République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A. MIRA-BEJAIA
Faculté des Sciences Exactes
Département d'Informatique
Laboratoire d'Informatique Médicale et des Environnements Dynamiques et intelligents (LIMED)

THÈSE

Présentée par

AIT HACENE OUHADDA Souhila

Pour l'obtention du grade de

DOCTEUR EN SCIENCES

Filière : Informatique

Option : Réseaux et Systèmes distribués

Thème

Composition de services avec contraintes non-fonctionnelles dans le contexte des applications médicales.

Soutenue le : 09 juillet 2025

Devant le Jury composé de :

Nom et Prénom	Grade		
Mr AMROUN Kamal	Professeur	Univ. de Bejaia	Président
Mr TARI Abdelkamel	Professeur	Univ. de Bejaia	Rapporteur
Mr ELMIR Youssef	Professeur	ESTIN	Examinateur
Mr BOUYAKOUB Fayçal M'hamed	Professeur	USTHB	Examinateur
Mr AZOUAOU Faical	Professeur	ESTIN	Examinateur
Mme EL BOUHISSI BRAHAMI Houda	MCA	Univ. de Bejaia	Examinatrice
Mme CHIBANI SADOUKI Samia	MCA	Univ. de Bejaia	Invitée

Année Universitaire : 2024/2025

Dédicace

« Je dédie ce travail à toutes les personnes qui me sont chères. »

Remerciement

En premier lieu, je remercie le bon Dieu de m'avoir donné la force, la santé, la volonté et la patience nécessaires pour achever ce travail de recherche.

Je tiens à exprimer ma gratitude et mon appréciation à mon directeur de thèse, le Professeur TARI Abdelkamel, pour m'avoir donné l'opportunité de poursuivre mon doctorat sous sa supervision. Au cours des dernières années, il m'a prodigué des conseils de recherche inestimables et un soutien continu tout au long de mes études doctorales.

Je remercie le Pr AMROUN Kamal de m'avoir fait l'honneur de présider le jury de cette thèse. Je remercie également l'ensemble des membres du jury ; le Pr AZOUAOU Faical et le Pr ELMIR Youssef de l'ESTIN, le Pr BOUYAKOUB Fayçal M'hamed de l'USTHB ainsi que le Dr EL BOUHISSI BRAHAMI Houda de l'université de Bejaia pour avoir accepté de participer à cette soutenance.

Je remercie également le Dr ACHROUFENE Achour, de l'université de Bejaia, pour l'aide et l'orientation qu'il m'a apportées.

Je porte un remerciement particulier à ma collègue et amie, le Dr CHIBANI SADOUKI Samia de l'université de Bejaia, pour sa présence à mes côtés, son soutien personnel et scientifique et ses encouragements tout au long de ce travail de recherche, ainsi que pour avoir accepté d'assister à cette soutenance.

Mes derniers mots de remerciement sont dédiés à ma chère famille et en particulier à mon conjoint dont le soutien constant, l'écoute attentive et les encouragements ont été essentiels à la réalisation de cette thèse. Un clin d'œil à mes adorables enfants, Adlane et Ines, qui ont été une source de motivation et de joie tout au long de ce parcours.

Résumé

Des maisons intelligentes aux villes intelligentes, l'Internet des objets (IoT) a révolutionné nos vies en connectant des milliards d'appareils dans un réseau interconnecté. Chacun de ces objets offre des fonctions spécifiques, telles que le déverrouillage des portes à distance ou l'envoi d'alertes, que nous appelons services IoT. Plusieurs objets interagissent généralement les uns avec les autres pour répondre à une demande complexe de l'utilisateur. Cela implique la composition de plusieurs services IoT. Compte tenu du grand nombre de services IoT fonctionnellement équivalents mais présentant des niveaux de qualité de service (QoS) différents, la composition de services s'impose comme l'un des principaux défis dans les environnements IoT.

Cette thèse propose deux nouvelles approches basées sur des métaheuristiques pour aborder ce problème. La première combine un arbre de décision (DT) avec l'algorithme d'optimisation des baleines (WOA) afin de réduire l'espace de recherche et de retourner une composition de bonne qualité en un temps raisonnable en tenant compte des préférences de QoS des utilisateurs. La seconde approche repose sur l'algorithme d'optimisation des lions (LOA) avec prise en compte des préférences et des contraintes globales de QoS des utilisateurs. La validation de ces deux approches a été réalisée par des scenarii et une série de tests en comparaison avec des approches existantes. Parmi les scénarios explorés, un cas d'étude médical a été particulièrement mis en avant, à savoir celui de la prise en charge d'un patient étranger malentendant, hébergé dans une maison de retraite. Ce patient est équipé d'une prothèse auditive intelligente permettant la collecte et l'échange de données en temps réel et d'un smartphone. C'est dans ce cadre que notre algorithme DALOA intervient, en retournant une composition de services optimisée et adaptée aux besoins spécifiques du patient, tout en garantissant une qualité de service conforme aux exigences médicales.

Mots-clés : Qualité de service, Composition de services, Internet des objets, arbre de décision, Algorithme d'Optimisation de la Baleine, Algorithme d'Optimization des Lions.

Abstract

From smart homes to smart cities, the Internet of Things (IoT) has revolutionized our lives by connecting billions of devices into an interconnected network. Each smart device provides specific functionalities, such as remote door unlocking or sending alerts, which are referred to as IoT services. Multiple devices usually interact with one another to meet a complex user's request. This involves the composition of several IoT services. Given the large number of functionally equivalent services with different quality of service (QoS) levels, the service composition problem remains one of the main challenges in IoT environments.

This thesis addresses this issue by proposing two new metaheuristic-based approaches. First, a Decision Tree (DT) is combined with the Whale Optimization Algorithm (WOA) to reduce the search space and look for a near-to-optimal composition considering the users' QoS preferences. The second proposal is based on the Lion Optimization Algorithm (LOA) to select a composite service while taking into account both user preferences and global QoS constraints. The validation of these two approaches was carried out through scenarios and a series of tests comparing them with existing methods. Among the scenarios explored, a medical use case was particularly emphasized, namely the care of a foreign hearing-impaired patient living in a retirement home. This patient is equipped with a smart hearing aid and a connected smartphone, enabling real-time data collection and exchange. In this context, our DALOA algorithm ensures optimized service composition tailored to the patient's specific needs while guaranteeing a level of quality of service that meets medical requirements.

Keywords: Quality of Service, Service Composition, Internet of Things, Decision Tree, Whale Optimization Algorithm, Lion Optimization Algorithm.

ملخص

من المنازل الذكية إلى المدن الذكية، أحدث إنترنت الأشياء ثورة في حياتنا من خلال ربط مليارات الأجهزة في شبكة مترابطة. يوفر كل جهاز ذكي وظائف محددة، مثل فتح الأبواب عن بُعد أو إرسال تنبيهات، وهو ما نسميه خدمات إنترنت الأشياء. تتفاعل عدة أجهزة بشكل عام مع بعضها البعض للاستجابة لطلب المستخدم المعقد. هذا يعني تركيب العديد من خدمات إنترنت الأشياء. نظرًا لوجود عدد كبير من الخدمات المتكافئة وظيفيًا بقيم مختلفة لجودة الخدمة، تظل مشكلة تركيب الخدمات أحد التحديات الرئيسية في بيئات إنترنت الأشياء.

تتناول هذه الأطروحة هذه المشكلة من خلال اقتراح نهجين جديدين قائمين على الميتاهيوريستيك. أولاً، يتم دمج شجرة القرار مع خوارزمية تحسين الحوت لتقليل مساحة البحث والبحث عن خدمة مركبة قريبة من الأمثل مع مراعاة تفضيلات المستخدمين في جودة الخدمة. أما الاقتراح الثاني فيستند إلى خوارزمية تحسين الأسد لاختيار خدمة مركبة مع مراعاة تفضيلات المستخدمين وقيود جودة الخدمة الإجمالية. تم التحقق من صحة هذين النهجين من خلال سيناريوهات وسلسلة من الاختبارات التي قارنتهما بالطرق الحالية. من بين السيناريوهات التي تم استكشافها، تم التركيز بشكل خاص على حالة استخدام طبية متمثلة في إدارة حالة مريض أجنبي يعاني من ضعف السمع يعيش في دار للمسنين. هذا المريض مزود بسماعة طبية ذكية وهاتف ذكي متصل، مما يتيح جمع البيانات وتبادلها في الوقت الفعلي. في هذا السياق، تضمن خوارزميتنا تكوين خدمة محسّنة ومصممة خصيصًا لتلبية الاحتياجات المحددة للمريض، مع ضمان مستوى من جودة الخدمة يلبي المتطلبات الطبية.

الكلمات المفتاحية: جودة الخدمة، وتكوين الخدمة، وإنترنت الأشياء، وشجرة القرار، وخوارزمية تحسين الحوت، وخوارزمية تحسين الأسد، وخوارزمية تحسين الأسد

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations

- IoT	Internet of Things	L'Internet des Objets
- QoS	Quality of Service	La qualité de services
- DT	Decision Tree	Arbre de décision
- WOA	Whale Optimization Algorithm	Algorithme d'optimisation des baleines
- DT-WOA	Decision Tree with Whale Optimization Algorithm	Arbre de décision avec algorithme d'optimisation des baleines
- LOA	Lion Optimization Algorithm	Algorithme d'optimisation des lions
- DALOA	Discrete Adaptive Lion Optimization Algorithm	Algorithme discret d'optimisation des lions adapté
- RFID	Radio Frequency Identification	Identification par radiofréquence
- SOA	Service Oriented Architecture	Architectures orientées service
- SAW	Simple Additive Weighed function	Fonction additive simple pondérée
- WoT	Web of Things	Web des objets
- ACO	Ant Colony Optimization	Optimisation par colonies de fourmis
- GA	Genetic Algorithm	Algorithme génétique
- GWA	Grey Wolf Algorithm	L'algorithme des loups gris.
- MIP	Mixed Integer Programming	Programmation en nombres entiers mixtes
- SFLA	Shuffled Frog Leaping Algorithm	Algorithme de saut de grenouille mélangé
- FACO	Flying Ant Colony Optimization	Optimisation par colonies de fourmis volantes
- PSO	Particle Swarm Optimization	Optimisation par essaims de particules
- ABC	Artificial Bee Colony	Colonie d'abeilles artificielle
- EHO	Elephant Herding Optimization	Optimisation par regroupement d'éléphants
- NN	Neural Network	Réseaux de neurones
- NBA	Novel Bat Algorithm	Nouvel algorithme de chauve-souris
- MOEA	Multi-Objective Evolutionary Algorithm	Algorithme évolutif multi-objectifs

Chapitre 1

Introduction générale

1.1 Contexte

Durant la dernière décennie, l'Internet des Objets (IoT) a révolutionné de nombreux secteurs, tels que le marketing, l'agriculture, l'industrie et la santé. L'IoT est un concept étroitement lié à l'informatique ubiquitaire, qui offre aux objets physiques la capacité de s'interconnecter, de communiquer et d'échanger des données pour atteindre un objectif commun en exploitant l'infrastructure Internet existante [5,6]. De ce fait, l'IoT constitue le meilleur moyen de connecter le monde physique et virtuel. Grâce à cette technologie, de nombreux objets du quotidien, comme les montres, les cannes, les prothèses auditives, les boîtes à pilules et les caméras, sont désormais équipés de capteurs [7, 8]. Ainsi, ces objets offrent une multitude de services aux utilisateurs, tels que le suivi d'activité physique, la télésurveillance médicale ou la domotique. Un service IoT est une fonctionnalité fournie par un objet intelligent. Il est caractérisé par des propriétés fonctionnelles et non fonctionnelles. Les propriétés non fonctionnelles, appelées qualité de service (Quality of Service) et notées QoS, incluent le temps de réponse, la disponibilité, le débit, la réussite et la fiabilité [1,9]. Divers objets IoT peuvent offrir le même service avec des niveaux de QoS variables [1,6].

En pratique, les demandes des utilisateurs comprennent des exigences fonctionnelles et non fonctionnelles qui sont généralement complexes et ne peuvent être satisfaites par un seul service. Par conséquent, des services IoT à valeur ajoutée sont définis en combinant deux ou plusieurs services atomiques selon un plan de composition. Le processus de composition de services permet de combiner les fonctionnalités des objets IoT pour construire un service IoT composite répondant à la fois aux exigences fonctionnelles et de qualité de service de l'utilisateur [5, 10, 11].

1.2 Exemple illustratif

Considérons un hôpital équipé de divers dispositifs IoT tels que des capteurs de santé, des dispositifs de surveillance, des dispositifs de gestion des ressources ainsi qu'un système de notification. Supposons que chacun de ces objets IoT offre un ou plusieurs services comme suit :

1. **Capteurs de santé**
   - Service de mesure des signes vitaux : les capteurs de santé collectent et transmettent en continu les signes vitaux des patients (température, fréquence cardiaque, saturation en oxygène).
2. **Dispositif de surveillance vidéo**
   - Service de surveillance vidéo en temps réel : enregistre et transmet les flux vidéo en temps réel de la chambre du patient.
   - Service de détection de mouvements : détecte des mouvements inhabituels ou une absence de mouvement prolongée.
3. **Dispositif de gestion de ressources : base de données médicale**
   - Service de stockage de données : stocke les données du patient en temps réel.
   - Service de détection d'anomalies : un algorithme d'analyse en temps réel situé dans la base de données médicale est utilisé pour détecter d'éventuelles anomalies (par exemple, une chute soudaine de la fréquence cardiaque).
4. **Système de notification**
   - Service d'alerte médicale : envoi des alertes au personnel médical via des applications mobiles ou des pagers en cas d'anomalie détectée.
   - Service de notification prioritaire : S'assure que les alertes critiques sont traitées en priorité.

Supposons maintenant qu'un utilisateur (membre du personnel soignant ou médecin) demande à développer une application qui permet de satisfaire la requête suivante : « Surveiller en temps réel les constantes (valeurs des signes vitaux) d'un patient et alerter le personnel médical en cas d'anomalie ».

Pour satisfaire cette demande, une série de tâches spécifiques doit être exécutée en appelant différents services IoT. Dans cet exemple (figure 1.1), la requête de l'utilisateur peut être satisfaite par la combinaison des services suivants : service de mesure des signes vitaux, service de stockage de données, service de détection d'anomalies, et service d'alerte médicale.

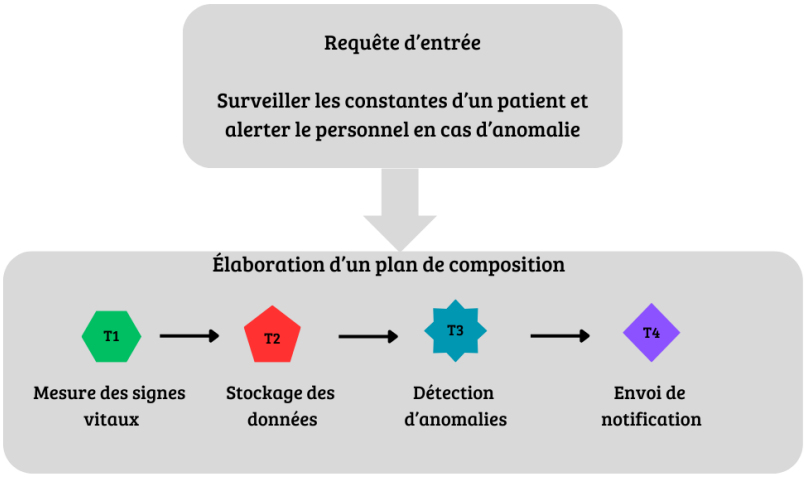

Figure 1.1: Élaboration du plan de composition correspondant à la requête d'entrée

1.3 Problématique

Le processus de composition de services sensible aux QoS se déroule en plusieurs étapes, à savoir : la construction du plan de composition, la découverte de services basée sur les propriétés fonctionnelles, la sélection de services pilotée par les valeurs de QoS, et l'exécution du service composite [1, 12]. Les travaux de cette thèse se concentrent sur l'étape de sélection de services composites basée sur les critères de QoS dans un environnement IoT. Cela consiste à choisir, parmi un ensemble de services IoT fonctionnellement équivalents, celui qui satisfait au mieux les exigences en QoS de l'utilisateur.

La sélection de services lors du processus de composition est un problème combinatoire NP-difficile, comme démontré dans plusieurs études [1, 13–17]. Cela est dû d'une part au nombre considérable de dispositifs IoT offrant des services fonctionnellement similaires avec divers degrés de QoS. D'autre part, la sélection de services basée sur la QoS est considérée comme un problème de prise de décision multicritère, car différents paramètres de QoS sont souvent pris en compte [18]. Dans l'exemple illustratif de la figure 1.1, supposons que l'utilisateur prend en considération les critères de QoS suivants : coût, fiabilité et temps de réponse, et qu'il définit les préférences et contraintes globales de QoS suivantes : le client accorde plus d'importance au coût de la composition et il accorde le même poids de préférences aux deux autres QoS, à savoir la fiabilité et les temps de réponse. Pour les contraintes globales, il demande que le coût total ne dépasse pas 400 euros, le temps de réponse total doit être inférieur ou égal à 3 secondes et enfin la fiabilité totale doit être supérieure ou égale à 99%. Dans ce cas, il faudra chercher

parmi toutes les combinaisons possibles la composition candidate qui satisfait au mieux les exigences de l'utilisateur. Supposons que nous avons m services pouvant réaliser chaque tâche du plan de composition. Sachant que, dans ce scénario, le plan de composition est constitué de 4 tâches, le nombre de compositions candidates possibles est donc, 4^m comme l'illustre la figure 1.2.

