont être triées selon la valeur Lp sachant que la solution optimale aurait la valeur minimale de Lp.

La métrique Lp est calculée par la formule mathématique suivante :[104].

$$\left[Lp\left(f(\vec{x})\right) = \sum_{i=1}^{k} \left| \frac{\left(f_i^0 - f_i(\vec{x})\right)}{f_i^0} \right| + \sum_{i=k+1}^{m} \left| \frac{\left(f_i^0 - f_i(\vec{x})\right)}{f_i(\vec{x})} \right| \right]^{(1/2)}$$

$$\overrightarrow{f^0} = [f_1^0, f_2^0, \dots, f_m^0]$$

m : c'est le nombre de fonctions objectif, m = 4 dans notre cas.

Nous avons quatre fonctions objectifs : le coût, le temps, la disponibilité et la valeur de réputation à optimiser. Le vecteur f^0 (où f_i^0 désigne l'optimum de la i^{ème}fonction) est idéal pour un problème multi-objectif (MOP), et le point dans R^n qui détermine que ce vecteur est

la solution idéale (utopique), et est par conséquent appelé le vecteur idéal [105]. Le deuxième paramètre utilisé dans cette comparaison est la taille de l'ensemble des services candidats de chaque tâche (appelé s).De plus, nous avons travaillé sur le benchmark généré aléatoirement (Random) comme référence dans notre implémentation (voir Data availability statement).

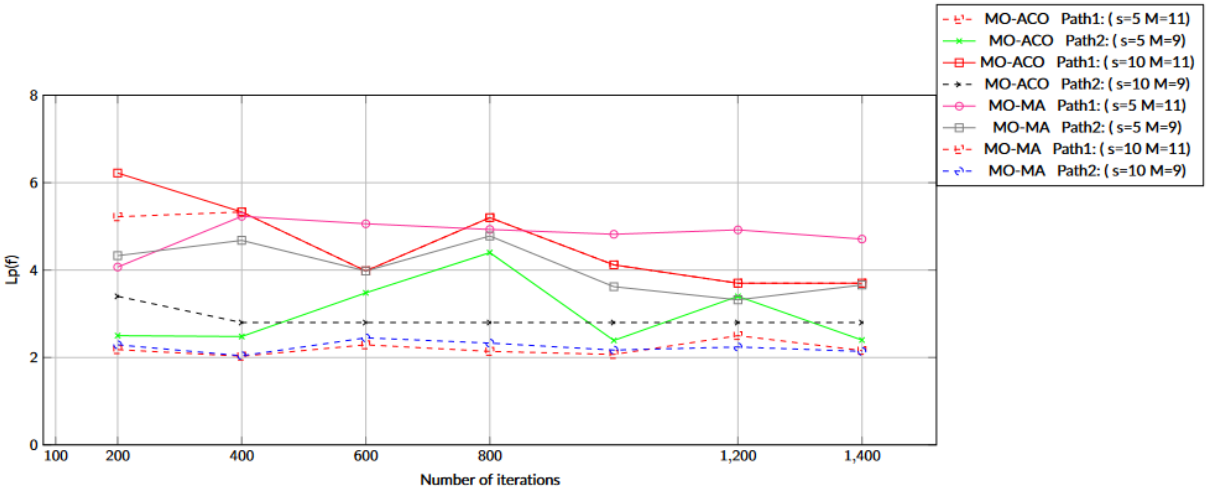

Figure V-20: Comparaison de performance entre MO-MA & MO-ACO

Data availability statement :
Les données qui étayent les conclusions de cette étude sont librement disponibles dans le dossier yacineazouz à l'adresse :

https://github.com/yacineazouz/mon_site_web/blob/master/dataset_randomly_for_web_service_composition.xlsx.

Nous avons considéré quatre cas dans cette comparaison, qui sont :

• Instance 01 = Path1 : (s = 5 M = 11).

• Instance 02 = Path2 : (s = 5 M = 9).

• Instance 03 = Path1 : (s = 10 M = 11).

• Instance 04 = Path2 : (s = 10 M = 9).