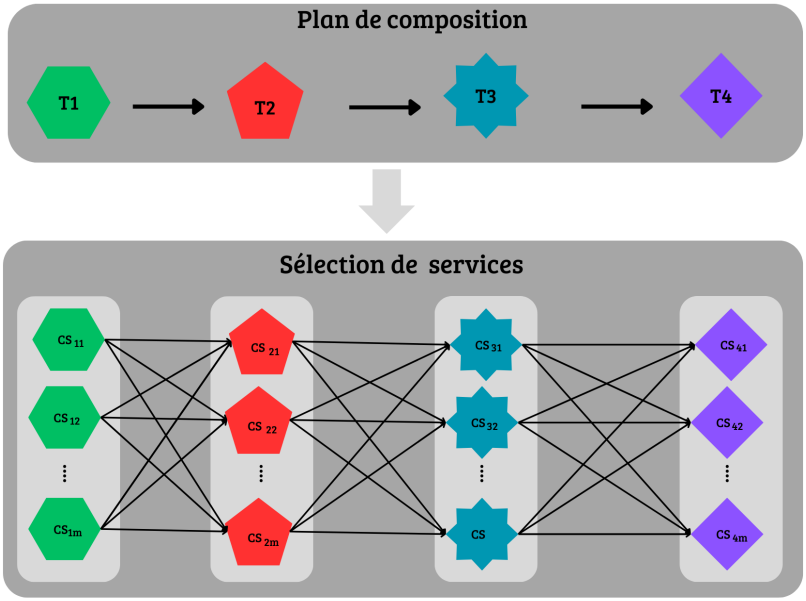

Figure 1.2: Illustration du problème de sélection de services lors du processus de composition

1.4 Objectifs et contributions

La composition de services IoT basée sur les QoS est un enjeu central dans le développement de systèmes IoT intelligents et réactifs. La diversité des dispositifs IoT, la variabilité des environnements et l'évolution des besoins en termes de QoS rendent ce problème particulièrement complexe. Comme la sélection de services à base de QoS est reconnue comme étant un problème NP-difficile [1, 13–17], il est nécessaire de proposer des approches efficaces en termes de temps qui fournissent des solutions proches de l'optimale. Pour cela, de nombreux travaux [1, 19–21] se sont focalisés sur cette problématique et ont proposé des solutions s'appuyant sur des métaheuristiques pour sélectionner de manière efficace les services à inclure dans la solution. Cependant, ces approches se heurtent à quelques défis tels que :

— **Le nombre croissant de services proposant les mêmes fonctionnalités :** L'augmentation de services offrant des fonctionnalités similaires mais avec des niveaux de QoS différents explose le nombre de combinaisons possibles et rend la sélection d'une bonne solution plus compliquée.

— **L'équilibre délicat entre les phases d'exploration et d'exploitation :** les métaheu-

ristiques doivent à la fois explorer de vastes espaces de recherches et exploiter les régions prometteuses pour trouver des solutions de haute qualité en un temps de réponse raisonnable. En effet, un déséquilibre entre ces deux phases peut conduire à des résultats non satisfaisants avec des temps de calcul trop élevés.

— **La prise en compte de contraintes de QoS multiples et parfois conflictuelles :** La composition de services est un problème d'optimisation multi-objectif, où il faut retourner une solution qui satisfait au mieux un ensemble de contraintes de QoS souvent contradictoires, tout en optimisant une fonction d'utilité représentant les préférences de l'utilisateur.

Pour traiter le problème de composition de services basé sur les QoS dans un environnement IoT, nous proposons dans cette thèse deux approches basées sur des métaheuristiques inspirées de la nature :

— Notre premier travail propose une nouvelle approche combinant un arbre de décision (DT) avec l'algorithme d'optimisation des baleines [3] (WOA) nommée DT-WOA [22], pour la composition de services IoT avec prise en compte des préférences de l'utilisateur. L'arbre de décision est utilisé pour classifier les services candidats dans le but de réduire l'espace de recherche en supprimant les services candidats les moins prometteurs par rapport aux besoins en QoS exigés par l'utilisateur, tandis que WOA est employé pour identifier la meilleure solution proche de l'optimale. Les principales contributions de DT-WOA sont : (1) la réduction de l'espace de recherche en éliminant les services les moins pertinents, (2) la proposition d'une version discrète de WOA pour traiter le problème de sélection de services basé sur la QoS, qui implique des variables discrètes, et (3) la validation de DT-WOA à travers un scénario de surveillance d'une maison intelligente.

— Notre second travail propose une adaptation de l'algorithme d'optimisation des lions [23] (LOA), nommée DALOA [12] qui prend en considération à la fois les préférences de l'utilisateur et les contraintes globales de QoS. LOA est une métaheuristique bio-inspirée à base de population de solutions qui imite le mode de vie particulier des lions et leurs comportements coopératifs, en modélisant leurs activités dans la nature, telles que la promenade (roaming), l'accouplement (mating) et la migration. Ainsi, pour offrir un bon équilibre entre l'exploration et l'exploitation et éviter d'être piégé dans des optimums locaux, DALOA combine des stratégies de recherche locales et globales tout en tenant compte des préférences de l'utilisateur et des contraintes globales de QoS. À notre connaissance, LOA n'a jamais été appliqué au problème de la composition de services IoT pilotée par la QoS au moment de la réalisation de notre travail. Les principales contributions de DALOA sont comme suit :

  — Parmi les divers opérateurs de LOA, nous avons sélectionné et adapté les opérateurs de promenade, d'accouplement et de migration en raison de leur complémentarité pour notre problématique. Cette combinaison permet de garantir une diversité au

sein de la population tout en assurant un bon équilibre entre les phases d'exploration et d'exploitation de l'espace de recherche. Cela permet à DALOA d'offrir un bon compromis entre la qualité de la solution retournée et le temps de calcul.

— La définition d'un nouvel opérateur noté $\oplus$ pour discrétiser la métaheuristique LOA afin de traiter le problème de la composition de services IoT, reconnu comme un problème combinatoire à variables discrètes.

— L'évaluation de l'approche proposée à travers une série de simulations basées sur un scénario médical réaliste de composition de services IoT. Celui-ci porte sur la prise en charge d'un patient malentendant, parlant une langue étrangère, hébergé dans une maison de retraite et équipé d'une prothèse auditive intelligente et d'un smartphone.

— L'intégration des contraintes globales de QoS dans DALOA, tout en préservant ses atouts en termes de temps de réponse et de qualité de la solution retournée.

1.5 Structure de la thèse

Le manuscrit de cette thèse est organisé en quatre chapitres comme suit :

- **Chapitre 1 – Introduction générale**

  Le premier chapitre présente le contexte général de notre travail ainsi qu'un exemple illustratif d'une requête complexe nécessitant la composition de divers services pour la satisfaire. Ensuite, la problématique de recherche est formulée de manière précise, en soulignant les défis rencontrés par les approches existantes. Ce chapitre identifie également les objectifs de la thèse et présente ses contributions. Enfin, la structure globale de la thèse est présentée, offrant au lecteur une vue d'ensemble de l'organisation du manuscrit.

- **Chapitre 2 – Composition de services dans les environnements IoT**

  Le deuxième chapitre est organisé en trois parties. La première partie est consacrée à la présentation des concepts fondamentaux de l'IoT. Ensuite, la deuxième partie se penche sur la problématique de la composition de services sensible aux QoS et rappelle quelques définitions importantes liées à ce défi. Enfin, la dernière partie étudie certaines solutions basées sur des métaheuristiques proposées dans la littérature pour résoudre cette problématique.

- **Chapitre 3 – Nos contributions pour la sélection de services IoT sensible aux QoS**

  Le troisième chapitre est dédié aux contributions apportées dans le cadre de cette thèse. Il commence par une présentation de notre première contribution DT-WOA [22] en précisant son principe de fonctionnement. Par la suite, notre seconde contribution, DALOA [12], est

introduite et détaillée.

- **Chapitre 4 – Étude des performances de DT-WOA et DALOA**

  Le dernier chapitre de cette thèse présente l'étude des performances des approches proposées DT-WOA et DALOA ainsi qu'une discussion des résultats obtenus.

- **Conclusion générale**

  Nous clôturons ce manuscrit par une conclusion générale, synthétisant les contributions et résultats obtenus, ainsi que quelques perspectives futures.

Chapitre 2

Composition de services dans les environnements IoT

2.1 Introduction

L'Internet des objets, ou Internet of Things en anglais, regroupe des milliards de dispositifs connectés à Internet tels que des capteurs, des appareils portables, des smartphones et des véhicules. Ces dispositifs interconnectés sont capables de recueillir des informations sur le monde physique et de les transmettre à des applications IoT [24] qui les exploitent pour prendre des décisions, comme l'ajustement de la température dans une maison. Par ailleurs, il convient de préciser qu'un objet IoT peut offrir un ou plusieurs services à l'utilisateur.

Ce chapitre est structuré en trois parties. La première introduit les fondements des technologies IoT. La deuxième partie est consacrée à la présentation des concepts fondamentaux liés au processus de composition de services ainsi qu'aux définitions des concepts de base du problème de sélection de services basé sur les QoS. Enfin, la troisième partie fournit un état de l'art des approches traitant le problème de sélection de services web et de services IoT sensible aux QoS.

2.2 L'Internet of Things

L'IoT est considéré comme une concrétisation technique de la vision de Mark Weiser [25] de l'informatique ubiquitaire. Cette vision se caractérise par une intégration transparente de la technologie dans notre environnement, s'effaçant au point de devenir invisible. Un des avantages de cette nouvelle technologie est qu'elle ne se limite plus à un seul objet, mais au contraire, elle se manifeste sous la forme de dispositifs spécialisés et faciles à utiliser, capables de communiquer via divers types de réseaux.

En 1999, Kevin Ashton [26] a introduit le concept d'Internet des objets dans le but de décrire les objets équipés de puces RFID (identification par radiofréquence). Ces objets possèdent un

identificateur unique ainsi qu'un ensemble d'informations les concernant telles que leur position et leur état actuel, que d'autres objets peuvent consulter [24]. Ce mécanisme d'interconnexion favorise un réseau de communication et d'échange entre les objets, constituant ainsi le fondement de l'Internet des objets [27].

L'évolution de l'IoT avec le temps a permis la connexion d'un grand nombre d'objets distribués à Internet, leur permettant non seulement de fournir des services, mais aussi de collecter des données de manière autonome. Plus qu'une simple collecte de données, les objets IoT sont aussi capables d'analyser les informations collectées, d'en déduire des conclusions et même d'agir en conséquence pour modifier leur environnement. L'IoT s'apparente ainsi à un vaste réseau d'objets intelligents distribués et autonomes, capables de s'adapter et de réagir en temps réel aux besoins de l'utilisateur.

Diverses définitions de l'IoT existent dans la littérature. Dans ce qui suit, nous présentons quelques définitions de cette technologie.

Selon Bassi et Horn [28], l'IoT se définit comme un réseau d'objets interconnectés adressables de manière unique, basé sur des protocoles de communication standard dont le point de convergence est l'Internet.

Han et Zhang [29] définissent l'IoT comme un réseau qui relie et combine les objets en utilisant l'Internet, en suivant les protocoles qui garantissent leurs communications et l'échange d'informations à travers une variété de dispositifs.

Selon Miorandi et al. [30], le terme « Internet des objets » est employé comme mot-clé générique pour couvrir divers aspects liés à l'extension d'Internet et du Web dans le domaine physique, grâce au déploiement généralisé de dispositifs distribués dans l'espace avec des capacités d'identification, de détection et/ou d'actionnement. L'IoT prévoit un futur où les entités numériques et physiques peuvent être reliées, en utilisant des technologies appropriées de l'information et de la communication, pour permettre l'émergence d'une toute nouvelle classe d'applications et de services.

Selon Atzori et al. [31], l'idée de base de l'IoT repose sur l'omniprésence autour de nous d'une variété de choses ou d'objets, tels que des étiquettes d'identification par radiofréquence, des capteurs, des actionneurs, des téléphones portables, lesquels, grâce à des schémas d'adressage uniques, sont capables d'interagir les uns avec les autres et de coopérer avec leurs voisins pour atteindre des objectifs communs.

2.2.1 Définition d'un objet en IoT

Un objet dans un environnement IoT est une entité physique (un lave-linge, un bureau, un portable) qui possède un identifiant unique et propre à lui [27,31]. Cet identifiant est associé à une identité qui décrit à la fois les caractéristiques constantes de l'objet (le type, le poids, la couleur) ainsi que son état, à savoir l'ensemble des caractéristiques qui peuvent changer dans le temps,

telles que la position, et le niveau de batterie [31]. En plus de ses caractéristiques, les objets IoT sont souvent équipés de composants matériels, à savoir des capteurs et des actionneurs [32], qui leur permettent de recueillir des données et d'interagir avec leur environnement.

2.2.2 Les capteurs et les actionneurs

Les capteurs et les actionneurs transforment les objets de notre quotidien en objets connectés. En effet, ils permettent à un objet de percevoir les divers phénomènes qui l'entourent et d'interagir avec son environnement de manière autonome.

Les capteurs

Les capteurs jouent un rôle essentiel en tant que canaux sensoriels, recueillant des informations détaillées sur l'environnement. Ces dispositifs sont capables de percevoir et de mesurer des phénomènes physiques, structurels, environnementaux, chimiques ou biologiques. L'intégration des capteurs avec l'IoT a révolutionné divers secteurs tels que la santé, l'agriculture, les transports, la construction et les villes intelligentes grâce à la collecte de données en temps réel, à la surveillance et au post-traitement des données récoltées [32–34].

Les actionneurs

Les actionneurs IoT sont des dispositifs qui ont pour fonction de convertir les signaux électriques en actions physiques [35]. Ainsi, les actionneurs confèrent aux systèmes IoT la capacité d'interagir avec le monde physique en déclenchant des actions spécifiques comme l'activation de l'éclairage, le déclenchement d'une alarme ou le contrôle d'un dispositif moteur.

2.2.3 Les types d'objets IoT

Dans le domaine de l'IoT, on distingue deux catégories principales d'objets connectés : les objets actifs et les objets passifs.

2.2.3.1 Les objets actifs

Les objets actifs sont dotés de capacités de calcul, de mesure (capteur) et/ou d'action (actionneur) [30]. Ils peuvent également stocker tout ou une partie de leur identité et échanger ces informations avec d'autres objets.

2.2.3.2 Les objets passifs

À la différence des objets actifs, les objets passifs sont dépourvus de capacités de calculs, de mesure, et d'action et ne peuvent être suivis et détectés que par des objets actifs. Leur

identité, à l'exception de l'identifiant, n'est pas stockée dans l'objet lui-même, mais nécessite une infrastructure externe pour son stockage.

En plus des ressources matérielles, un objet IoT possède un ensemble de propriétés telles que la mobilité ou encore le mode d'alimentation qui peuvent servir de base pour d'autres classifications.

2.2.4 Définition d'un service IoT

Dans un environnement IoT, un objet intelligent offre une ou plusieurs fonctionnalités appelées services IoT. Un service IoT offre des fonctionnalités aux utilisateurs avec différents niveaux de qualité de service. Ces services peuvent être interrogés à l'aide d'une requête et fournir une ou plusieurs réponses [1, 11]. Dans notre travail, nous représentons un service IoT sous la forme d'un quadruplet $S =< I, O, Fn, QoS >$, où I désigne l'ensemble des données d'entrée nécessaires à l'appel du service S, O désigne les données de sortie produites à la fin de l'exécution du service S, Fn correspond à la fonctionnalité offerte par le service S, et QoS représente le vecteur des valeurs des paramètres de qualité de service offerts par S.

Synthèse

Les services IoT sont connus pour être hétérogènes, en raison de la diversité des appareils intelligents qui les composent [36], de la variété des protocoles réseaux et des formats de données utilisés par les objets intelligents [37]. Cette hétérogénéité rend la conception et l'intégration d'une couche de communication particulièrement complexes. En effet, cette couche doit répondre à des exigences multiples et parfois contradictoires, telles que la dynamique, le couplage faible, la robustesse, la polyvalence et la flexibilité [36]. À cet effet, diverses architectures ont été utilisées pour masquer l'hétérogénéité du matériel, des logiciels, des formats de données, des technologies et de la communication caractérisant les systèmes IoT. Des études antérieures [6,37,38] indiquent que l'architecture orientée services (SOA) constitue l'un des styles architecturaux les plus adaptés pour traiter l'interopérabilité des entités hétérogènes composant les applications IoT. Ainsi, afin de permettre la composition et la combinaison de services IoT, leur encapsulation dans les services web a été largement adoptée dans la littérature [6].

Après avoir posé les bases de l'IoT en lien avec notre travail, nous allons passer à l'étude du processus de composition et de sélection de services.

2.3 Processus de composition de services IoT

Avant de parler de composition de services IoT, il convient de souligner que le domaine de l'informatique offre une diversité de services, y compris les services web, les services cloud et les services IoT. Cependant, indépendamment de la nature spécifique du service, le processus de composition suit généralement les mêmes étapes. En effet, comme illustré dans la figure

2.1, le processus de composition de services se déroule principalement en quatre étapes [1, 12] : l'élaboration du plan de composition, la découverte des services candidats, la sélection des services concrets, et finalement, l'exécution de la composition.

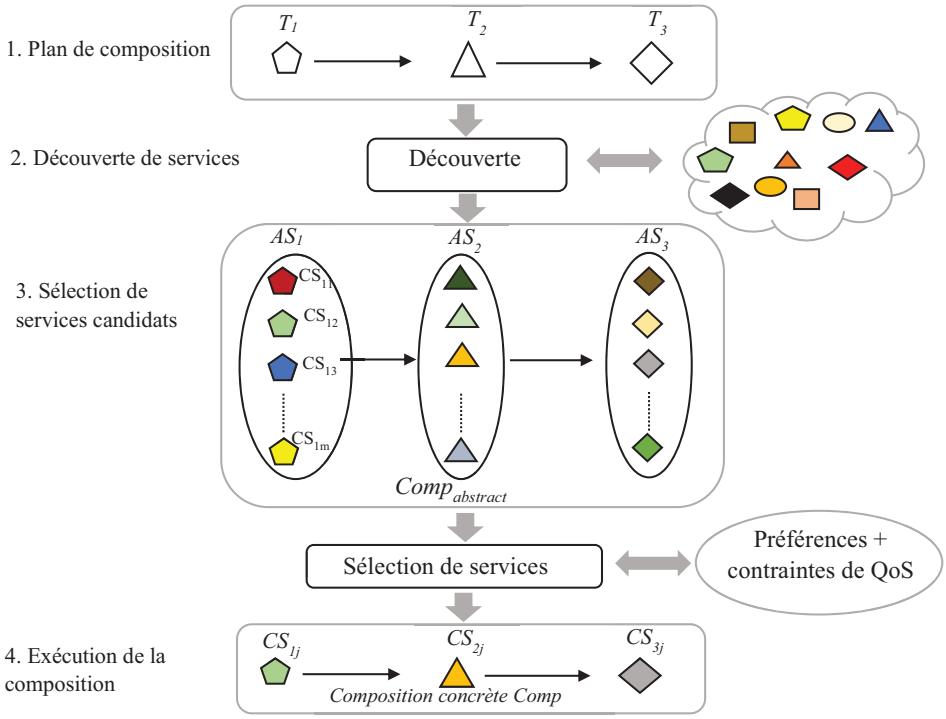

Figure 2.1: Schéma résumant les étapes du processus de composition de services [1]

2.3.1 Étapes du processus de composition de services

Cette partie définit les phases du processus de composition de services avec prise en compte des préférences et contraintes globales de QoS de l'utilisateur.

2.3.1.1 Élaboration du plan de composition (composition abstraite)

À ce niveau, l'accent est mis sur l'identification des exigences fonctionnelles attendues du service résultant de la composition. Ces exigences sont ensuite décomposées pour créer une description abstraite de la composition, comprenant un ensemble de tâches nécessaires pour satisfaire l'utilisateur. En d'autres termes, il s'agit de traduire une demande complexe d'un utilisateur en un enchaînement de tâches.

2.3.1.2 Découverte des services

Cette phase consiste à rechercher les services réels disponibles qui peuvent accomplir les tâches définies dans le plan de composition. Elle permet de recenser tous les services candidats

fonctionnellement équivalents pour les regrouper dans des ensembles appelés services abstraits. La recherche effectuée à cette étape se base généralement sur les descriptions syntaxiques ou sémantiques, en mettant l'accent sur la correspondance fonctionnelle sans prendre en compte les attributs de QoS.

2.3.1.3 Sélection de services en fonction des QoS

La phase de sélection des services commence par choisir, pour chaque tâche du plan de composition, un service candidat parmi ceux disponibles. Ce choix peut être réalisé aléatoirement ou en se basant sur une technique de sélection précise. Les différentes combinaisons de services ainsi obtenues constituent un ensemble de compositions candidates.

2.3.1.4 Exécution et contrôle de la composition

L'exécution de la composition consiste à mettre en œuvre la composition sélectionnée précédemment par une succession d'appels de services et d'échanges de données. Après sa mise en service, cette composition devient accessible aux utilisateurs, qui peuvent l'appeler comme un nouveau service autonome.

Notre travail de recherche porte sur la troisième étape du processus de composition, à savoir la sélection de services sensible aux QoS. En effet, la difficulté de trouver une composition optimale ou proche de l'optimale dans un délai raisonnable augmente avec l'augmentation de la taille de l'espace de recherche. Outre cet enjeu, il faudra considérer un ensemble de contraintes sur les valeurs de qualité de service imposées par l'utilisateur. Cela représente un défi supplémentaire dans la recherche d'une composition qui satisfait à la fois les préférences et les contraintes globales de QoS.

2.3.2 Définitions des concepts fondamentaux liés à la sélection de services sensibles aux QoS

2.3.2.1 Service abstrait et service candidat

Un service abstrait noté AS, désigne une classe de services qui offrent la même fonctionnalité [1, 2], c'est-à-dire ceux qui ont les mêmes paramètres d'entrée/sortie, mais se distinguent par les valeurs de leurs attributs de qualité de service, comme illustré dans la figure 2.1. En d'autres termes, AS représente un ensemble de services candidats fonctionnellement équivalents notés CS. L'équation 2.1 est utilisée pour la représentation formelle d'un service abstrait.

$$AS_i = \{CS_{i1}, ..., CS_{ij}, ..., CS_{im}\} \tag{2.1}$$

où CS_{ij} / $(1 \leq i \leq n)$ et $(1 \leq j \leq m)$ est le j^{ime} service candidat qui peut réaliser la i^{ime} tâche AS_i du plan de composition.

2.3.2.2 Composition abstraite ($Comp_{abstract}$)

Une composition abstraite, aussi appelée plan de composition dans la figure 2.1, est définie comme une séquence de n tâches exécutées en réponse à une requête complexe de l'utilisateur. Chaque tâche de ce plan de composition est appelée service abstrait. L'équation 2.2 est utilisée pour la représentation formelle d'une composition abstraite [1, 12, 15].

$$Comp_{abstract} = \{AS_1, ..., AS_i, ...AS_n\} \tag{2.2}$$

où AS_i fait référence à la i^{ime} tâche du plan de composition.

Les services abstraits dans un plan de composition peuvent être reliés entre eux à l'aide de différentes structures, appelées constructeurs. Nous présentons dans ce qui suit les constructeurs de base d'une composition.

Les constructeurs d'une composition

Quatre modèles de constructeurs (figure 2.2) peuvent être utilisés pour lier les services abstraits d'un plan de composition, à savoir le modèle séquentiel, le modèle en boucle, le modèle parallèle ainsi que le conditionnel [2, 39].

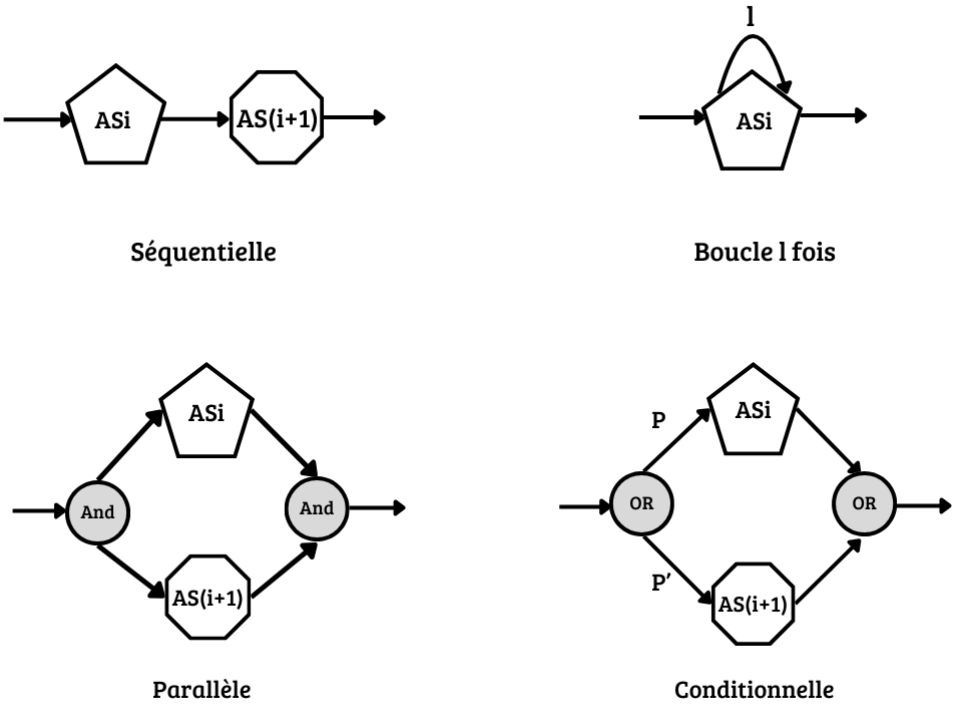

Figure 2.2: Les constructeurs de base d'une composition de services

Lorsque deux services sont reliés via une structure séquentielle, cela signifie que le service de AS_{i+1} est exécuté uniquement après que le service de AS_i a entièrement achevé son exécution.

Dans une structure en boucle, le service de AS_i est exécuté l fois avant que le service suivant ne soit appelé. Un opérateur en boucle peut être perçu comme une composition séquentielle spéciale répétant le même service l fois [2]. La figure 2.2 présente également un constructeur de composition parallèle où les services AS_i et AS_{i+1} s'exécutent simultanément. Par ailleurs, une composition conditionnelle de services signifie que le service AS_i a une probabilité p d'être choisi, tandis que le service AS_{i+1} a une probabilité p', sachant que $p' = 1 - p$.

2.3.2.3 Composition concrète ($Comp$)

Durant l'étape de sélection, un service candidat est choisi pour chaque tâche, et cela forme une composition concrète qui satisfait fonctionnellement les besoins de l'utilisateur. Autrement dit, une composition concrète, abrégée en « composition » et notée $Comp$, est une instanciation du plan de composition obtenue en liant les services candidats sélectionnés pour chaque tâche. En pratique, un service composite concret représente la mise en œuvre d'un service composite abstrait, en invoquant pour chaque service abstrait AS_i le service candidat approprié CS_{ij}, comme le montre l'équation 2.3.

$$Comp = \{CS_{1j_1}, ..., CS_{ij_i}, ..., CS_{nj_n}\} \tag{2.3}$$

où j_i / $(1 \leq j \leq m)$ et $(1 \leq i \leq n)$ représente l'indice du service candidat sélectionné pour effectuer la $i^{\text{ième}}$ tâche du plan de composition.

Dans cette thèse, une composition $Comp$ est représentée par un vecteur de dimension n d'entiers correspondant aux indices des services inclus dans cette dernière, comme illustré dans la figure 2.3. Dans cette figure, le 1^{er} service candidat est choisi pour réaliser la tâche 1 et le $3^{\text{ième}}$ service candidat exécute la tâche AS_2 et ainsi de suite.

2.3.2.4 Les critères de qualité de service (QoS)

Le terme QoS désigne un ensemble d'indicateurs utilisés pour mesurer et décrire certaines caractéristiques d'un service IoT [6, 12] telles que le temps de réponse, le coût, la fiabilité, le débit, la réputation et la disponibilité [39, 40]. Soit CS_{ij} le $j^{\text{ième}}$ service IoT candidat offrant la fonctionnalité de la tâche AS_i. Le vecteur $QoS(CS_{ij})$, présenté dans l'équation 2.4, est utilisé pour représenter les valeurs des r paramètres de QoS offerts par CS_{ij}.

$$QoS(CS_{ij}) = \{q_1(CS_{ij}), ..., q_k(CS_{ij}), ..., q_r(CS_{ij})\} \tag{2.4}$$

où $q_k(CS_{ij})$ / $(1 \leq k \leq r)$ est la valeur du $k^{\text{ième}}$ paramètre de QoS offerte par le service CS_{ij}.

Les critères de qualité de service peuvent être classés en deux catégories : les attributs positifs, tels que la disponibilité, le débit et le taux de réussite, qui doivent être maximisés, et les attributs négatifs, comme la latence, le temps de réponse et le coût, qui doivent être minimisés. Nous présentons ci-après la définition de quelques attributs de QoS.

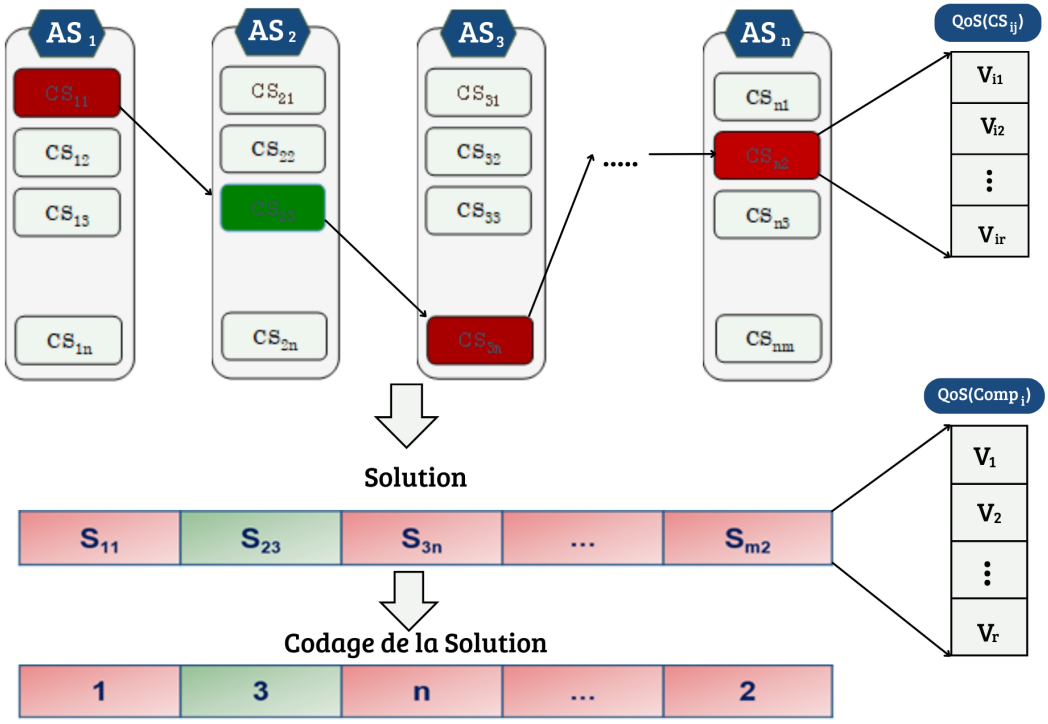

Figure 2.3: Modélisation du problème de sélection des services à base de QoS et codage de la solution [2].

- **Le coût** désigne le prix que l'utilisateur doit payer pour l'invocation du service.
- **Le temps de réponse (ou latence)** représente l'intervalle de temps entre la soumission d'une demande et la réception de la réponse, incluant le retard causé par le réseau lors de l'appel au service.
- **La fiabilité** représente le taux de réussite de la transmission de données aux utilisateurs sur une période donnée.
- **L'emplacement** est la distance entre le service et la destination spécifiée par l'utilisateur.
- **La disponibilité** est un pourcentage qui indique quand le service est disponible.
- **Le taux de réussite** est le rapport entre le nombre de réponses et le nombre de messages.
- **La sécurité** désigne le degré de sécurité fourni par un service, notamment à travers des fonctionnalités telles que l'authentification et le cryptage.
- **Le débit** désigne la quantité de données qui peut être transférée d'un point à un autre dans un réseau par unité de temps, généralement mesurée en octets/sec.
- **La réputation** est une mesure de la crédibilité du service. Elle dépend principalement des expériences d'utilisateurs finaux.

D'autres critères de QoS spécifiques au domaine de l'IoT peuvent être considérés, tels que l'interopérabilité et l'emplacement géographique. L'interopérabilité désigne la capacité d'un service IoT à fonctionner avec divers systèmes et plateformes, ainsi qu'à interagir avec d'autres

services IoT. L'emplacement représente la distance entre le service (l'objet IoT) et la destination spécifiée par l'utilisateur. Il est important de noter que plus cette distance est réduite, plus les données transmises sont précises [6].

Différentes unités de mesure sont utilisées pour quantifier les paramètres de QoS. Pour permettre l'évaluation et la comparaison de ses paramètres, la normalisation de leurs valeurs est nécessaire. L'équation 2.5 est utilisée pour normaliser les paramètres de QoS positifs, tandis que les paramètres de QoS négatifs sont normalisés selon l'équation 2.6.

$$q'_k(CS_{ij}) = \frac{q_k - Qmin}{Q_{max} - Q_{min}} \tag{2.5}$$

$$q'_k(CS_{ij}) = \frac{Qmax - q_k}{Q_{max} - Q_{min}} \tag{2.6}$$

où Q_{max} et Q_{min} sont respectivement les valeurs maximale et minimale possibles pour le $k^{\text{ième}}$ paramètre. q_k est la valeur du $k^{\text{ième}}$ paramètre offerte par le service CS_{ij} et q'_k représente la valeur normalisée du $k^{\text{ième}}$ paramètre.

2.3.2.5 Calcul du vecteur QoS d'une composition

Pour calculer le vecteur QoS d'une composition donnée, deux éléments importants doivent être pris en compte : les constructeurs utilisés pour définir la structure du plan de composition (figure 2.2), ainsi que les paramètres de QoS pris en compte [16]. De ce fait, la valeur d'un paramètre QoS offert par une composition est le résultat de l'agrégation des valeurs des paramètres QoS fournis par chaque service IoT appartenant à cette composition, comme l'illustre la figure 2.4. Chaque composition possède un vecteur QoS nommé $QoS(Comp_i)$ pour représenter les valeurs de ses r paramètres de QoS, comme indiqué dans l'équation 2.7.

$$QoS(Comp_i) = \{q_1(Comp_i), ..., q_k(Comp_i), ..., q_r(Comp_i)\} \tag{2.7}$$

où $q_k(Comp_i)$ est la valeur du $k^{\text{ième}}$ paramètre QoS offerte par $Comp_i$.

Dans ce qui suit, nous détaillons les fonctions d'agrégation utilisées pour le calcul des vecteurs QoS des compositions candidates.

Fonction d'agrégation

Comme le montrent les tables 2.1 et 2.2, la fonction d'agrégation à utiliser pour calculer la valeur QoS d'une composition dépend de deux paramètres : le constructeur et le critère de qualité de service considérés. Rappelons que la variable i $(1 \leq i \leq n)$ est le nombre de tâches du plan de composition, j $(1 \leq j \leq m)$ représente le nombre de services candidats dans un service

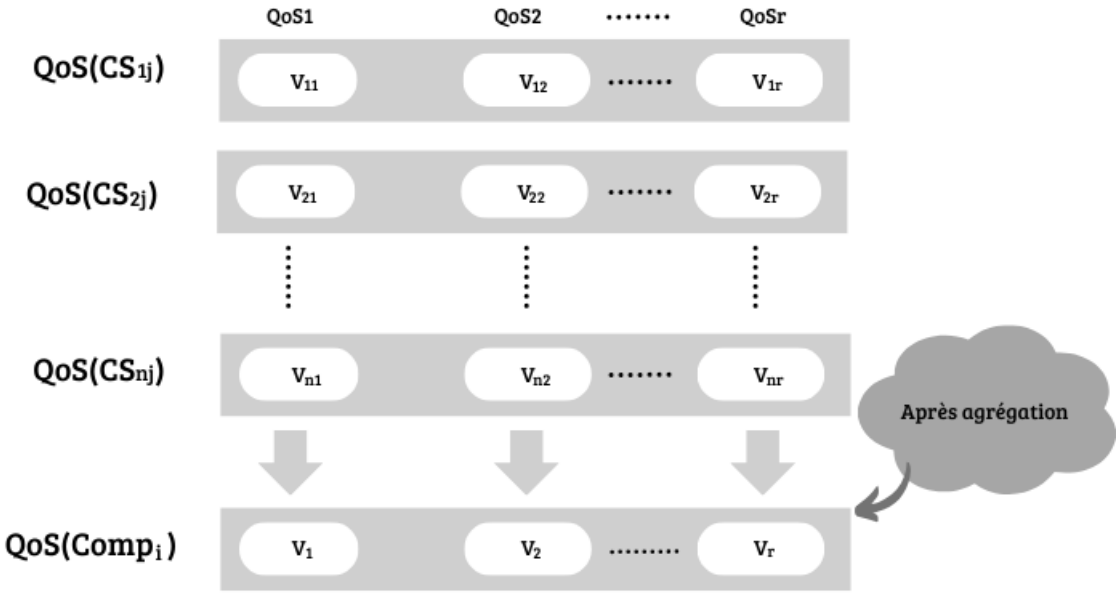

Figure 2.4: Calcul du vecteur QoS d'une composition

abstrait, k $(1 \leq k \leq r)$ est le nombre de QoS considérées, l est le nombre de fois qu'un service est exécuté dans une structure en boucle, et p est la probabilité qu'une branche soit choisie dans une structure conditionnelle.

Structure	Addition	Multiplication	Min/Max
Séquentielle	$\sum_{i=1}^{n} QoS_k(CS_{ij})$	$\prod_{i=1}^{n} QoS_k(CS_{ij})$	$Min/MaxQoS_k(CS_{ij})$
Parallèle	$\sum_{i=1}^{n} QoS_k(CS_{ij})$	$\prod_{i=1}^{n} QoS_k(CS_{ij})$	$Min/MaxQoS_k(CS_{ij})$
Conditionnelle	$\sum_{i=1}^{n} QoS_k(CS_{ij}) * p$	$\sum_{i=1}^{n} QoS_k(CS_{ij}) * p$	$\sum_{i=1}^{n} QoS_k(CS_{ij}) * p$
Boucle	$l * QoS_k(CS_{ij})$	$QoS_k(CS_{ij})^n$	$k * QoS_k(CS_{ij})$

Tableau 2.1: Fonctions d'agrégation utilisées pour le calcul du vecteur QoS d'une composition en fonction de la structure du plan de composition

Des exemples d'agrégation via les fonctions présentées dans la table 2.1 pour quelques attributs de QoS sont présentés dans la table 2.2.