Nous traçons les courbes de la Figure V-21 pour comparer les méthodes MO-MA et MA-ACO. Comme le montre la Figure V-28, on peut dire que les valeurs de Lp des solutions optimales trouvées par MO-MA sont très inférieures à celles trouvées par l'approche MO-ACO. Par exemple, en appliquant MO-MA sur l'instance Path1 : (s = 5, M = 11), nous avons trouvé une solution optimale avec Lp = 4,07 en 100 itérations. Cependant, lors de l'application de la méthode MO-ACO, la meilleure solution optimale a une valeur de Lp = 5,22.

A propos de l'instance Path2 : (s = 5, M = 9), les valeurs de Lp des solutions optimales trouvées en appliquant MO-MCO sont meilleures que celles de MO-MA. Par exemple pour le nombre d'itération égal à 200, 400 et 1400, les valeurs de Lp sont respectivement autour de :

2,5, 4,40 et 2,40. Cependant, en appliquant la méthode MO-MA, les valeurs de Lp calculés pour le nombre d'itérations égales à 200, 400 et 1400 sont respectivement : 4,33, 4,78 et 3,66. Concernant l'instance Path1 : (s = 10, M = 11), les valeurs de Lp trouvées par MO-MA à l'itération 200, 400, 1400 sont respectivement : 2.18, 2.03, et 2.16. Lors de l'utilisation de MO-MCO, les valeurs de Lp trouvées sont d'environ 6,22, 5,33 et 3,7. Le même comportement est noté pour l'instance Path2 : (s = 10, M = 9). L'approche MO-MA est meilleure que l'approche MO-MCO et où le La valeur de Lp trouvée par MO-MA est comprise entre 2,04 et 2,45 tandis que les valeurs trouvées par MO-ACO dont les valeurs Lp sont d'environ 2,8.

Dans l'ensemble, nous pouvons dire que la méthode proposée MO-MA est capable de produire des résultats prometteurs pour résoudre le problème de la composition des services Web. Les résultats obtenus montrent une performance intéressante en faveur de notre méthode MO-MA proposée.

V.6.7 Principaux résultats et limites de la recherche

Nous avons proposé une approche multi-objectifs pour résoudre le problème de composition optimale des services web. Ce problème est modélisé comme un problème d'optimisation combinatoire avec quatre fonctions objectifs qui doivent être optimisées. L'objectif est de produire un bon service composite qui minimise les coûts et le temps et maximise la disponibilité et la réputation. Nous avons proposé une méta-heuristique basée sur la recherche locale multi-objectifs (MO-LS) et un algorithme d'évolution génétique (MO-GA) pour traiter notre problème. Puis nous combinons les deux méthodes en une nouvelle, à savoir un algorithme mémétique multi-objectif (MO-MA). Le MO-MA proposé fusionne les principes des deux méthodes LS et GA. Il combine une méthode de recherche locale et des opérations évolutives de croisement sans mutation. La méthode MO-MA qui en résulte est capable d'exploiter les avantages complémentaires de l'efficacité GA qui effectue l'exploration tandis que la méthode de recherche locale effectue une exploitation efficace sur les solutions générées par GA et l'opération de croisement. Les techniques proposées sont validées sur certains jeux de données générés aléatoirement et sur le jeu de données QWS bien connu. Les résultats sont prometteurs et montrer les avantages de MO-MA. Ce dernier réussit à trouver de meilleurs résultats que MO-GA et MO-LS.

L'étude comparative avec certaines méthodes de l'état de l'art ont confirmé nos résultats. Lorsque la méthode LS est intégrée à la méthode GA, les performances sont améliorées. Par conséquent, nous pouvons dire que la nouvelle méthode MO-MA proposée est compétitive et

capable de résoudre le problème mentionné ci-dessus. La méthode MO-MA tire sa force à la fois de la recherche globale assurée par l'algorithme GA et de l'exploitation faite par la méthode de recherche locale LS. Nous notons que la méthode MO-MA proposée est capable de produire de bonnes solutions en un temps raisonnable. Cependant, MO-MA en tant que méthode approchée a quelques inconvénients et limites qui fragilisent les résultats, en particulier lorsque nous traitons de grandes instances du problème. Par conséquent, il est important de concevoir des méthodes appropriées pour éliminer ces limitations et être capable de résoudre des instances à grande échelle.