Attribut	Séquentielle	Parallèle	Conditionnelle	Boucle
Temps de réponse	$\sum_{i=1}^{n} RT_{ij}$	$Max(RT_{1j}...RT_{ij})$	$Max(RT_{1j}...RT_{ij})$	$k * RT_{ij}$
Disponibilité	$\prod_{i=1}^{n} A_{ij}$	$\prod_{i=1}^{n} A_{ij}$	$Min(A_{1j}...A_{ij})$	A_{ij}
Débit	$Min(T_{1j}...T_{ij})$	$Min(T_{1j}...T_{ij})$	$Min(T_{1j}...T_{ij})$	T_{ij}
Taux de réussite	$\prod_{i=1}^{n} S_j$	$\prod_{i=1}^{n} S_j$	$Min(S_{1j}...S_{ij})$	S_{ij}
Fiabilité	$\prod_{i=1}^{n} R_{ij}$	$\prod_{i=1}^{n} R_{ij}$	$Min(R_{1j}...R_{ij})$	R_{ij}
Emplacement	$\sum_{i=1}^{n} L_{ij}$	$\prod_{i=1}^{n} L_{ij}$	$Min(L_{1j}...L_{ij})$	$k * L_{ij}$

Tableau 2.2: Fonctions d'agrégation utilisées pour le calcul du vecteur QoS d'une composition en fonction de l'attribut et du constructeur.

2.3.2.6 Fonction d'utilité (ou de fitness)

À partir d'un plan de composition, une multitude de solutions peuvent être générées. Pour évaluer ces solutions et choisir celle qui satisfait au mieux les exigences de l'utilisateur, il est nécessaire d'utiliser une fonction d'évaluation nommée fonction d'utilité ou de fitness. La fonction d'utilité permet d'associer à chaque composition candidate un score numérique. Dans nos travaux de recherche, nous avons utilisé la fonction d'évaluation additive pondérée "Simple Additive Weighed" (SAW) [1, 21, 41]. Avec SAW, la valeur d'utilité d'une composition est calculée en sommant les produits de chaque valeur de QoS par son poids respectif, comme illustré dans l'équation 2.8. Le poids associé au QoS est défini par l'utilisateur pour refléter l'importance accordée à chaque QoS.

$$U(Comp_i) = \sum_{k=1}^{r} pref_k * q_k(Comp_i) \tag{2.8}$$

où $pref_k$ désigne un poids attribué au $k^{\text{ième}}$ paramètre QoS. $q_k(Comp_i)$ est la valeur du $k^{\text{ième}}$ paramètre d'une composition $Comp_i$.

2.3.2.7 Les contraintes globales de QoS

En plus de prendre en considération les préférences de QoS de l'utilisateur, nous avons également considéré les contraintes globales de qualité lors du traitement du problème de sélection de services durant le processus de composition. Les contraintes globales de QoS représentent les exigences de l'utilisateur en termes de valeurs des critères de qualité de service. Elles sont utilisées comme bornes supérieure et inférieure lors de la sélection d'une composition de services candidates. Ces contraintes sont représentées dans notre travail [12] sous la forme d'un vecteur

de r valeurs nommé GC (Global Constraints) et représenté dans l'équation 2.9.

$$GC = \{c_1, ..., c_k, ..., c_r\} \tag{2.9}$$

où c_k, avec $1 \leq k \leq r$, représente la valeur seuil à ne pas dépasser pour le QoS considéré.

Deux types de contraintes globales sont considérés en fonction du critère de QoS. L'équation 2.10 est utilisée pour un paramètre QoS positif, tandis que l'équation 2.11 est utilisée pour un paramètre QoS négatif.

$$q_k(Comp_i) \geq c_k \tag{2.10}$$

$$q_k(Comp_i) \leq c_k \tag{2.11}$$

Les sections précédentes ont permis de définir les concepts fondamentaux liés à la composition de services. Nous passons maintenant à l'étude de certains travaux de recherche traitant le problème de composition de services avec prise en compte des QoS.

2.4 Approches pour la composition de services sensible aux QoS

Comme pour les services web, les services IoT sont encapsulés dans des architectures SOA (Service Oriented Architecture) pour permettre leur interopérabilité et leur combinaison [6]. Cette intégration a conduit à l'émergence du Web of Things (WoT), une approche visant à tirer parti des technologies et protocoles du Web afin de connecter, interpréter et coordonner les objets IoT de manière standardisée et interopérable. De ce fait, nous avons entamé nos recherches par une étude des travaux existants sur la composition de services web. Cette première analyse nous a permis d'acquérir les connaissances nécessaires pour aborder la composition de services dans un environnement IoT.

Diverses classifications des travaux portant sur la composition de services existent dans la littérature. Nous présentons dans ce qui suit des exemples de classification existantes.

- Jatoth et al. [42] classent les approches portant sur la composition de services web, selon la méthodologie, en trois catégories : les approches non-heuristiques (exactes), heuristiques et basées sur des métaheuristiques. La première catégorie vise à fournir des solutions optimales et est appropriée pour les petites instances de problèmes [13, 15–17, 40]. La deuxième catégorie est basée sur des approches heuristiques souvent trop spécifiques et difficiles à réutiliser et à adapter à d'autres problèmes [43–48]. La troisième catégorie, les approches basées sur les métaheuristiques, sont connues pour offrir un bon compromis entre la qualité de la solution et le temps de calcul et retournent en général des solutions proches de l'optimale [39, 49–52].

- Wang et al. [53] se basent sur la stratégie d'optimisation et proposent de classer les travaux traitant la composition de services web en deux catégories : les approches adoptant une démarche d'optimisation locale et les approches basées sur une optimisation globale.
- Khadir et al. [54] classent les approches traitant la composition de services IoT en deux catégories : les approches basées sur une stratégie de sélection globale et les approches basées sur une stratégie de sélection locale.

2.4.1 État de l'art sur les approches de composition de services IoT/Web basées sur les QoS

Les travaux présentés dans cette partie sont classés en deux catégories, à savoir les approches basées sur des techniques déterministes comme la programmation linéaire en nombres entiers et celles qui sont basées sur des métaheuristiques comme Ant Colony Optimization (ACO), Genetic Algorithm (GA), Grey Wolf Optimization (GWO) et Particle Swarm Optimization (PSO).

2.4.1.1 Travaux proposant des solutions basées sur des techniques déterministes

Zeng et al. [40] ont proposé deux algorithmes d'optimisation locale et globale basés sur la programmation linéaire en nombres entiers. L'algorithme d'optimisation locale est utilisé pour sélectionner, pour chaque tâche, le service optimal. Cela signifie qu'à chaque étape du processus, l'algorithme évalue les options disponibles et choisit celle qui offre les meilleures performances selon les critères définis. En revanche, l'algorithme d'optimisation globale sert à sélectionner le plan d'exécution optimal parmi tous les itinéraires possibles, afin de trouver le meilleur service composite impliquant une évaluation plus large et plus complexe. Alrifai et al. [13, 15] se sont penchés sur le problème de la sélection dynamique de services web. Ils ont proposé une solution hybride combinant l'optimisation globale avec des techniques de sélection locale distribuée. Pour l'optimisation globale, les auteurs ont utilisé la programmation en nombres entiers mixtes (MIP) pour décomposer les contraintes globales de QoS en contraintes locales. Ensuite, ils ont utilisé des techniques de sélection locale distribuée pour identifier les meilleurs services web répondant à ces contraintes locales. Dans [13] les auteurs prennent en considération les plans de composition contenant des structures séquentielles seulement, tandis que dans [15], ils prennent en compte aussi les autres structures de plan de composition possibles.

Yu et Bouguettaya [55] ont proposé de réduire l'espace de recherche en exploitant la relation de dominance entre les fournisseurs de services et ont identifié un ensemble des meilleurs services composites parmi toutes les compositions possibles. Pour calculer les services Skyline Composites, Yu et Bouguettaya ont développé trois algorithmes : l'algorithme One Pass (OPA), l'algorithme Dual Progressive (DPA) et l'algorithme Bottom-Up (BUA). L'OPA vise à identifier rapidement

les services dominants, mais il génère des faux positifs, entraînant une perte de temps et d'espace. Le DPA améliore la précision de la sélection des services, mais est coûteux en termes de temps de calcul. Le BUA combine les points positifs du DPA avec une stratégie de composition linéaire et un cadre de calcul ascendant. Gabrel et al. [14] ont traité le problème de composition de services basée sur les workflows et sensible aux QoS en utilisant la programmation linéaire en nombres entiers mixtes et un graphe de dépendance. Ils ont affirmé que la difficulté de cette sélection dépend de la structure de la composition abstraite et du type de QoS pris en compte. Rodriguez-Mier et al. [56] ont proposé une approche hybride pour la composition automatique de services, combinant une recherche heuristique locale quasi-optimale avec une recherche combinatoire globale. Au lieu de trouver une composition avec un nombre fixe de services abstraits, leur méthode consiste à chercher des plans de composition avec le moins de services abstraits possible tout en optimisant les QoS. Leur approche se déroule en quatre étapes principales : la génération du graphe de composition pour une requête, l'identification du plan de composition optimal, la réduction de l'espace de recherche en identifiant les services équivalents et dominés, et enfin la recherche hybride de la meilleure composition avec des valeurs de QoS optimales.

Gabrel et al. [16] ont abordé la complexité du problème de composition de services en fonction du type de QoS considéré, en utilisant un graphe orienté nommé Service-Data. Ils ont démontré que, pour le temps d'exécution et le débit, le problème de composition peut être modélisé comme un problème d'ordonnancement avec des contraintes de précédence AND/OR. Par la suite, ils ont affirmé que ce problème devient NP-difficile pour les critères de coût et de fiabilité. Ghobaei-Arani et Souri [17] ont abordé le problème de la composition de services web dans des environnements cloud géographiquement distribués. Ils ont proposé un algorithme de programmation linéaire, nommé LP-WSC. L'algorithme LP-WSC modélise le problème de composition de services comme un problème d'optimisation linéaire. Il prend en compte les contraintes de QoS spécifiques à chaque requête et utilise des techniques de programmation linéaire pour déterminer la combinaison optimale de services web.

Dans cette étude [57], Wang et al. ont présenté une approche pour résoudre plusieurs défis liés aux systèmes cyber-physiques-sociaux, notamment le coût élevé en temps et la faible fiabilité. L'approche proposée, appelée FRSkyline, repose sur plusieurs étapes clés. Tout d'abord, elle utilise le calcul des composants Skyline pour éliminer les composants redondants et identifier les composants les plus performants. Ensuite, l'approche FRSkyline emploie le coefficient de variation pour évaluer la fluctuation des QoS et garantir une bonne fiabilité. Le coefficient de variation permet de mesurer la dispersion des données de QoS, assurant ainsi que les composants sélectionnés maintiennent une performance stable. Enfin, Wang et al. ont recours à la programmation mixte en nombres entiers pour déterminer la meilleure composition possible, tout en réduisant le temps de calcul. Chen et al. [58] ont proposé un algorithme de composition de services basé sur des techniques de sélection partielle pour réduire l'espace de recherche. Cette approche, appe-

lée DPSA (Dynamic Partial Service Composition Algorithm), combine des techniques d'élagage et de la parallélisation des composants. Les techniques d'élagage sont utilisées pour éliminer les composants non pertinents dès les premières étapes du processus. Ensuite, la parallélisation des composants à plusieurs niveaux permet à DPSA de diviser le processus de composition en sous-tâches parallèles afin de traiter plusieurs composants simultanément. Dans ce travail de recherche [59], Urbieta et al. ont développé un cadre pour la composition de services sensibles au contexte, en utilisant un modèle de service abstrait appelé wEASEL. Ce modèle représente les services et les tâches de l'utilisateur en termes de signature, de spécification et de conversation. L'approche wEASEL permet de décrire les services de manière flexible et modulaire, facilitant ainsi leur composition dynamique en fonction du contexte de l'utilisateur.

Synthèse et discussion

Les approches basées sur des techniques déterministes s'avèrent performantes pour la résolution des problèmes de petites instances, où la solution peut être obtenue dans un temps de calcul raisonnable. Ce type d'approche s'appuie sur l'exploration exhaustive de l'ensemble des combinaisons potentielles avant d'atteindre une solution optimale. De ce fait, cette catégorie d'approches n'est pas adaptée pour les problèmes avec un large espace de recherche, car elles peuvent entrainer des temps de calcul significatifs. Aussi, une seconde catégorie d'approches basée sur des métaheuristiques est proposée pour la résolution de ce défi. Les métaheuristiques, telles que l'algorithme génétique, l'algorithme de colonies de fourmis et l'algorithme des lions, sont des algorithmes d'optimisation génériques conçus pour résoudre des problèmes complexes, souvent NP-difficiles. Ces algorithmes s'appuient sur des stratégies d'optimisation qui s'inspirent de phénomènes naturels et de processus stochastiques. En outre, les métaheuristiques combinent des stratégies d'exploration et d'exploitation de l'espace de recherche pour trouver des solutions proches de l'optimum global, tout en évitant les pièges des optimums locaux. L'exploration correspond à la capacité de parcourir différentes régions de l'espace de recherche, tandis que l'exploitation est la capacité d'obtenir des solutions de haute qualité au sein d'une région donnée [60]. La performance d'une métaheuristique dépend de sa faculté à maintenir un équilibre efficace entre ces deux phases, conditionnant ainsi sa capacité à générer des solutions pertinentes dans un délai raisonnable.

2.4.1.2 Travaux proposant des solutions basées sur des métaheuristiques

Les métaheuristiques ont été largement employées pour résoudre le problème de la composition de services IoT et web sensibles aux QoS. Dans cette section, nous présentons divers travaux de recherche qui ont abordé cette problématique.

Ding et al. [49] ont proposé une approche d'optimisation globale basée sur l'algorithme génétique, intitulée « Transaction and QoS-aware service Selection » (TQS). TQS est proposée

pour la résolution du problème de composition de service web prenant en compte à la fois les propriétés transactionnelles et les exigences de QoS. Les auteurs introduisent les propriétés transactionnelles d'un service web individuel et des services web composites, ainsi que les règles transactionnelles utilisées pour les composer. Dans [61], les auteurs considèrent les propriétés transactionnelles et les contraintes globales de QoS spécifiées par l'utilisateur, lors de la sélection de services. Ils utilisent ACO pour rechercher une solution proche de l'optimale. De plus, dans cet article, le problème de sélection est modélisé sous la forme d'un graphe acyclique dirigé avec des contraintes. Ainsi, le problème de sélection de services se transforme en recherche du meilleur chemin satisfaisant les contraintes globales.

Sun et al. [50] proposent une approche tenant compte des fluctuations pour la composition prédictive de services web. Contrairement à la plupart des méthodes existantes qui supposent que les QoS des services candidats restent constantes dans le temps, stochastiques, ou limitées, la solution proposée dans cet article tient compte des fluctuations des QoS et modélise leurs variations dans le temps. Pour ce faire, Sun et al. ont combiné un algorithme génétique avec un modèle de prédiction des séries temporelles basé sur ARIMA, permettant d'élaborer des calendriers de composition dynamiques et prédictifs.

Dans [62], Ibrahim et al. ont introduit une approche métaheuristique hybride nommée « Hybrid Shuffled Frog Leaping Algorithm and Genetic Algorithm » (SFGA) combinant l'algorithme de saut de grenouilles avec l'algorithme génétique. L'algorithme de saut de grenouilles est utilisé pour générer les compositions de la population initiale, tandis que GA est utilisé pour explorer l'espace de recherche en mettant à jour des compositions via les opérateurs de mutation et de croisement. L'objectif de SFGA est d'optimiser la composition de services dans un environnement cloud mobile en prenant en compte des critères tels que la consommation d'énergie, le temps de réponse et les coûts. Thangaraj et Balasubramanie [63] ont proposé une approche hybride combinant l'algorithme génétique et l'algorithme de recherche tabou pour le problème de composition de services web. Leur méthode repose sur l'évaluation de paramètres QoS tels que le temps de réponse, le coût et la fiabilité, afin de déterminer le service web le mieux adapté à l'utilisateur. En plus de ces QoS, l'approche proposée prend en compte l'accessibilité du service, le débit et l'interopérabilité entre les services. Les auteurs utilisent le mécanisme de sélection de services basé sur la localisation qui considère la localisation géographique d'un service lors du processus de composition.

Yu et al. [64] travaillent sur la composition de services web dans un environnement multi-cloud. En raison des coûts de communication élevés lors de l'utilisation de services web provenant de différents clouds, les auteurs ont présenté un algorithme glouton ainsi qu'une version améliorée d'ACO. L'objectif de ces algorithmes est de minimiser le nombre de clouds impliqués dans une séquence de composition de services. Alayed, et al. ont proposé dans [65] une amélioration de

l'algorithme d'optimisation par colonies de fourmis pour résoudre le problème de sélection de services web sensible aux QoS. Leur approche repose sur un concept d'échange visant à éviter les pièges d'optimum locaux et à réduire la durée de recherche. En favorisant une meilleure diversité, cette amélioration permet à ACO de retourner des compositions de meilleure qualité.

Les auteurs de [51] ont proposé une approche basée sur Flying Ant Colony Optimization (FACO) appelée Enhanced Flying Ant Colony Optimization (EFACO). L'EFACO a introduit une nouvelle stratégie de vol et un nouveau processus de sélection des voisins, qui améliorent la qualité de la composition obtenue. Les auteurs ont intégré un concept multi-agents dans un système parallèle pour optimiser la composition et obtenir les meilleurs résultats possibles. Xia et al. [66] proposent une approche d'optimisation multi-objectifs avec contraintes pour la composition de services web basée sur les QoS. Les auteurs utilisent une version améliorée de l'algorithme PSO multi-objectifs. Cette approche génère un ensemble de solutions Pareto optimales, répondant aux différentes contraintes imposées par les utilisateurs.

Wang et al. [67] ont proposé un algorithme d'optimisation par essaim de particules amélioré, nommé iPSOA, qui combine plusieurs stratégies. Pour améliorer la diversité de la population et éviter la convergence prématurée de PSO, iPSOA applique une stratégie de mutation non uniforme à la meilleure solution globale. Cette stratégie est utilisée pour mettre à jour la meilleure solution globale dans les zones les moins explorées de l'espace de recherche. De plus, iPSOA intègre également les stratégies d'ajustement adaptatif des poids (Adaptive Weight Adjustment) et du meilleur local d'abord (Local Best First). Ludwig [68] propose une approche combinant l'algorithme PSO amélioré avec l'algorithme de Munkres pour optimiser la sélection de services web dans un contexte complexe où plusieurs objectifs et plans d'exécution doivent être pris en compte. Pour chaque plan, PSO explore différentes combinaisons de services web, tandis que l'algorithme de Munkres identifie l'affectation optimale des tâches aux services web sélectionnés. Dans [69], deux solutions basées sur PSO sont présentées et appliquées pour résoudre le problème de composition de services web en se basant sur des propriétés fonctionnelles et non fonctionnelles. Dans la première solution, les auteurs modélisent le problème de composition en utilisant un graphe orienté où les nœuds représentent les services et les arcs représentent les dépendances entre eux. Tandis que dans la seconde approche, Da et al. ont proposé une approche PSO utilisant un modèle basé sur une stratégie gloutonne.

Xu et al. [70] ont proposé une solution basée sur l'algorithme Artificial Bee Colony (ABC) pour le problème de composition de services web. Ils ont développé une version discrète et améliorée de l'algorithme ABC en utilisant une approche approximative. L'approche proposée a permis de simplifier le problème de sélection de services en réduisant la complexité computationnelle tout en maintenant une bonne qualité des compositions retournées. Arunachalam et Amuthan [71] ont proposé une amélioration du schéma d'optimisation ABC basée sur la similarité cosinus

pour la composition de services web. Cette amélioration vise à déterminer le service candidat optimal à partir du plan de composition. L'algorithme amélioré prend en charge les propriétés transactionnelles ainsi que les contraintes de qualité de service.

D'autres travaux proposés par Chibani et Tari [2,9] ont utilisé la métaheuristique des éléphants, Elephant Herding Optimization (EHO), pour aborder le problème de la composition de services web sensible aux QoS. Dans [9], les auteurs ont modélisé le problème de composition de services comme un problème d'optimisation mono-objectif et ont utilisé la fonction SAW pour évaluer la qualité des compositions. Dans [2], Chibani et Tari ont formulé le problème comme un problème d'optimisation multi-objectif et ont développé, d'abord, une version discrète de l'EHO appelée (D-EHO), puis une version multi-objective du D-EHO appelée (MO-D-EHO) en intégrant une approche Pareto avec une archive externe pour stocker le front de Pareto à chaque itération. Gavala et al. [41] ont proposé de combiner l'algorithme d'optimisation des baleines avec la stratégie de l'aigle (Eagle Strategy ES) pour traiter la composition de services sensible aux QoS avec prise en compte des contraintes globales. La stratégie ES s'inspire du comportement de recherche de nourriture des aigles et se compose de deux paramètres principaux : la recherche aléatoire et la poursuite intensive [72]. L'algorithme WOA s'inspire de la stratégie d'attaque des baleines envers leurs proies, appelée le mécanisme d'attaque au filet à bulles. Ce mécanisme comprend deux phases : l'encerclement de la proie et l'attaque de la proie. Dans l'algorithme ESWOA, la stratégie de l'aigle est utilisée pour maintenir un équilibre approprié entre l'exploration et l'exploitation lors de l'application de l'algorithme WOA. Gavala et al. utilisent la fonction SAW pour évaluer l'utilité d'une composition.

Les auteurs de l'article [73] ont exploré la capacité de plusieurs métaheuristiques, notamment les plus connues comme PSO, ACO et ABC, à maintenir un bon équilibre entre l'exploration et l'exploitation de l'espace de recherche. Ils ont observé que, bien que ces métaheuristiques soient généralement efficaces, elles peuvent parfois échouer à trouver un bon équilibre entre ces deux phases. En effet, dans certains cas, ces méthodes offrent un équilibre limité entre ces deux objectifs souvent conflictuels, à savoir : la recherche de nouvelles solutions (exploration) et l'amélioration des solutions existantes (exploitation).

Après avoir présenté les études traitant la composition de services web, nous passons aux articles traitant la problématique de composition de services dans un environnement IoT. Dans ce contexte, Boucetti et al. [1] ont proposé une approche combinant un algorithme génétique avec les réseaux de neurones (NN) artificiels pour la composition de services IoT tenant compte des contraintes globales de QoS. GA est utilisé pour obtenir une composition théorique idéale avec une optimisation globale de la QoS. Ensuite, les réseaux de neurones éliminent les services IoT inappropriés, ne conservant que ceux qui correspondent aux catégories des services théoriques établies précédemment. Kouicem et al. [21] ont développé une approche pour la composition

de services IoT, en utilisant le nouvel algorithme de chauve-souris (NBA) appelé QC-NBA. Cet algorithme vise à optimiser la fonction d'utilité basée sur SAW prenant en compte les préférences de l'utilisateur en terme de QoS. Dans l'article de Asghari et al. [74], les auteurs ont combiné un algorithme génétique avec un algorithme SFLA pour gérer la composition de services IoT tout en préservant la confidentialité des utilisateurs. SFLA est utilisé pour générer une population initiale diversifiée et explorer de nouvelles solutions, et GA pour affiner ces solutions. L'algorithme prend en compte les contraintes de confidentialité en intégrant des mécanismes de protection des données sensibles. Khansari et al. [75] se sont concentrés sur la sélection et la composition de services orientées QoS dans des applications IoT basées sur le cloud. Les paramètres de QoS pris en compte dans cette étude incluent la latence, la disponibilité, la fiabilité et le prix. Pour optimiser ces paramètres, les auteurs ont utilisé un algorithme génétique inspiré par la mécanique quantique, en y ajoutant des opérateurs de croisement et de mutation.

Certains travaux ont considéré les fluctuations des QoS lors de la résolution du problème de composition de services IoT. Dans [54], Khadir et al. ont proposé une approche pour la sélection de services IoT basée sur un algorithme évolutif multi-objectifs (multi-objective evolutionary algorithm MOEA). Cette méthode propose une approche distribuée quasi-optimale qui décompose les contraintes globales de QoS en contraintes locales distinctes. En prenant en compte les fluctuations de la QoS, les auteurs améliorent la fiabilité de la composition des services IoT. Cette approche permet de mieux gérer les variations de performance des services IoT, assurant ainsi une composition plus robuste et fiable.

D'autres travaux ont pris en compte la gestion de l'énergie lors de l'étude de la composition de services IoT axée sur la qualité de service. Ibrahim et al. [62] ont présenté une approche métaheuristique hybride combinant SFLA et GA pour minimiser la consommation d'énergie, le temps de réponse et le coût afin d'atteindre une QoS optimale dans un environnement cloud mobile. Zhou et al. [76] ont proposé une approche pour la découverte et la sélection de services en tenant compte de l'efficacité énergétique dans les réseaux de capteurs sans fil des dispositifs IoT. Ils ont abordé le problème de la composition de services comme un problème d'optimisation multi-objectifs et multi-contraintes. Pour cela, ils ont proposé deux approches : la première basée sur PSO et la seconde sur GA. Dans l'article [20], Wang et al. se sont concentrés sur la découverte et la sélection de services IoT dans un environnement cloud. Ils ont proposé un algorithme métaheuristique hybride multi-objectifs basé sur l'algorithme GWO et GA. GWO est utilisé pour explorer l'espace de recherche afin de trouver des solutions potentielles. Ensuite, GA affine ces solutions pour évoluer vers de meilleures compositions.

Baek et Ko [77] traitent le problème de la sélection dynamique de services IoT. Ils proposent une nouvelle métrique d'efficacité améliorée. Cette dernière prend en compte les préférences utilisateur et les spécificités des applications pour évaluer la qualité de la restitution des effets

sensoriels (visuels et auditifs) par les dispositifs IoT. Sur la base de cette métrique, ils introduisent DEOSA (Dynamic Effect-Oriented Service Selection Algorithm), un agent intelligent capable d'apprendre et de s'adapter pour sélectionner dynamiquement les services IoT. Hamzei et al. [78] proposent une approche de composition de services IoT basée sur le cloud et le fog, introduisant un nouvel algorithme hybride flou. Cet algorithme combine l'optimisation par colonies de fourmis et l'optimisation par essaim d'abeilles artificielles, en considérant la consommation d'énergie et les préférences de QoS.

Alors que les métaheuristiques sont souvent mises en œuvre pour trouver des solutions proches de l'optimale, nous présentons dans ce qui suit certains travaux qui ont proposé des approches basées sur des métaheuristiques retournant des solutions optimales comme dans Pavan et al. [19], Kurdi et al. [79], Zebouchi et Aklouf [80], Hosseinzadeh et al. [81], Tang et al. [82] et Kashyap et al. [83].

Pavan et al. [19] ont proposé une approche hybride combinant un algorithme génétique et un arbre de décision. L'arbre de décision permet de réduire l'espace de recherche en éliminant les services IoT candidats qui ne répondent pas aux critères de QoS requis. Une fois l'espace de recherche réduit, l'algorithme génétique est appliqué pour identifier la composition optimale. L'algorithme de recherche du coucou (CSA) a été utilisé par Kurdi et al. [79] pour aborder la composition de services IoT orientée QoS dans un environnement multi-cloud. Les auteurs ont considéré le problème de composition comme un problème de prise de décision multicritères. Dans cette recherche, seul le temps d'exécution a été amélioré. Pour la même problématique, Zebouchi et Aklouf [80] présentent une nouvelle métaheuristique hybride multi-objectifs, nommée pRTMNSGA-III, qui allie les forces des algorithmes RNSGA-III et TMNSGA-III. pRTMNSGA-III génère des compositions Pareto-optimales, c'est-à-dire des solutions qui offrent le meilleur compromis possible entre différents objectifs, tout en éliminant celles qui sont défavorables aux préférences de l'utilisateur.

Hosseinzadeh et al. [81] ont traité le problème de composition de services IoT dans le cloud-edge. Les auteurs ont proposé un algorithme hybride, appelé ANN-PSO, combinant un réseau de neurones artificiels (ANN) et PSO. L'ANN permet une initialisation intelligente des compositions candidates et une prédiction initiale de la qualité de ses compositions. Cette étape permet de guider l'algorithme PSO vers des régions prometteuses de l'espace de recherche dès le début du processus d'optimisation, améliorant ainsi l'efficacité et la rapidité de la recherche de solutions optimales.

Kashyap et al. [83] ont proposé un algorithme qui repose sur l'utilisation de NSGA II, un algorithme génétique avancé qui permet de gérer plusieurs objectifs simultanément. L'algorithme de Kashyap et al. suit un processus en plusieurs étapes : il commence par générer une population initiale de solutions potentielles, puis évalue ces solutions selon les critères de QoS. Ensuite, il

utilise un tri non dominé pour sélectionner les solutions les plus prometteuses et applique des opérateurs génétiques pour créer de nouvelles solutions. Ce cycle se répète jusqu'à ce qu'un critère d'arrêt soit atteint. Tang et al. [82] ont développé une version améliorée de l'algorithme SFLA nommée ISFLA pour la recherche de solutions optimales. Leur approche intègre la théorie du chaos et la théorie de l'apprentissage inverse (reverse learning theory) pour définir la population initiale. Après avoir initialisé la population, l'ISFLA utilise la distance euclidienne pour la diviser en groupes. Avant de mener une recherche locale, une mutation gaussienne (Gaussian mutation) est appliquée à la meilleure solution de chaque groupe. Enfin, une méthode de mise à jour locale, combinant l'apprentissage mutuel du SFLA et la mutation croisée de GA, est utilisée pour faire évoluer la population.

Selon [79, 83], les approches basées sur des métaheuristiques qui retournent des compositions optimales ne supportent généralement pas les espaces de recherche de grande taille et deviennent moins performantes en termes de temps de réponse.

Nous présentons dans ce qui suit un tableau résumant les travaux traitant le problème de composition de services dans un environnement IoT étudiés. Dans cette table, nous nous sommes focalisés sur quatre points de comparaison, à savoir la métaheuristique utilisée, le type de la solution retournée (optimale ou proche de l'optimale), la prise en considération du critère énergie et enfin, la considération des contraintes globales de QoS en plus des préférences de l'utilisateur.

Approche	Métaheuristique	Optimalité de la solution	Énergie	Contraintes globales
Kurdi et al. 2018 [79]	CSA	Optimale	Non	Non
Khansari et al. 2018 [75]	GA	Proche de l'optimale	Non	Non
Zhou et al. 2018 [76]	PSO, GA	Proche de l'optimale	Oui	Oui
Ibrahim et al. 2020 [62]	SFLA + GA	Optimale	Oui	Non
Kashyap et al. 2020 [83]	MOEA	Optimale	Non	Non
Asghari et al. 2020 [74]	SFLA + GA	Proche de l'optimale	Non	Non
Hosseinzadeh et al. 2020 [81]	PSO + ANN	Optimale	Non	Non
Boucetti et al. 2022 [1]	NN + GA	Proche de l'optimale	Non	Oui
Khadir et al. 2022 [54]	MOEA	Proche de l'optimale	Non	Oui
Pavan et al. 2022 [19]	DT + GA	Optimale	Non	Oui
Wang et Lu 2022 [20]	GWO + GA	Proche de l'optimale	Oui	Non
Kouicem et al. 2022 [21]	NBA	Proche de l'optimale	Non	Non
Zebouchi et al. 2024 [80]	RNSGA-III + TMNSGA-III	Optimale	Non	Non
Tang et al. 2024 [82]	SFLA + GA + théorie du chaos + apprentissage inverse +mutation gaussienne	Optimale	Non	Non
Hamzei et al. 2023 [78]	ABC + ACO + logique floue	Proche de l'optimale	Oui	Non

Tableau 2.3: Tableau récapitulatif des approches basées sur des métaheuristiques pour la composition de services IoT

Discussion

Plusieurs travaux [20,21,62,74,75,78–83] présentés dans le tableau 2.3 ne tiennent pas compte des contraintes globales de qualité de service, en raison de la complexité additionnelle qu'elles impliquent. En revanche, les travaux présentés dans [1,19,54,76] intègrent les contraintes globales de QoS, mais présentent certaines limitations. Les auteurs de [84] rapportent que l'approche proposée dans [1] se caractérise par une diversité limitée de la population, ce qui peut réduire

significativement la qualité de la solution retournée. Dans [19], les auteurs proposent de réduire l'espace de recherche avant de rechercher une composition IoT optimale, ce qui peut entraîner la perte de certaines solutions optimales. En effet, le processus de réduction peut éliminer des services pouvant faire partie de la solution optimale. De plus, l'approche proposée dans [54] décompose les contraintes globales de QoS en contraintes locales, alors que certains attributs QoS, tels que la fiabilité et la disponibilité, sont difficilement décomposables, et cette opération peut conduire à une perte d'information. Enfin, les résultats expérimentaux présentés par Zhou et al. [76] montrent que le processus d'itération de l'algorithme génétique utilisé est relativement coûteux en temps ; par ailleurs, [85] indique que le temps de réponse et la latence sont élevés, et [7] souligne que l'approche de Zhou et al. présente un niveau de complexité élevé.

2.5 Conclusion

Ce chapitre a posé les bases de notre étude en explorant le processus de composition de services IoT et en définissant les concepts clés qui y sont associés. Nous avons introduit la notion de service IoT, distingué les services abstraits des services candidats, et défini les compositions abstraites et concrètes. Nous sommes passés par la suite à la présentation des critères de qualité de service et nous avons détaillé les étapes de calcul du vecteur QoS d'une composition. L'état de l'art a été examiné en deux parties. Dans un premier temps, nous avons analysé les approches de composition de services basées sur des techniques déterministes, soulignant leurs avantages et leurs limitations. Ensuite, nous avons étudié les travaux proposant des solutions basées sur des métaheuristiques, qui offrent une alternative prometteuse pour la recherche de solutions optimales ou proches de l'optimale dans de larges espaces de recherche. Nous avons clôturé cette section par un tableau comparatif des approches traitant la sélection de services IoT suivi d'une discussion qui précise leur faiblesse. Dans la suite de cette thèse, nous allons présenter nos travaux qui incluent notre première contribution, DT-WOA, et la seconde nommée DALOA.

Chapitre 3

Nos contributions pour la sélection de services IoT sensible aux QoS

3.1 Introduction

Les approches basées sur les métaheuristiques se sont révélées efficaces pour la résolution des problèmes d'optimisation dans différents domaines d'application. En effet, on observe ces dernières années la proposition de différentes approches pour la résolution du problème de sélection de services sensible aux QoS. Parmi les approches existantes dans le domaine de l'IoT, nous citons l'approche de Boucetti et al. (2022) [1] qui combine GA avec NN, Wang et Lu (2022) [20] qui combinent l'algorithme GWO avec GA, et Zebouchi et Aklouf (2024) [80] qui ont combiné RNSGA-III et TMNSGA-III. Dans ce chapitre, nous allons présenter nos contributions intitulées Decision Tree with Whale Optimization Algorithm (DT-WOA) [22] et Discrete Adaptive Lion Optimization Algorithm (DALOA) [12].

3.2 Motivation

Face à l'explosion du nombre de services IoT fonctionnellement équivalents, il convient de développer des méthodes efficaces permettant d'identifier rapidement des combinaisons de services adaptées aux besoins des utilisateurs, tout en assurant un bon équilibre entre l'exploration et l'exploitation de l'espace de recherche. Dans cette perspective, nous avons proposé deux approches basées sur les métaheuristiques WOA et LOA.

Tout d'abord, nous avons défini DT-WOA, une approche qui associe les arbres de décision, reconnus pour leur capacité à classifier et structurer les informations [86], et l'algorithme WOA, réputé pour son efficacité à explorer efficacement l'espace de recherche [41]. Parmi les techniques d'apprentissage automatique existantes, telles que K-means, les arbres de décision, les forêts

aléatoires ou les réseaux bayésiens, notre choix s'est porté sur les arbres de décision pour la classification des services en fonction de leurs valeurs de QoS pour diverses raisons. En effet, les arbres de décision sont connus pour être simples à mettre en œuvre, rapides à exécuter et efficaces pour extraire des règles de décision claires à partir de données structurées. De plus, leur capacité à traiter des données hétérogènes sans prétraitement complexe les rend particulièrement adaptés à notre contexte de travail. Cet ensemble d'avantages est mis en évidence dans les travaux de [86,87], qui soulignent la large adoption des arbres de décision pour les tâches de classification supervisée. Dans le cadre de DT-WOA, les arbres de décision sont utilisés pour réduire l'espace de recherche en éliminant les services candidats les moins pertinents en terme de QoS. Cela permet de privilégier les combinaisons de services les plus prometteuses et ainsi de réduire considérablement le nombre de compositions candidates à explorer par WOA.

Sachant que les préférences de QoS ne suffisent pas pour satisfaire pleinement l'utilisateur, nous avons proposé une deuxième approche DALOA basée sur une adaptation de l'algorithme d'optimisation des lions qui prend en charge les préférences et les contraintes globales de QoS. DALOA combine trois opérateurs : le roaming, le mating et la migration. L'objectif de cette combinaison est de permettre à DALOA d'atteindre un bon équilibre entre l'exploration et l'exploitation de l'espace de recherche et ainsi d'offrir un bon rapport entre la qualité de la solution retournée et le temps de réponse de DALOA.