V.7 Conclusion et travaux futures

Dans ce chapitre, nous avons proposé trois solutions méta-heuristiques pour résoudre le problème de composition optimale des services web : recherche locale, algorithme génétique et algorithme mémétique.L'approche mémétique multi-objectifs proposée permet de trouver un résultat apprécié d'une combinaison de services qui répond à un ensemble de contraintes et d'exigences de l'utilisateur.La méthode MO-MA a été évaluée sur deux jeux de données.Les résultats numériques sont encourageants et démontrer les avantages de la méthode proposée dans la composition de services web.Nous prévoyons d'améliorer le travail actuel en appliquant une hyper-heuristique qui cherche à automatiser par l'incorporation de techniques d'apprentissage automatique, le processus de sélection, de combinaison, de génération ou adapter plusieurs heuristiques plus simples pour résoudre le problème de composition de service web.

Conclusion Générale

Dans cette thèse, nous avons proposé principalement trois approches méta-heuristiques pour le problème d'optimisation de la composition de services web. Un service web composite est une structure parapluie regroupant plusieurs autres services web élémentaires et composites, qui interagissent les uns avec les autres selon le modèle de processus.

Dans les approches proposées, les services composites sont définis sous-forme de matrice (services/taches) et une formalisation simplifiée du modèle QoS est utilisée. .

Le problème de la composition de services web est expérimenté sous forme d'un problème d'optimisation multi-objectif et les algorithmes proposés sont appliqués pour sa résolution.

Le codage des solutions est binaires où chaque bit code la présence ou l'absence d'un service candidat dans la composition indépendamment des autres services.

Pour évaluer chaque solution candidate de composition des services web, nous avons utilisé quatre fonctions objectifs constituées des quatre critères de modèle QoS présenté dans [106]permettant de rechercher l'ensemble de services composite qui optimise les valeurs de ces fonctions objectifs.

L'implémentation a été faite pour les algorithmes multi-objectifs proposés et des expérimentations ont été réalisées en utilisant deux types de dataset (dataset de [85] et un deuxième généré aléatoirement).

Dans toutes les expérimentations faites sur les approches proposées, nous pourrons déterminer très nettement que les solutions pareto-optimale par l'approche MOMA-SLS sont très performantes et les résultats de cette dernière en général sont très satisfaisants. Cela a été conclu par une publication sur le journal international « Expert System » d'un papier journal dont le titre est « Multi-objective memetic approach for the optimal services web composition ».

De point de vue de la qualité des solutions optimales trouvées qui mesurée par les valeurs des fonctions objectifs, les résultats sont indépendants de type de datasets utilisés ou de nombre de services candidats pour chaque tache à réaliser pour le client.

Après la réalisation de notre travail, nous avons acquis beaucoup de connaissances sur les domaines de services web, composition de services web, l'optimisation multi-objectif, les méta-heuristiques et sur l'implémentation des algorithmes SLS, GA, MA, LS, VNS.

Comme perspectives à ce travail, nous prévoyons :

- L'ajout d'une hyper-heuristique et l'incorporation de techniques d'apprentissage automatique pour automatiser par le processus de sélection, de combinaison services.

- L’introduction du concept de parallélisme dans le processus de composition de services web pour exécuter la chaîne de tâches, cela permet d’avoir un ou plusieurs sous-ensembles de tâches de s’exécuter en même temps. Le calcul parallèle consiste en l'exécution simultanée de plusieurs tâches afin de pouvoir être répartie entre plusieurs processeurs en vue de traiter plus rapidement le problème d’optimisation multi-objectif de la composition de services web.
- L’introduction du concept de Cloud Computing. L'émergence de services web avec une qualité différente mais des fonctionnalités similaires a apporté de nouveaux défis à notre problème d’optimisation de la composition des services. Sur ce, nous proposons une approche d'optimisation multi-objectif parallèle basée dans un environnement distribué Spark.

Bibliographies:

[1] H. Holger H et S. Thomas, *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[2] A. T. Chan, J. Cao, et C. Chan, « Webgop: Collaborative web services based on graph-oriented programming », *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 35, n° 6, p. 811‑830, 2005.

[3] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, et H. Chang, « QoS-aware middleware for web services composition », *IEEE Transactions on software engineering*, vol. 30, n° 5, p. 311‑327, 2004.

[4] B. Li, X. Tang, et J. Lv, « The research and implematation of services discovery agent in web services composition framework », in *2005 International Conference on Machine Learning and Cybernetics*, IEEE, 2005, p. 78‑84.

[5] Z. Maamar, S. K. Mostefaoui, et H. Yahyaoui, « Toward an agent-based and context-oriented approach for Web services composition », *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, n° 5, p. 686‑697, 2005, doi: 10.1109/TKDE.2005.82.

[6] P. Maheshwari, H. Tang, et R. Liang, « Enhancing web services with message-oriented middleware », in *Proceedings. IEEE International Conference on Web Services, 2004.*, IEEE, 2004, p. 524‑531.

[7] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz, et B. A. Hamilton, « Reference model for service oriented architecture 1.0 », *OASIS standard*, vol. 12, n° S 18, 2006.

[8] E. Newcomer et G. Lomow, *Understanding SOA with Web Services*. in Independent technology guides. Addison-Wesley, 2005. [En ligne]. Disponible sur: https://books.google.dz/books?id=YhZTAAAAMAAJ

[9] E. Thomas, « Service-Oriented Architecture Concepts , Technology , and Design », 2005. [En ligne]. Disponible sur: https://www.arcitura.com/wp-content/uploads/2017/09/Erl_SOABook2_Ch07-2.pdf

[10] W. Wentz, « Service-Oriented Architecture (SOA), Web Services, and Microservices ». 22 août 2021. [En ligne]. Disponible sur: https://wentzwu.com/2021/08/22/service-oriented-architecture-soa-web-services-and-microservices/
[11] L. Kathryn B and L. Kenneth "Service oriented architecture." Wiley Interdisciplinary Reviews: Computational Statistics 1.1 (2009): 101-105.
[12] M. H. Valipour, B. Amirzafari, K. N. Maleki and N. Daneshpour, « A brief survey of software architecture concepts and service oriented architecture », *2nd IEEE International Conference on Computer Science and Information Technology*, Beijing, Chine, p. 34-38, 2009.
[13] Sprott, David, and L. Wilkes. "Understanding service-oriented architecture." *The Architecture Journal* 1.1 (2004): 10-17.
[14] A. D. DEHANE et Z. KEBIR, « Evaluation des techniques de codage d ontologies sur les performances de la composition de services Web », Université Abou Bakr Belkaid Tlemcen Faculté des Sciences Département d'Informatique, 2012.
[15] Hirsch, Frederick, J. Kemp, and J. Ilkka. Mobile web services: architecture and implementation. John Wiley & Sons, 2007.
[16] Alonso, G., Casati, F., Kuno, H., Machiraju, V., « Web Services », *Data-Centric Systems and Application*, Berlin, Heidelberg, 2004.
[17] P. Louridas, "SOAP and Web Services," in IEEE Software, vol. 23, no. 6, pp. 62-67, Nov.-Dec. 2006, doi: 10.1109/MS.2006.172.
[18] J. Robinson, I. Wakeman, et T. Owen, « Scooby: middleware for service composition in pervasive computing », in *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, 2004, p. 161-166.
[19] J. O'Sullivan, D. Edmond, et A. Ter Hofstede, « What's in a Service? », *Distributed and Parallel Databases*, vol. 12, n° 2, p. 117-133, 2002.
[20] Leymann, F., Roller, D. Modeling business processes with BPEL4WS. ISeB 4, 265–284 (2006). https://doi.org/10.1007/s10257-005-0025-2
[21] C. Severance, "Roy T. Fielding: Understanding the REST Style," in *Computer*, vol. 48, no. 6, pp. 7-9, June 2015, doi: 10.1109/MC.2015.170.
[22] Srinivasan, Latha, and Jem Treadwell. "An overview of service-oriented architecture, web services and grid computing." *HP Software Global Business Unit* 2 (2005): 1-13.
[23] F. Christopher, and J. Farrell. "What are web services?." *Communications of the ACM* 46.6 (2003): 31.
[24] Hubert Kadima, Valérie Monfort, *Les Web services*. Paris: Edition DUNOD, 2003.
[25] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, et others, « Web services description language (WSDL) 1.1 ». Citeseer, 2001.
[26] J. Rao et X. Su, « A survey of automated web service composition methods », in *International Workshop on Semantic Web Services and Web Process Composition*, Springer, 2004, p. 43-54.
[27] V. Portchelvi, V. P. Venkatesan, et G. Shanmugasundaram, « Achieving web services composition—a survey », *Softw Eng*, vol. 2, n° 5, p. 195-202, 2012.
[28] T. H. Nguyen, T. C. Son, et E. Pontelli, « Automatic web services composition for phylotastic », in *International Symposium on Practical Aspects of Declarative Languages*, Springer, 2018, p. 186-202.
[29] T. H. Nguyen, E. Pontelli, et T. C. Son, « On repairing web services workflows », in *International Symposium on Practical Aspects of Declarative Languages*, Springer, 2020, p. 37-53.