3.3 Contribution 1 : Decision Tree with Whale Optimization Algorithm (DT-WOA)

Cette section est dédiée à la présentation de DT-WOA. Afin de comprendre le fonctionnement de DT-WOA, nous commençons cette section par un bref rappel sur les arbres de décisions.

3.3.1 Aperçu sur les arbres de décision

Les arbres de décision, ou Decision Tree en anglais, constituent une méthode d'apprentissage automatique largement employée pour la modélisation de problèmes de classification et de régression. Ils offrent une représentation graphique intuitive des processus décisionnels, facilitant ainsi leur interprétation.

3.3.1.1 Définition

Un arbre de décision est un algorithme d'apprentissage qui imite le processus de décision humaine. Il se structure en une hiérarchie de nœuds. Le nœud de départ représente la racine de l'arbre, les nœuds internes correspondent à des tests sur les attributs des données, tandis que les feuilles représentent les classes ou les valeurs prédites [87, 88]. La construction d'un arbre

de décision commence par la sélection de l'attribut le plus pertinent qui divise le mieux les données. À partir de ce point, un nœud racine est créé, représentant cet attribut. Ensuite, les données sont réparties en sous-ensembles basés sur les valeurs de l'attribut. Ce processus se répète de manière récursive pour chaque sous-ensemble, choisissant à chaque étape l'attribut le plus pertinent jusqu'à ce que toutes les données soient correctement classées ou qu'un critère d'arrêt soit atteint, comme une profondeur maximale de l'arbre ou une quantité minimale de données par nœud. Les feuilles de l'arbre représentent les décisions finales ou les classifications obtenues à partir des règles de décision établies le long des branches. Cette méthode permet de créer une structure hiérarchique simple et facilement interprétable, illustrant le cheminement des décisions.

De façon plus simple, la construction d'un arbre de décision implique une partition récursive de l'espace des données, guidée par un critère d'impureté, jusqu'à ce qu'un critère d'arrêt soit atteint. Cette approche permet d'extraire des règles décisionnelles claires et compréhensibles, même pour des non-spécialistes.

Les arbres de décision présentent l'avantage de pouvoir traiter des données hétérogènes et de capturer des interactions non linéaires entre les variables. Ils sont utilisés pour la classification de données (par exemple, pour déterminer si un e-mail est un spam ou non) et pour la régression (par exemple, pour prédire le prix d'une maison en fonction de ses caractéristiques). En résumé, un arbre de décision est un modèle d'apprentissage supervisé qui permet de représenter graphiquement un processus de décision complexe en le décomposant en une série de décisions simples.

Rappelons que les arbres de décision sont intégrés dans notre approche pour réduire l'espace de recherche dans le but d'améliorer la qualité des solutions proposées par DT-WOA en se basant sur l'exploration des zones les plus pertinentes par rapport aux critères de l'utilisateur.

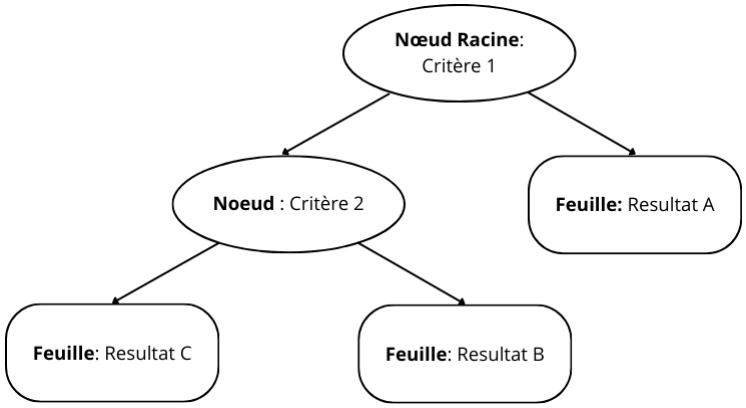

Figure 3.1: Exemple de structure d'un arbre de décision [88]

L'arbre de décision illustré dans la figure 3.1 met en évidence la manière dont un modèle peut hiérarchiser les critères de décision dans le but de classer un ensemble de données. En effet, l'attribut A constitue un premier critère de séparation, permettant d'identifier une sous-population homogène, dans le cas où les exemples ont une valeur de A qui est égale à a_1. Cependant, pour les exemples ne répondant pas à ce premier critère, il est nécessaire d'utiliser l'attribut B comme critère secondaire de classification, et cela permet d'affiner la partition lorsque l'attribut A ne suffit pas [88].

3.3.1.2 Les phases de construction d'un arbre de décision

La construction d'un arbre de décision dédié à la résolution des problèmes de classification se fait en deux étapes principales :

- **Phase d'apprentissage**
  Un modèle d'arbre de décision est induit à partir d'un ensemble d'apprentissage. Cette phase consiste en une partition récursive de l'espace des données, guidée par un critère d'impureté. À chaque nœud interne, un attribut est sélectionné en fonction de sa capacité à maximiser l'homogénéité des classes dans les sous-arbres. À la fin de cette étape, un modèle d'arbre de décision est obtenu avec tous les attributs et les règles de décision nécessaires pour la classification d'un nouvel ensemble de données non étiqueté.

- **Phase de classification**
  Un ensemble de données est classé en parcourant l'arbre de décision construit dans la première phase, depuis la racine jusqu'à une feuille. La classe associée à la feuille atteinte correspond à la classe prédite pour l'instance.

Nous avons choisi de combiner DT avec l'algorithme WOA en raison de sa capacité à gérer efficacement les problèmes d'optimisation complexes, tout en offrant la flexibilité nécessaire pour intégrer des techniques d'apprentissage automatique telles que des arbres de décision [41, 73]. Cette combinaison nous permet de tirer parti des avantages de WOA, tels que sa capacité à explorer efficacement l'espace de recherche, tout en focalisant la recherche de solution sur les régions les plus prometteuses de l'espace de recherche.

Après un rappel sur les arbres de décision, nous passons maintenant à la présentation de l'algorithme d'optimisation des baleines original.

3.3.2 L'algorithme WOA original

L'algorithme d'optimisation par essaim de baleines est une métaheuristique bio-inspirée développée par Mirjalili et Lewis en 2016 [3]. Cet algorithme imite le comportement collaboratif des baleines à bosse qui chassent les bancs de krill ou de petits poissons près de la surface en utilisant une technique appelée "the bubble-net feeding mechanism" (attaque au filet à bulles).

Cette technique consiste à émettre des bulles en spirale autour de la proie, comme illustré dans la figure 3.2.

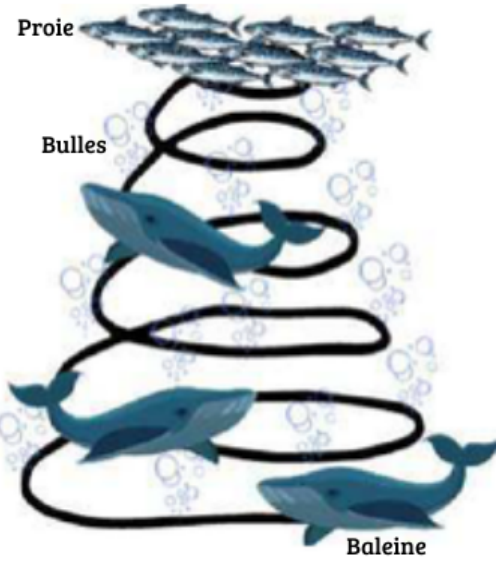

Figure 3.2: Technique de chasse en filet à bulles des baleines à bosse

Avant de présenter le modèle mathématique de WOA, il est pertinent d'expliquer brièvement le comportement de chasse des baleines à bosse. Ces mammifères marins sont parmi les plus grands au monde, qui peuvent atteindre 30 mètres de long et peser jusqu'à 190 tonnes à l'âge adulte. Lors de la recherche de nourriture, les baleines créent des bulles de différentes tailles le long d'un cercle ou d'une trajectoire en forme de "9". Deux techniques associées aux bulles sont utilisées par les baleines, à savoir : a) les spirales ascendantes et b) les doubles boucles. Les auteurs de WOA se sont concentrés sur la première technique, où les baleines à bosse remontent en spirale à environ 15 mètres de profondeur tout en émettant des bulles de tailles variées. Ces bulles remontent simultanément à la surface, formant un filet de bulles entourant la proie et la dirigeant vers le centre du cercle de bulles. Les baleines ouvrent alors la bouche pour capturer la proie.

Selon la description du comportement naturel des baleines, le comportement de chasse modélisé dans WOA peut être représenté par trois opérateurs : (1) la recherche de la proie, (2) l'encerclement de la proie, et (3) la manœuvre d'alimentation par filet à bulles.

Dans l'algorithme d'optimisation WOA, chaque baleine virtuelle (ou individu) représente une solution potentielle au problème d'optimisation, agissant en tant qu'agent de recherche. La proie, quant à elle, représente une solution optimale ou la meilleure solution connue à un moment précis. La population de baleines évolue itérativement en s'inspirant des trois opérateurs de WOA afin de se rapprocher le plus possible de la proie.

3.3.2.1 Modèle mathématique de l'algorithme WOA

Le modèle mathématique de l'algorithme WOA repose sur une alternance entre deux phases complémentaires : une phase d'exploration, où les individus visitent de larges zones de l'espace de recherche, et une phase d'exploitation, où ils se concentrent autour des meilleures solutions trouvées. Avant de détailler les deux phases de WOA, commençons par présenter ses paramètres $\vec{A}$ et $\vec{C}$.

$$\vec{A} = 2a \cdot \vec{r} - \vec{a} \tag{3.1}$$

$$\vec{C} = 2 \cdot r \tag{3.2}$$

où $\vec{a}$ diminue linéairement de 2 à 0 au cours des itérations, et $\vec{r}$ est un vecteur de valeurs aléatoires entre 0 et 1.

Phase d'exploitation : Technique d'attaque au filet à bulles

Le comportement de chasse au filet à bulles des baleines est modélisé mathématiquement par deux opérateurs : le mécanisme d'encerclement par rétrécissement et le mécanisme d'attaque.

a) **Mécanisme d'encerclement par rétrécissement**

Dans cet opérateur, les baleines ajustent leur position en fonction de celle de la meilleure solution trouvée. Cela permet aux baleines de se rapprocher progressivement de cette solution, comme l'illustre la figure 3.3.

Cet opérateur est modélisé mathématiquement par les équations 3.3 et 3.4, qui simulent le rétrécissement progressif du cercle de chasse [3].

$$\vec{D} = \left| C \cdot \vec{X}^*(t) - \vec{X}(t) \right| \tag{3.3}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - A \cdot \vec{D} \tag{3.4}$$

où $\vec{D}$ indique la distance entre la solution actuelle et la meilleure solution, $\vec{X}(t)$ représente la position d'une solution à un instant donné, $\vec{X}^*(t)$ désigne la position de la meilleure solution trouvée jusqu'à l'instant t.

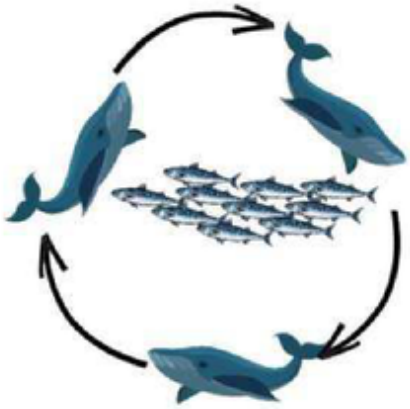

Figure 3.3: Encerclement par rétrécissement de la proie (meilleure solution)

L'opérateur d'encerclement par rétrécissement repose sur la diminution progressive de la valeur de a dans l'équation 3.1 au fil des itérations, ce qui entraîne la réduction du paramètre A. En conséquence, la nouvelle solution sera située entre sa position initiale et celle de la meilleure solution. La figure 3.4 illustre un exemple de cet opérateur pour un problème en deux dimensions où (X, Y) représente la position d'une solution de la population et (X^*, Y^*) la position de la meilleure solution actuelle [3].

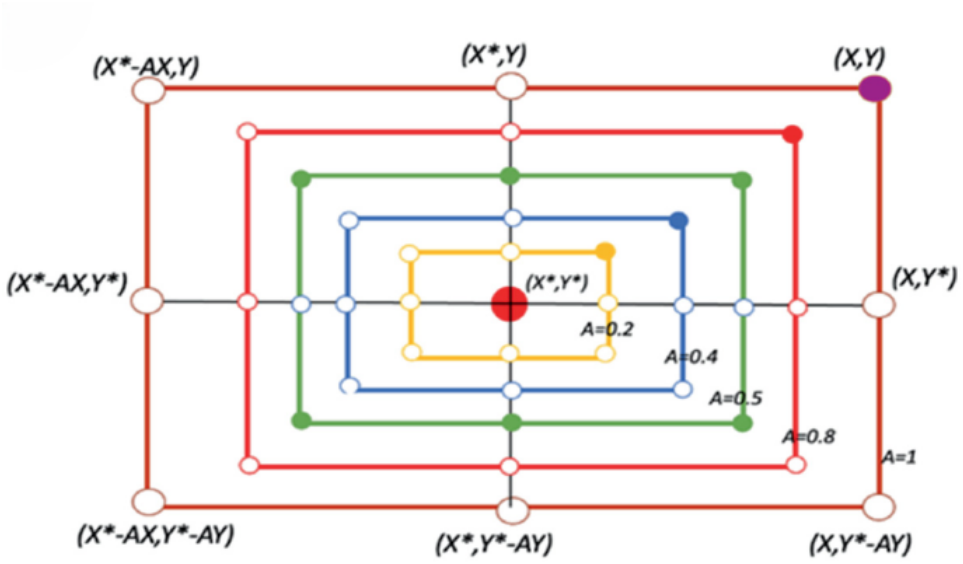

Figure 3.4: Principe de fonctionnement de l'opérateur d'encercelement [3].

b) **Mise à jour de la position en spirale**

Pour imiter le comportement de chasse des baleines à bosse, l'algorithme WOA introduit un mécanisme de déplacement en spirale. Chaque baleine suit la baleine ayant la meilleure position selon une trajectoire en forme d'hélice, comme illustré sur la figure 3.5. Cela

permet aux baleines de se rapprocher de la meilleure solution (X^*, Y^*). L'équation 3.5 décrit mathématiquement ce mouvement.

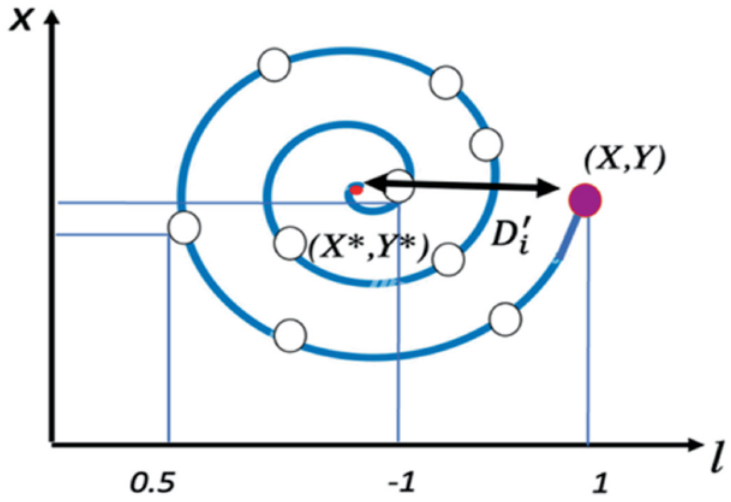

Figure 3.5: Mise à jour en spirale de la position de la baleine [3].

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) \tag{3.5}$$

où l est un nombre aléatoire dans l'intervalle $[-1, 1]$, b est une constante qui définit la forme de la spirale logarithmique, et $\vec{D'}$ est la distance entre la solution actuelle et la meilleure solution à cet instant.

La phase d'exploitation, qui consiste à se déplacer suivant une trajectoire en spirale autour de la meilleure solution connue à un instant, t permet à l'algorithme WOA d'explorer efficacement l'espace de recherche local autour de la meilleure solution actuelle.

Phase d'exploration : Recherche aléatoire de la proie

Tout comme les baleines réelles alternent entre des phases de chasse ciblée et des phases d'exploration plus aléatoire, l'algorithme WOA combine un mécanisme d'encerclement, simulant la poursuite d'une proie connue, et une phase d'exploration libre permettant aux baleines virtuelles de découvrir de nouvelles zones de nourriture. Ce mécanisme permet à WOA d'assurer une exploration plus large de l'espace de recherche et d'éviter de se retrouver piégé dans des optimums locaux [3, 41].

Contrairement au mécanisme d'encerclement des proies qui met à jour la position actuelle de la baleine en fonction de la meilleure baleine, lors de la recherche aléatoire, la mise à jour d'une position est effectuée en fonction d'une baleine sélectionnée aléatoirement à travers l'espace de recherche et déterminée par le vecteur A. Cet opérateur est modélisé mathématiquement par les équations 3.6 et 3.7.

$$\vec{D} = |C \cdot X_{rand}(t) - X(t)| \tag{3.6}$$

$$X(t+1) = X_{rand}(t) - A \cdot D \tag{3.7}$$

En combinant ces deux phases, l'algorithme WOA assure un bon équilibre entre la phase d'exploitation et la phase d'exploration [41, 89].

3.3.2.2 Algorithme WOA original

Pour balancer entre la phase d'exploration et celle d'exploitation, les auteurs ont utilisé un paramètre noté p, un nombre aléatoire entre 0 et 1. Le principe de l'algorithme WOA est résumé dans l'algorithme 1 [3].

Algorithm 1 L'algorithme WOA original

Initialiser la population des baleines X_i $(i = 1, 2, \ldots, n)$
Évaluer chaque baleine.
X^* = Position de la meilleure baleine.
Tant que $t <$ **le nombre max d'itération**
  Pour chaque baleine, faire :
    Mettre à jour a, A, C, l, et p
    Si $p < 0.5$ **alors**
      Si $|A| < 1$
        Encercelement par retrécissement
          Mettre à jour la position de la baleine actuelle par l'équation 3.3
      Sinon
        Recherche aléatoire
          Sélectionner une baleine aléatoirement (X_{rand})
          Mettre à jour la position de la baleine actuelle par l'équation 3.7
      FinSi
    Sinon
      Mise à jour en spirale
        Mettre à jour la position de la baleine par l'équation 3.5
    FinSi
  FinPour
    Mettre à jour X^*.
    $t = t + 1$
FinTanque
Retourner X^*

Dans la suite de ce chapitre, nous allons passer à la présentation de l'approche DT-WOA.