[30] A. L. Lemos, F. Daniel, et B. Benatallah, « Web service composition: a survey of techniques and tools », *ACM Computing Surveys (CSUR)*, vol. 48, nº 3, p. 1‑41, 2015.
[31] R. Khalaf, N. Mukhi, et S. Weerawarana, « Service-Oriented Composition in BPEL4WS. », in *WWW (Alternate Paper Tracks)*, 2003, p. 27‑28.
[32] F. Lécué, E. Silva, et L. F. Pires, « A framework for dynamic web services composition », in *Emerging Web Services Technology, Volume II*, Springer, 2008, p. 59‑75.
[33] M. Halilali, « Description, découverte et composition de services Web géospatiaux », Université de Pau et des Pays de l'Adour, Pau et des Pays de l'Adour, 2021.
[34] B. Medjahed, A. Bouguettaya, et A. K. Elmagarmid, « Composing Web services on the Semantic Web », *The VLDB Journal*, vol. 12, nº 4, p. 333‑351, nov. 2003, doi: 10.1007/s00778-003-0101-5.
[35] M. E. Khanouche, Y. Amirat, A. Chibani, M. Kerkar, et A. Yachir, « Energy-centered and QoS-aware services selection for Internet of Things », *IEEE Transactions on Automation Science and Engineering*, vol. 13, nº 3, p. 1256‑1269, 2016.
[36] A. Yachir, « Composition dynamique de services sensibles au contexte dans les systèmes intelligents ambiants », École doctorale Mathématiques, Sciences et Technologies de l'Information et de la Communication (Champs-sur-Marne, Seine-et-Marne), Laboratoire Images, Signaux et Systèmes Intelligents (Créteil), 2014.
[37] C. Peltz, « Web services orchestration and choreography », *Computer*, vol. 36, nº 10, p. 46‑52, 2003.
[38] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, et X. Xu, « Web services composition: A decade's overview », *Information Sciences*, vol. 280, p. 218‑238, 2014.
[39] T. Osman, D. Thakker, et D. Al-Dabass, « Bridging the gap between workflow and semantic-based web services composition », *6789@ ABCDE FGHC6D• I*, p. 13, 2005.
[40] I. Taylor, M. Shields, et I. Wang, « Distributed p2p computing within triana: A galaxy visualization test case », in *Proceedings International Parallel and Distributed Processing Symposium*, IEEE, 2003, p. 8-pp.
[41] F. Leymann, *Web Services Flow Language (WSFL 1.0)*. ResearchGate. 2001.
[42] S. Thatte, « XLANG Web Services for Business Process Design », ResearchGate. janv. 2001.
[43] Nickolas Kavantzas, Oracle, David Burdett, Commerce One, Gregory Ritzinger, Novell, Tony Fletcher, Choreology, Yves Lafon, W3C, et Charlton Barreto, Adobe Systems Incorporated, « Web Services Choreography Description Language Version 1.0 », *W3C Candidate Recommendation*, 9 novembre 2005.
[44] D. Martin *et al.*, « Bringing semantics to web services: The OWL-S approach », in *International Workshop on Semantic Web Services and Web Process Composition*, Springer, 2004, p. 26‑42.
[45] N. K. Neapolitan, R., *Foundations of Algorithms using C++ Pseudo code (3rd ed.)*, 3rd Edition. 2004. [En ligne]. Disponible sur: https://www.amazon.com/Foundations-Algorithms-Using-C-Pseudocode/dp/0763723878
[46] C. Jansson et O. Knüppel, « A branch and bound algorithm for bound constrained optimization problems without derivatives », *Journal of Global Optimization*, vol. 7, nº 3, p. 297‑331, 1995.

[47] I. H. Toroslu et A. Cosar, « Dynamic programming solution for multiple query optimization problem », *Information Processing Letters*, vol. 92, n° 3, p. 149‑155, 2004, doi: https://doi.org/10.1016/j.ipl.2004.06.018.
[48] S. Balev, N. Yanev, A. Fréville, et R. Andonov, « A dynamic programming based procedure for the multidimensional 0-1 knapsack problem », *European Journal of Operational Research*, vol. 186, p. 63‑76, févr. 2008, doi: 10.1016/j.ejor.2006.02.058.
[49] R. Marti, M. Gallego, et A. Duarte, « A branch and bound algorithm for the maximum diversity problem », *European Journal of Operational Research*, vol. 200, p. 36‑44, janv. 2010, doi: 10.1016/j.ejor.2008.12.023.
[50] A. A. Keller, « Mathematical Optimization Terminology-Chapter 1 - Elements of Mathematical Optimization », *ScienceDirect*, p. 1‑12, 2018.
[51] R. Roman Denysiuk, « Evolutionary Multiobjective Optimization: Review, Algorithms, and Applications ». [En ligne]. Disponible sur: https://www.researchgate.net/publication/305409038_Evolutionary_Multiobjective_Optimization_Review_Algorithms_and_Applications
[52] Y. Azouz et D. Boughaci, « An effective multi-objective hybrid local search method for the optimal web services composition », in *2020 4th International Symposium on Informatics and its Applications (ISIA)*, 2020, p. 1‑6. doi: 10.1109/ISIA51297.2020.9416533.
[53] X. Gandibleux et M. Ehrgott, « 1984-2004–20 years of multiobjective metaheuristics. but what about the solution of combinatorial problems with multiple objectives? », in *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 2005, p. 33‑46.
[54] M. Aïder et A. Skoudarli, *Le problème de tournées de véhicules multi-objectif avec incertitude*. Editions universitaires europeennes; ISBN: 978-3-8416-7393-02016.
[55] Russell Stuart J., P. Norvig, et E. Davis, *Artificial intelligence: a modern approach*, Prentice Hall. New Jersey, 1995.
[56] G. Laporte et I. H. Osman, « Routing problems: A bibliography », *Annals of operations research*, vol. 61, n° 1, p. 227‑262, 1995.
[57] S. Voß, S. Martello, I. H. Osman, et C. Roucairol, « Meta-heuristics: Advances and trends in local search paradigms for optimization », 1999.
[58] A. Lazar et R. G. Reynolds, « Heuristic knowledge discovery for archaeological data using genetic algorithms and rough sets ». Department of computer Science, Wayne State University, 2003. [En ligne]. Disponible sur: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.387.2961&rep=rep1&type=pdf
[59] Ezugwu, A.E., Shukla, A.K., Nath, R. *et al*. Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. *Artif Intell Rev***54**, 4237–4316 (2021). https://doi.org/10.1007/s10462-020-09952-0.
[60] M. Birattari, L. Paquete, T. Stützle, et K. Varrentrapp, « Classification of Metaheuristics and Design of Experiments for the Analysis of Components », 2001.
[61] R. J. Vaessens, E. H. Aarts, et J. K. Lenstra, « A local search template », *Computers & Operations Research*, vol. 25, n° 11, p. 969‑979, 1998.
[62] R. Congram, C. Potts, et S. Van De Velde, « An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem », *INFORMS Journal on Computing*, vol. 14, p. 52‑67, févr. 2002, doi: 10.1287/ijoc.14.1.52.7712.