3.3.3 Approche proposée : Decision Tree with Whale Optimization Algorithm

Le principe de la composition de services consiste à assembler deux ou plusieurs services en réponse à une demande utilisateur complexe, tout en respectant ces exigences en termes de valeur de QoS. L'algorithme DT-WOA est proposé dans ce chapitre pour traiter le problème de composition de services IoT avec prise en compte des préférences QoS de l'utilisateur. DT-WOA est une combinaison d'un arbre de décision avec une version discrète de l'algorithme WOA. En effet, WOA est une métaheuristique proposée pour les problèmes d'optimisation à variables continues. Pour l'adapter au problème de composition, nous avons utilisé la fonction round lors de l'application des opérateurs de WOA afin de garantir que les résultats seront toujours des variables discrètes.

3.3.3.1 Fonctionnement de l'approche proposée DT-WOA

DT-WOA se déroule en deux étapes principales, à savoir la classification de services et la sélection de services. Rappelons qu'une composition abstraite notée $Comp_{abstract} = \{\{AS_1, AS_2, ...AS_n\}$ est un enchainement de n services abstraits. Chaque AS contient m services candidats fonctionnellement équivalents, une composition concrète est un vecteur d'entiers noté, $comp = \{x_1, x_2.., x_n\}$ et enfin, chaque composition est décrite par son vecteur de QoS. La table 3.1 fournit la description des paramètres utilisés lors de la modélisation du problème de composition.

Paramètre	Description
n	Nombre de AS dans une composition.
m	Nombre de CS dans chaque AS.
r	Nombre de QoS pris en compte.
$nPop$	Taille de la population.
$Pref_k$	Poids attribué à l'attribut k.

Tableau 3.1: Description des paramètres de modélisation du problème de composition

1. Classification et sélection des services candidats via l'arbre de décision

L'objectif de la classification est d'identifier la catégorie à laquelle appartient une nouvelle donnée en fonction de ses paramètres. Pour cela, nous avons développé un modèle d'arbre de décision pour évaluer chaque service selon son vecteur de QoS et l'affecter à une catégorie. Pour réduire l'espace de recherche, DT-WOA classe d'abord les services IoT, puis sélectionne ceux qui présentent les meilleures évaluations globales en termes de QoS. Pour la mise en œuvre du modèle d'arbre de décision de DT-WOA, les deux datasets QWS 1.0 et QWS 2.0 [4, 90] ont été utilisés.

QWS 1.0 est un dataset étiqueté utilisé pour l'entrainement supervisé du modèle de l'arbre de décision. Ce dataset contient des vecteurs QoS de 365 services candidats réels caractérisés par neuf attributs de QoS et classés en quatre catégories, à savoir bronze, argent, or et platine. Ensuite, le modèle DT obtenu est appliqué sur le QWS 2.0 pour classer les services dans le but de réduire l'espace de recherche en supprimant les services les moins pertinents par rapport aux besoins de l'utilisateur. Enfin, l'algorithme WOA discrétisé est appelé pour retrouver une solution satisfaisant les besoins de QoS sur le nouvel espace de recherche. L'algorithme 2 représente les étapes du modèle d'arbre de décision utilisé pour la réduction de l'espace de recherche.

Algorithm 2 Algorithme du modèle d'arbre de décision de DT-WOA

Entrées : QWS 1.0, QWS 2.0
Sorties : QWS 2.0 réduit.
DEBUT

1: Lire le dataset QWS 1.0 ;
2: Faire appel à la fonction DecisionTreeClassifier() pour l'entrainement supervisé du modèle ;
3: Lire QWS 2.0 ;
4: Utiliser le modèle de l'arbre de décision pour classer les services du QWS 2.0 ; en quatre catégories : "platine", "or", "argent" et "bronze".
5: Sélectionner les services qui font partie des deux meilleures catégories, à savoir les catégories "platine" et "or".

FIN

Dans l'algorithme 2, la fonction DecisionTreeClassifier() issue de la bibliothèque Scikit-learn, a été utilisée pour l'entraînement du modèle DT sur le dataset QWS 1.0 caractérisé par neuf attributs de QoS. Cette fonction permet la construction d'une structure arborescente décisionnelle, où chaque nœud interne correspond à une condition de test sur un critère QoS spécifique, tandis que les feuilles représentent les décisions finales de classification. L'algorithme procède à une division récursive des services en se basant sur les attributs QoS les plus significatifs, afin de former des groupes aussi homogènes que possible.

2. Recherche d'une composition proche de l'optimale avec WOA discrétisé

Dans le contexte de la composition de services sensibles aux QoS, une composition correspond à une baleine et la proie représente la composition optimale ou une composition proche de celle-ci. Dans cette section, nous allons détailler les étapes du WOA discrétisé.

a. **Générer aléatoirement les compositions de la population initiale**

À cette étape, on génère aléatoirement des compositions à travers l'espace de recherche. Les valeurs générées ne doivent pas dépasser le nombre de services candidats dans les AS

(c'est-à-dire que les x_i de chaque composition ont des valeurs entre 1 et m).

b. **Évaluation de la population initiale**

Après avoir généré la population initiale, nous procédons au calcul de la fitness de chaque composition en utilisant la fonction d'utilité présentée dans l'équation 2.8. Ensuite, la composition ayant la meilleure valeur de fitness est enregistrée et sauvegardée. Dans notre cas, nous cherchons à minimiser la valeur de la fitness, donc c'est la composition ayant la plus petite valeur de fitness qui sera retenue.

c. **Appliquer les opérateurs de WOA discrétisé à la population initiale**

Pour chaque composition de la population, on applique les opérateurs de WOA discrétisée pour mettre à jour les compositions. Deux opérations sont définies dans WOA, l'encerclement et l'attaque. Notez qu'à la fin de chaque opération, une nouvelle composition est générée et ira remplacer l'ancienne.

c.1. **L'encerclement**

L'algorithme WOA commence par identifier et sauvegarder la meilleure composition de la population en calculant et comparant les valeurs de fitness des compositions. Étant donné que la composition optimale n'est pas connue au départ, la valeur de fitness de la meilleure composition sera utilisée pour mettre à jour les autres compositions [89], et cela selon l'équation 3.8.

$$\vec{comp}(t+1) = \vec{comp}^*(t) - \vec{A} \cdot |\vec{C} \cdot \vec{comp}^*(t) - \vec{comp}(t)| \tag{3.8}$$

Où $\vec{comp}^*$ représente la meilleure composition obtenue jusqu'à présent, $\vec{comp}$ représente la composition actuelle, $\vec{comp}(t+1)$ représente la nouvelle composition, t représente l'itération courante, $\vec{A}$ et $\vec{C}$ sont des vecteurs de coefficients qui peuvent être calculés comme suit :

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{3.9}$$

$$\vec{C} = \vec{a} \cdot \vec{r} \tag{3.10}$$

Où $\vec{a}$ décroît linéairement de 2 à 0 au cours de la recherche, et $\vec{r}$ est un vecteur aléatoire qui suit une distribution uniforme dans l'intervalle [0, 1]. Ainsi, l'équation 3.8 permet à chaque composition de mettre à jour sa position dans le voisinage de la meilleure solution actuelle. L'encerclement par rétrécissement est mis en œuvre en réduisant progressivement la valeur de $\vec{a}$ dans les équations 3.9 et 3.10 au fil des itérations. En limitant la valeur de $\vec{A}$ à l'intervalle [-1, 1], la nouvelle composition peut être définie n'importe où entre sa position actuelle et la position de la composition ayant la meilleure fitness.

c.2. **L'attaque au filet à bulles (Phase d'exploitation).**

Le processus de mise à jour en spirale repose sur l'utilisation d'équations permettant d'imiter le mouvement en hélice des baleines à bosse [89], comme suit :

$$\vec{comp}(t+1) = \vec{D'} \cdot e^{bl} \cdot cos(2\pi l) + \vec{comp}^*(t) \tag{3.11}$$

Où $\vec{D'} = |\vec{comp}^*(t) - \vec{comp}(t)|$ définit la distance entre la i^{ime} composition et la solution actuelle, b est une constante définissant une spirale logarithmique, l est un vecteur aléatoire défini selon une distribution uniforme dans l'intervalle $[-1, 1]$, et $\vec{comp}^*$ représente la meilleure composition obtenue à l'itération t.

Puisque les baleines à bosse nagent autour de la proie dans un cercle rétréci tout en suivant un chemin en forme de spirale, l'algorithme WOA suppose qu'il existe une probabilité de 0,5 de choisir l'un ou l'autre de ces processus à chaque itération, comme le montre la formule suivante :

$$\vec{comp}(t+1) = \begin{cases} \vec{comp}^*(t) - \vec{A} \cdot |\vec{C} \cdot \vec{comp}^*(t) - \vec{comp}(t)|, & p < 0.5 \\ \vec{D'} \cdot e^{bl} \cdot cos(2\pi l) + \vec{comp}^*(t), & p \geq 0.5 \end{cases} \tag{3.12}$$

Où p est un nombre aléatoire compris entre 0 et 1.

c.3. **Recherche de solutions (Phase d'exploration)**

Contrairement à la phase d'exploitation, WOA met à jour la population de composition en se basant sur une composition choisie aléatoirement à travers l'espace de recherche lors de l'exploration comme suit [89] :

$$\vec{comp}(t+1) = \vec{comp}_{rand} - \vec{A} \cdot |\vec{C} \cdot \vec{comp}_{rand} - \vec{comp}(t)| \tag{3.13}$$

où $\vec{comp}_{rand}$ représente une composition choisie au hasard parmi l'ensemble actuel des compositions. La valeur du coefficient A est utilisée pour décider si une composition doit effectuer une recherche aléatoire. Lorsque $|A| \geq 1$, cela signifie que la composition doit faire de l'exploration.

d. **Choisir la meilleure solution de la population**

À la fin de DT-WOA, l'algorithme retourne la meilleure composition parmi toutes les compositions générées pendant les itérations. Cela est possible car, à chaque fin d'itération, la meilleure composition retrouvée est sauvegardée, permettant ainsi une comparaison des valeurs de fitness. En suivant ces étapes, l'algorithme WOA explore l'espace de recherche en

générant et en évaluant de nouvelles compositions à chaque itération. Au fil des générations, il tend à converger vers une bonne composition.

3.3.3.2 Algorithme WOA pour la composition de services

Les étapes de WOA décrites dans la section précédente sont résumées dans l'algorithme 3.

Algorithm 3 Algorithme WOA discrétisé

Entrées : n, r, m, nPop, $Pref_k$,
Sortie : meilleure composition retrouvée,
DEBUT

1: Générer aléatoirement la population initiale ;
2: Évaluer de la population initiale en utilisant l'équation 2.8 ;
3: Sauvegarder la meilleur fitness et sa composition ;
4: **Tant que t< nbtIteration**
5: **Pour chaque composition, faire :**
6: Mises à jour de a, A, C, I, et p ;
7: **Si (p<0,5)**
8: **Si (|A|<1)**
9: Mises à jour de la composition en utilisant l'équation 3.8 ;
10: **Sinon**
11: Mises à jour de la composition en utilisant l'équation 3.13 ;
12: **FinSi**
13: **Sinon**
14: Mises à jour de la composition en utilisant l'équation 3.11 ;
15: **FinSi**
16: **FinPour**
17: Après chaque mise à jour de la population, évaluer les nouvelles compositions avec l'équation 2.8 ;
18: Sauvegarder la meilleure valeur de fitness et sa composition ;
19: t=t+1 ;
20: **FinTanque**
21: Retourner la meilleure composition et sa fitness.

FIN

Nous passons dans ce qui suit à la présentation de notre seconde contribution DALOA.

3.4 Contribution 2 : Discrete Adaptive Lion Optimization Algorithm (DALOA)

Dans cette section, nous allons présenter une brève description de la métaheuristique bio-inspirée LOA sur laquelle se base notre approche. Ensuite, nous détaillons l'approche proposée, qui consiste en une adaptation de LOA au problème de composition de services avec prise en compte des préférences et des contraintes globales de QoS.

3.4.1 Pourquoi LOA ?

L'algorithme des lions a été utilisé pour la résolution d'une grande variété de problèmes et a démontré de bons résultats. Nous avons recensé quelques approches utilisant LOA, tel que [91] où Almezeini et Hafez appliquent LOA pour la résolution du problème de planification des tâches (ordonnancement) dans le cloud computing ; Geeta et al. [92] ont utilisé LOA pour la construction de code de livres (codebook) pour la compression d'images médicales ; dans [93] Gope et al. utilisent LOA pour la gestion de la congestion du transport dans un système électrique ; LOA a aussi été utilisé par Saranya et al. [94] pour l'amélioration de la prise de décision dans les réseaux coopératifs de santé dans le cloud-edge (healthcare cloud-edge networks). D'après nos recherches, aucun papier n'avait appliqué LOA au problème de sélection de services web ou IoT avec prise en compte des préférences de l'utilisateur et des contraintes globales au moment de notre réalisation du papier. Motivés par ces observations, nous avons développé DALOA [12].

3.4.2 Lion Optimizer Algorithm (LOA)

LOA est une métaheuristique bio-inspirée basée sur une population de solutions proposée par Yazdani et al. [23] pour traiter les problèmes d'optimisation complexes où l'espace de recherche augmente exponentiellement avec la taille du problème. Le mode de vie spécial des lions et leurs caractéristiques de coopération ont été la motivation de base pour le développement de cet algorithme d'optimisation. En effet, les lions sont une espèce de félin social qui fait preuve d'un niveau élevé de coopération entre les individus. En plus, les lions ont deux types d'organisation sociale : les résidents, aussi dits les prides, et les nomades. Les prides vivent en tribu et possèdent un territoire qu'ils défendent et qu'ils peuvent étendre, tandis que les nomades ne possèdent pas de territoire et sont généralement toujours en déplacement. Les résidents vivent en groupes. Un groupe de pride comprend généralement environ cinq femelles, leurs petits, et un ou plusieurs mâles adultes. Les jeunes mâles sont exclus de leur pride de naissance lorsqu'ils deviennent matures. Le deuxième comportement organisationnel est appelé nomade, car les animaux se déplacent de façon irrégulière, soit par paires, soit individuellement. Les pairs sont souvent des mâles apparentés exclus de leur pride maternelle. De plus, l'algorithme LOA définit divers opérateurs modélisant les comportements des lions tels que la chasse (hunting), le

déplacement vers un endroit sûr du territoire, la promenade (roaming), la reproduction (mating), la défense et la migration (migration).

3.4.3 Approche proposée : Discrete Adaptive Lion Optimization Algorithm

Afin de garantir une bonne balance entre l'exploration et l'exploitation de l'espace de recherche et d'éviter d'être piégé dans un optimum local, DALOA combine trois opérateurs de LOA, le roaming, le mating et la migration. En plus de cette combinaison d'opérateurs, la population dans DALOA est divisée en deux types d'individus, à savoir les compositions nomades (*CompNomad*) et les compositions prides (*CompPride*). Chaque type de composition adopte une stratégie de recherche spécifique. Ainsi, nous distinguons les algorithmes suivants : roaming pride, roaming nomade, mating pride et mating nomade. L'opérateur de migration est appliqué à la fin de chaque itération et permet à une composition nomade de devenir pride et vice versa. Afin d'adapter LOA au problème de composition de services, un opérateur de discrétisation a été intégré dans DALOA. La table 3.2 présente les paramètres de DALOA.

Paramètre	**Description**
$N\%$	Pourcentage de nomades
$S\%$	Pourcentage de femelles dans un pride
$R\%$	Pourcentage de participants à la promenade
$Ma\%$	Pourcentage de participantes au mating
$Mi\%$	Pourcentage de participants à la migration

Tableau 3.2: Description des paramètres de DALOA

3.4.3.1 Les étapes de DALOA

Nous détaillons dans cette partie les étapes de DALOA et leur fonctionnement.

Étape 1 : Initialisation de la population

Tout d'abord, une population initiale de $nPop$ compositions est générée aléatoirement à travers l'espace de recherche. Cet ensemble de compositions est ensuite divisé en deux sous-populations, imitant le mode de vie des lions dans la nature. Ensuite, $N\%$ des compositions de l'ensemble de la population sont choisies pour former la sous-population nomade, tandis que le reste des compositions constitue la sous-population des prides.

Comme mentionné précédemment, une série d'opérateurs est combinée dans DALOA. Nous allons donc présenter ci-dessous les trois opérateurs, accompagnés de leur algorithme et d'une explication détaillée de leur fonctionnement.

Étape 2 : Opérateur de roaming

Cet opérateur permet à DALOA d'explorer l'espace de recherche pour découvrir de meilleures compositions, en utilisant deux stratégies de recherche distinctes. En effet, selon le type de composition pris en compte, deux algorithmes peuvent être exécutés : le roaming pride ou le roaming nomade.

a. Application de l'opérateur roaming pride

Dans un premier temps, un échantillon aléatoire de $(100-S)\%$ de la sous-population pride est sélectionné aléatoirement. Ensuite, pour chacune de ces compositions, une nouvelle composition est générée et évaluée en utilisant l'équation (2.8). Enfin, la composition avec la valeur d'utilité la plus élevée est conservée et la population est mise à jour. Les différentes étapes de cet opérateur sont décrites en détail dans l'algorithme 4. Notons que la fonction $FNbr(R)$ est utilisée dans nos travaux pour convertir un pourcentage en un nombre entier.

Algorithm 4 Roaming Pride

Entrée : R, S, population de prides
Sortie : Nouvelle population de prides
DÉBUT

Pour $nbr = 1$ to $FNbr(100 - S)$ **faire**
  Sélectionner aléatoirement une composition pride ($CompPride_{nbr}$) ;
  Pour $cpt = 1$ to $FNbr(R)$ **faire**
    Générer une nouvelle composition aléatoirement ;
    Évaluer les nouvelles compositions en utilisant la fonction d'utilité (2.8) ;
    Sauvegarder la nouvelle meilleure composition $newComp_{cpt}$;
  FinPour
Si $U(CompPride_{nbr}) < U(newComp_{cpt})$ **alors**
  Mettre à jour $CompPride_{nbr}$;
FinSi
FinPour

FIN

Il est important de rappeler que les compositions prides n'explorent qu'une partie de l'espace de recherche. En conséquence, cet opérateur permet d'effectuer une recherche locale approfondie,

optimisant ainsi la phase d'exploitation (recherche dans le voisinage d'une composition) pour obtenir une composition de meilleure qualité.

b. Application de l'opérateur roaming nomade

Contrairement à une composition pride, une composition nomade suit une stratégie de recherche aléatoire à travers tout l'espace de recherche. Cet opérateur est représenté par l'équation 3.14 et ses étapes sont illustrées dans l'algorithme 5. Le rôle de cet opérateur est de garantir une meilleure diversification de la population et d'éviter à DALOA de se retrouver piégée dans un optimum local.

$$CompNomad'_{nbr} = \begin{cases} CompNomad_{nbr} & \text{Si RAND} > Pr_{nbr} \\ RAND_{nbr} & \text{Sinon} \end{cases} \tag{3.14}$$

tel que $RAND_{nbr}$ représente une composition générée aléatoirement, $CompNomad_{nbr}$ représente la composition nbr de la sous-population nomade, et Pr_{nbr} est une probabilité calculée pour chaque composition nomade suivant l'équation présentée dans (3.15). Enfin, $RAND$ est une fonction qui génère aléatoirement un nombre entre 0 et 1.

$$Pr_{nbr} = 0.1 + \min\left(0.5, \left(\frac{U(CompNomad_{nbr}) - U(BestNomad)}{U(BestNomad)}\right)\right) \tag{3.15}$$

tel que $U(BestNomad)$ représente la meilleure valeur d'utilité de la sous-population nomade, et $U(CompNomad_{nbr})$ représente la valeur d'utilité de la composition nbr.