[63] F. Glover et C. McMillan, « The general employee scheduling problem. An integration of MS and AI », *Computers & Operations Research*, vol. 13, nº 5, p. 563‑573, 1986, doi: https://doi.org/10.1016/0305-0548(86)90050-X.
[64] N. Mladenović et P. Hansen, « Variable neighborhood search », *Computers & operations research*, vol. 24, nº 11, p. 1097‑1100, 1997.
[65] E. Bonabeau, M. Dorigo, G. Theraulaz, et G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999.
[66] P. K. Tripathi, S. Bandyopadhyay, et S. K. Pal, « Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients », *Information sciences*, vol. 177, nº 22, p. 5033‑5049, 2007.
[67] C. Voudouris et E. Tsang, « Partial constraint satisfaction problems and guided local search », *Proc., Practical Application of Constraint Technology (PACT'96), London*, p. 337‑356, 1996.
[68] Z. Beheshti et S. M. H. Shamsuddin, « A review of population-based meta-heuristic algorithms », *Int. J. Adv. Soft Comput. Appl*, vol. 5, nº 1, p. 1‑35, 2013.
[69] E.-G. Talbi, « A taxonomy of hybrid metaheuristics », *Journal of heuristics*, vol. 8, nº 5, p. 541‑564, 2002.
[70] B. Selman, H. A. Kautz, B. Cohen, et others, « Noise strategies for improving local search », in *AAAI*, 1994, p. 337‑343.
[71] D. T. Connolly, « An improved annealing scheme for the QAP », *European Journal of Operational Research*, vol. 46, nº 1, p. 93‑100, 1990.
[72] M. Fielding, « Simulated annealing with an optimal fixed temperature », *SIAM Journal on Optimization*, vol. 11, nº 2, p. 289‑307, 2000.
[73] M. G. Resende et C. C. Ribeiro, « Greedy randomized adaptive search procedures », *Kluwer Academic Publishers*, p. 219‑249, 2002.
[74] M. J. N. E. Correspondant, « Algorithmes génétiques et autres outils d'optimisation appliqués à la gestion du trafic aérien », PhD Thesis, Université d'Angers, 2004.
[75] P. Rebreyend, « Algorithmes génétiques hybrides en optimisation combinatoire », PhD Thesis, Ecole normale supérieure de lyon-ENS LYON, 1999.
[76] H. H. Hoos et T. Stützle, *Stochastic local search: Foundations and applications*. Elsevier, 2004.
[77] J. C. H. Hernandez, « Algorithmes métaheuristiques hybrides pour la sélection de gènes et la classification de données de biopuces », PhD Thesis, Université d'Angers, 2008.
[78] P. Moscato et others, « On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms », *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.
[79] C. Aranha *et al.*, « Metaphor-based metaheuristics, a call for action: the elephant in the room », *Swarm Intelligence*, vol. 16, nº 1, p. 1‑6, mars 2022, doi: 10.1007/s11721-021-00202-9.
[80] R. Silvia, H. Malte, et G. Charles, *KI 2007: Advances in Artificial Intelligence: 30th Annual German Conference on AI*. Osnabr¨uck, Germany, 2007.
[81] B. David, B. David R, et R. M. Ralph, « An overview of genetic algorithms: Part 1, fundamentals ». University computing, 1993. [En ligne]. Disponible sur: https://orca.cardiff.ac.uk/id/eprint/64436/1/ga_overview1.pdf

[82] D. B. Claro, P. Albers, et J.-K. Hao, « Selecting Web Services for Optimal Composition. », in *SDWP@ ICWS*, 2005.
[83] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, et Q. Z. Sheng, « Quality driven web services composition », in *Proceedings of the 12th international conference on World Wide Web*, 2003, p. 411‑421.
[84] E. Al-Masri et Q. H. Mahmoud, « Discovering the Best Web Service », in *Proceedings of the 16th International Conference on World Wide Web*, in WWW '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 1257‑1258. [En ligne]. Disponible sur: https://doi.org/10.1145/1242572.1242795
[85] E. Al-Masri et Q. H. Mahmoud, « QoS-based Discovery and Ranking of Web Services », in *2007 16th International Conference on Computer Communications and Networks*, 2007, p. 529‑534. doi: 10.1109/ICCCN.2007.4317873.
[86] M. van Steenderen, « Universal description, discovery and integration », *SA Journal of Information Management*, vol. 2, nº 4, 2000.
[87] R. Chinnici, J.-J. Moreau, A. Ryman, et S. Weerawarana, « Web services description language (wsdl) version 2.0 part 1: Core language », *W3C recommendation*, vol. 26, nº 1, p. 19, 2007.
[88] A. Ryman, « Simple object access protocol (SOAP) and Web services », in *Proceedings of the 23rd international conference on Software engineering*, 2001, p. 689.
[89] The DAML Services Coalition, « DAML-S (and OWL-S) 0.9 Draft Release ». 2003. [En ligne]. Disponible sur: http://www.daml.org/services/daml-s/0.9/
[90] M. Anis, « Recherche à voisinage variable pour des problèmes de routage avec ou sans gestion de stock », Universite de Valenciennes et du Hainaut-Cambresis, France, 2014.
[91] A. OUALI, « Méthodes hybrides parallèles pour la résolution de problèmes d'optimisation combinatoire : application au clustering sous contraintes », l'Université de Caen Normandie & l'Université d'Oran 1 Ahmed Ben Bella, Algérie, 2017.
[92] P. Hansen et N. Mladenovic, « Variable neighborhood search: Principles and applications », *European Journal of Operational Research*, vol. 130, p. 449‑467, févr. 2001, doi: 10.1016/S0377-2217(00)00100-4.
[93] Z. Sevkli et F. E. Sevilgen, « A Hybrid Particle Swarm Optimization Algorithm for Function Optimization », in *Applications of Evolutionary Computing*, M. Giacobini, A. Brabazon, S. Cagnoni, G. A. Di Caro, R. Drechsler, A. Ekárt, A. I. Esparcia-Alcázar, M. Farooq, A. Fink, J. McCormack, M. O'Neill, J. Romero, F. Rothlauf, G. Squillero, A. Ş. Uyar, et S. Yang, Éd., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, p. 585‑595.
[94] Y. Xiao, I. Kaku, Q. Zhao, et R. Zhang, « A reduced variable neighborhood search algorithm for uncapacitated multilevel lot-sizing problems », *European Journal of Operational Research*, vol. 214, nº 2, p. 223‑231, 2011.
[95] J. Brimberg, D. Urošević, et N. Mladenović, « Variable neighborhood search for the vertex weighted k-cardinality tree problem », *European Journal of Operational Research*, vol. 171, nº 1, p. 74‑84, 2006.
[96] P. Hansen, N. Mladenovic, et D. Perez-Britos, « Variable Neighborhood Decomposition Search », *Journal of Heuristics*, vol. 7, p. 335‑350, juill. 2001, doi: 10.1023/A:1011336210885.
[97] J. Lazić, S. Hanafi, N. Mladenović, et D. Urošević, « Variable neighbourhood decomposition search for 0–1 mixed integer programs », *Computers & Operations Research*, vol. 37, nº 6, p. 1055‑1067, 2010.

[98] P. Hansen, J. Brimberg, D. Urošević, et N. Mladenović, « Primal-dual variable neighborhood search for the simple plant-location problem », *INFORMS Journal on Computing*, vol. 19, nº 4, p. 552‑564, 2007.

[99] E. G. Pardo, N. Mladenović, J. J. Pantrigo, et A. Duarte, « Variable Formulation Search for the Cutwidth Minimization Problem », *Applied Soft Computing*, vol. 13, nº 5, p. 2242‑2252, 2013, doi: https://doi.org/10.1016/j.asoc.2013.01.016.

[100] N. Mladenović, F. Plastria, et D. Urošević, « Reformulation Descent Applied to Circle Packing Problems », *Comput. Oper. Res.*, vol. 32, nº 9, p. 2419‑2434, sept. 2005.

[101] D. C. Tulpan, H. H. Hoos, et A. E. Condon, « Stochastic Local Search Algorithms for DNA Word Design », in *DNA Computing*, M. Hagiya et A. Ohuchi, Éd., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, p. 229‑241.

[102] A. Mosavi et A. Vaezipour, « Reactive search optimization; application to multiobjective optimization problems », *Applied Mathematics*, vol. 3, nº 10A, p. 1572‑1582, 2012.

[103] Y. Huo, P. Qiu, J. Zhai, D. Fan, et H. Peng, « Multi-objective service composition model based on cost-effective optimization », *Applied Intelligence*, vol. 48, nº 3, p. 651‑669, mars 2018, doi: 10.1007/s10489-017-0996-y.

[104] W. Zhang, C. K. Chang, T. Feng, et H. Jiang, « QoS-Based Dynamic Web Service Composition with Ant Colony Optimization », in *2010 IEEE 34th Annual Computer Software and Applications Conference*, 2010, p. 493‑502. doi: 10.1109/COMPSAC.2010.76.

[105] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, et others, *Evolutionary algorithms for solving multi-objective problems*, vol. 5. Springer, 2007.

[106] Deb, K., Deb, K, « Multi-objective Optimization », *Search Methodologies*, Boston, 2014. [En ligne]. Disponible sur: https://link.springer.com/chapter/10.1007/978-1-4614-6940-7_15#citeas