Algorithm 5 Roaming Nomade

Entrée : population nomade
Sortie : nouvelle population nomade
DÉBUT
  Pour $nbr = 1$ à $FNbr(N)$ **faire**
    Calculer la probabilité Pr_{nbr} en utilisant (3.15) ;
    Si $Pr_{nbr} >$ RAND **alors**
      Mettre à jour $CompNomad$ aléatoirement ;
    FinSi
  FinPour
  Mettre à jour $BestNomad$;
FIN

Étape 3 : Opérateur de mating

L'opérateur de reproduction permet le partage d'informations entre les compositions au sein de la même sous-population. En effet, une composition pride peut se reproduire avec une ou plusieurs autres compositions pride, selon la valeur du paramètre Sc. Contrairement à cela, une composition nomade ne peut se reproduire qu'avec une seule autre composition nomade, ce qui conduit à deux algorithmes distincts pour cet opérateur : le mating pride et le mating nomade. Chaque opération de reproduction génère deux nouvelles compositions, appelées $NewComp_1$ (équation 3.16) et $NewComp_2$ (équation 3.17).

$$NewComp_1 = \beta_{nbr} CompMate_{nbr} \oplus \sum_{cpt=1}^{b} Sc_{cpt} \frac{1-\beta_{nbr}}{\sum_{cpt=1}^{b} Sc_{cpt}} Comp_{cpt} Sc_{cpt} \tag{3.16}$$

$$NewComp_2 = (1-\beta_{nbr}) CompMate_{nbr} \oplus \sum_{cpt=1}^{b} Sc_{cpt} \frac{\beta_{nbr}}{\sum_{cpt=1}^{b} Sc_{cpt}} Comp_{cpt} Sc_{cpt} \tag{3.17}$$

tel que b est le nombre de compositions dans une sous-population, β est un vecteur colonne de b réels générés aléatoirement avec une distribution normale, une valeur moyenne égale à 0,5 et un écart type égal à 0,1, Sc_{cpt} est égal à 1 si la composition $Comp_{cpt}$ est sélectionnée pour la reproduction et égal à 0 dans le cas contraire. $CompMate_{nbr}$ est la composition sélectionnée pour la reproduction.

Les étapes de l'opérateur mating pride sont présentées dans l'algorithme 6. Chez les nomades, les étapes sont les mêmes, sauf qu'une composition ne se reproduit qu'avec une seule autre composition nomade.

Algorithm 6 Mating Pride

Entrée : Ma, population pride ;
Sortie : Nouvelle population pride ;
DÉBUT

1: Initialiser le vecteur β ;
2: **Pour** $nbr = 1$ to $FNbr(Ma)$ **faire**
3: Sélectionner aléatoirement une composition pride ;
4: Appliquer les équations (3.16) et (3.17) ;
5: Sauvegarder les nouvelles compositions ;
6: **FinPour**
7: Evaluer l'ensemble des nouvelles compositions générées ;
8: Fusionner les nouvelles compositions avec la population des prides ;
9: Classer la nouvelle population en fonction des valeures d'utilité ;
10: Mise à jour de la population pride ;

FIN

Définition de l'opérateur de discrétisation $\oplus$ de DALOA

Dans les équations 3.16 et 3.17, une composition est multipliée par un nombre réel (élément du vecteur β), ce qui aboutit à une composition contenant des nombres réels. Rappelons qu'une composition est représentée sous la forme d'un vecteur d'entiers. Ces entiers correspondent aux indices des services IoT inclus dans la composition considérée. L'application d'opérations arithmétiques standards telles que l'addition et la multiplication, dans ces équations produit des valeurs réelles. Pour résoudre ce problème, nous définissons dans ce qui suit un opérateur de discrétisation noté $\oplus$ qui a pour rôle de convertir les valeurs réelles en valeurs entières. L'algorithme 7 présente le fonctionnement de l'opérateur $\oplus$, où $Integer(x)$ est une fonction qui reçoit le nombre réel x en entrée et renvoie la partie entière de x en sortie en respectant les bornes min/max précisées.

Afin d'augmenter la diversité de la population et d'explorer un grand nombre de compositions à travers l'espace de recherche, l'opérateur de multiplication standard est combiné à l'opérateur $\oplus$ pendant l'application des équations 3.16 et 3.17.

Algorithm 7 Opérateur de discrétisation $\oplus$

Entrée : Vec_1, Vec_2, $IndexMax$
Sortie : $NewComp$
DÉBUT

1: **Pour** id=1 to length (Vec_1) **faire**
2: $NewComp(id) = Integer(Vec_1(id) + Vec_2(id))$;
3: **Si** $NewComp(id) > IndexMax$ **alors**
4: $NewComp(id) = IndexMax$;
5: **Sinon**
6: **Si** $NewComp(id) < 1$ **alors**
7: $NewComp(id)$=1 ;
8: **FinSi**
9: **FinSi**
10: **FinPour**

FIN

Exemple d'application de l'opérateur $\oplus$

Afin de visualiser le fonctionnement de l'opérateur de discrétisation $\oplus$, considérons un exemple appliqué à une population nomade de six compositions. Dans cet exemple, l'équation 3.16 est utilisée avec un intervalle de valeurs autorisées de 1 à 10, correspondant au nombre de services IoT candidats pour chaque tâche du plan de composition.
Soit :

$\beta_{Nomad} = [0.1226\ 0.0284\ 0.0173\ 0.1431\ 0.1386\ 0.1581]$;

$\beta_{nbr} CompMate_{nbr} = [1.71\ 0.34\ 6.78\ 9.01] = Vec_1$ et

$\sum_{cpt=1}^{b} \frac{1-\beta_{nbr}}{\sum_{cpt=1}^{b} Sc_{cpt}} Comp_{cpt} Sc_{cpt} = [1.38\ 0.15\ 2.92\ 2.09] = Vec_2$,

Cas 1 : Utilisation des opérateurs arithmétiques standards
$Vec_1 + Vec_2 = [3.09\ 0.49\ 9.70\ 11.10]$

Cas 2 : Utilisation de l'opérateur $\oplus$
$Vec_1 \oplus Vec_2 = [3\ 1\ 9\ 10]$

Étape 4 : Opérateur de migration

Le rôle principal de cet opérateur est de permettre à une composition de changer de type, c'est-à-dire qu'une composition pride peut devenir nomade et vice-versa. Ainsi, l'opérateur de reproduction permet le partage et l'échange d'informations entre les deux sous-populations. En effet, lors de la migration, un nombre $FNbr(Mi)$ de compositions pride sont sélectionnées et définies par la suite comme des compositions nomades. Ensuite, la nouvelle population nomade est triée en fonction de la valeur de l'utilité, et les $FNbr(Mi)$ meilleures compositions migrent pour devenir pride. Cela permet de maintenir l'équilibre entre les deux sous-populations. Ce changement de type permet à DALOA de favoriser la diversité de la population en permettant l'échange d'informations entre les compositions prides et nomades. L'algorithme 8 résume les étapes de cet opérateur.

Algorithm 8 Migration

Entrée : Mi, population de prides, population de nomades ;
Sortie : Nouvelle population de prides et de nomades ;
DÉBUT

1: **Pour** $nbr = 1$ to $FNbr(Mi)$ **faire**
2: $CompPride_{nbr}$ migre et devient nomade ;
3: **FinPour**
4: Trier la nouvelle population de nomades en fonction de leur utilité ;
5: Les meilleures $FNbr(Mi)$ compositions nomades migrent et deviennent prides ;
6: Mettre à jour $BestNomad$;

FIN

3.4.3.2 L'algorithme DALOA

Nous clôturons cette section par la présentation des principales étapes de DALOA, résumées dans l'algorithme 9.

Algorithm 9 L'algorithme DALOA

Entrée : $NbrIteration$, n, m, r ;
Sortie : Composition proche de l'optimale
DÉBUT

Initialiser la population
For $NbrIt = 1$ to $NbrIteration$ **faire**
Opérateur de Roaming
    Appliquer Roaming Pride (Algorithme 4) ;
    Appliquer Roaming Nomad (Algorithme 5) ;
Opérateur de Mating
    Appliquer Mating Pride (Algorithme 6) ;
    Appliquer Mating Nomad ;
Opérateur de Migration
    Appliquer Migration (Algorithme 8) ;
Contraintes globales de QoS
    Vérifier la satisfaction des contraintes QoS globales ;
FinPour
Retourner la meilleure composition proche de l'optimale trouvée ;

FIN

3.5 Conclusion

Ce chapitre est partitionné en deux parties. Dans la première partie, nous avons présenté l'approche DT-WOA, un algorithme conçu pour optimiser la composition de services IoT en fonction des préférences QoS de l'utilisateur. La solution proposée utilise un modèle d'arbre de décision pour classer les services et cibler les zones de recherche les plus prometteuses, avant d'appliquer WOA pour identifier une composition satisfaisant l'utilisateur. La seconde partie de ce chapitre a été consacrée à la présentation de DALOA, une approche proposée pour la résolution du problème de composition de services IoT avec prise en compte des préférences et des contraintes globales de QoS. Pour adapter DALOA aux problèmes de sélection de services, l'opérateur $\oplus$ a été défini. De plus, DALOA combine trois opérateurs : le roaming, le mating et la migration afin d'assurer un bon compromis entre la qualité de la solution retournée et son temps de calcul. Le roaming et le mating adoptent une stratégie de recherche qui change selon le type de la composition. La migration, quant à elle, est appliquée dans DALOA pour permettre de changer de mode de vie, de sorte qu'une composition nomade puisse devenir pride et vice versa. Le chapitre suivant est dédié à l'analyse des performances des approches proposées. Il présente et examine les résultats obtenus, permettant ainsi d'évaluer leur efficacité dans le contexte étudié.

Chapitre 4

Étude des performances de DT-WOA et DALOA

4.1 Introduction

Dans ce chapitre, nous présentons les résultats des expériences menées pour évaluer les performances des approches proposées. Tout d'abord, nous commençons par l'évaluation de DT-WOA en comparaison avec l'algorithme WOA original. Ensuite, nous passons à l'étude de l'efficacité de DALOA en comparaison à des approches existantes.

4.2 Étude des performances de DT-WOA

Les tests ont été réalisés dans un environnement de simulation Python, en utilisant Jupyter Notebook sur un ordinateur portable équipé d'un processeur Intel® Core™ i3-6006U et de 4 Go de RAM. Le but de la comparaison de DT-WOA avec WOA est de démontrer l'impact de l'arbre de décision sur les performances de WOA.

4.2.1 Scénario du cas d'étude de DT-WOA

L'efficacité de l'algorithme DT-WOA a été démontrée à travers son application au problème de la composition de services IoT dans une maison intelligente, où l'objectif est de détecter toute tentative d'intrusion et de notifier le propriétaire en cas de danger réel. Pour atteindre cet objectif, il sera nécessaire de combiner les services IoT abstraits suivants : Service de détection de mouvement (Movement Detection Service – MDS), Service d'activation de caméra (Camera Activation Service – CAS), Service d'analyse vidéo en temps réel (Real-Time Video Analysis Service – RTVAS) et Service de notification du propriétaire (Owner Notification Service – ONS), comme illustré dans la figure 4.1.

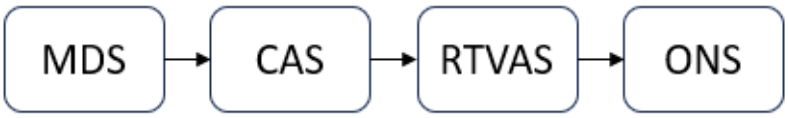

Figure 4.1: Plan de composition du scénario de l'étude de cas

Dans le plan de composition présenté dans la figure 4.1, chaque service abstrait peut être exécuté par plusieurs services IoT candidats avec différentes valeurs de QoS. Dans le tableau 4.1, nous présentons un échantillon de trois services IoT candidats réels pour chaque service abstrait du scénario présenté. Cela se traduit par 4^3 compositions de services IoT possibles.

AS	CS	Fournisseur du service
MDS	CS_1^1	ADT
	CS_2^1	SimpliSafe
	CS_3^1	Vivint
CAS	CS_1^2	Arlo
	CS_2^2	Nest
	CS_3^2	Ring
RTVAS	CS_1^3	IBM Watson Visual Recognition
	CS_2^3	Deep Sentinel
	CS_3^3	Avigilon
ONS	CS_1^4	Verisure
	CS_2^4	Frontpoint
	CS_3^4	Xfinity Home

Tableau 4.1: Exemples de services IoT candidats réels

4.2.2 Évaluation de DT-WOA

La présente section est consacrée à l'évaluation de l'algorithme DT-WOA. Nous commencerons par la description des datasets utilisés dans nos simulations, avant de procéder à l'analyse des résultats obtenus.

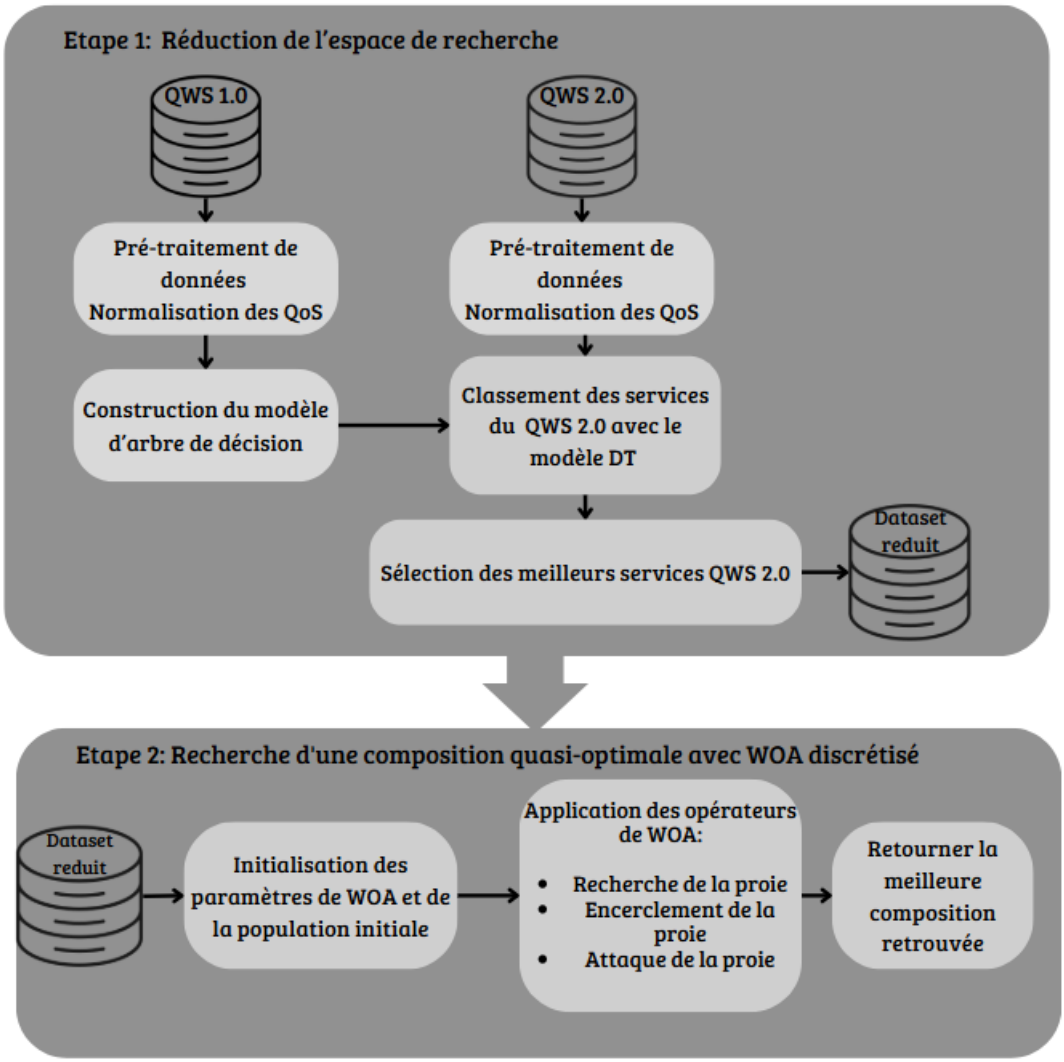

Figure 4.2: Processus de mise en œuvre de DT-WOA dans le scénario d'étude de cas

4.2.2.1 Datasets

Les simulations reposent sur deux datasets publics, QWS 1.0 et QWS 2.0 [4], qui fournissent des caractéristiques de services réels. Le dataset QWS 1.0 est composé de 365 services utilisés pour l'apprentissage supervisé du modèle d'arbre de décision. Chaque service de ce dataset est décrit par ses valeurs de QoS et une étiquette de catégorie parmi les suivantes : bronze, argent, platine ou or. L'implémentation de l'arbre de décision a été réalisée à l'aide de la bibliothèque Scikit-Learn. Le dataset QWS 2.0, comprenant 2507 enregistrements de services réels, a servi à la fois pour la classification des services à l'aide du modèle d'arbre de décision préalablement entraîné et pour la phase de sélection d'une composition de services en fonction des préférences de l'utilisateur, via l'algorithme WOA. Ce dataset a été utilisé dans des travaux portant sur la composition de services IoT, notamment ceux présentés par [1,21]. Les deux datasets considèrent les mêmes neuf premiers paramètres de QoS, à savoir : temps de réponse, disponibilité, débit,

taux de succès, fiabilité, conformité, bonnes pratiques, latence et documentation.

La figure 4.2 présente l'organigramme qui illustre les étapes principales de la mise en œuvre de l'approche proposée.

4.2.2.2 Résultats des simulations

Cette section présente une étude comparative entre l'algorithme proposé DT-WOA et l'algorithme WOA sans le modèle DT. Les paramètres de simulation sont présentés dans le tableau 4.2. Rappelons que, dans notre cas, nous utilisons une fonction de fitness à minimiser.

Paramètre	Valeur
DT labels	bronze, argent, or ou platine.
$nPop$	30
n	4
m	20, 40, 60, 80, 100
r	9

Tableau 4.2: Paramètres de simulation de DT-WOA

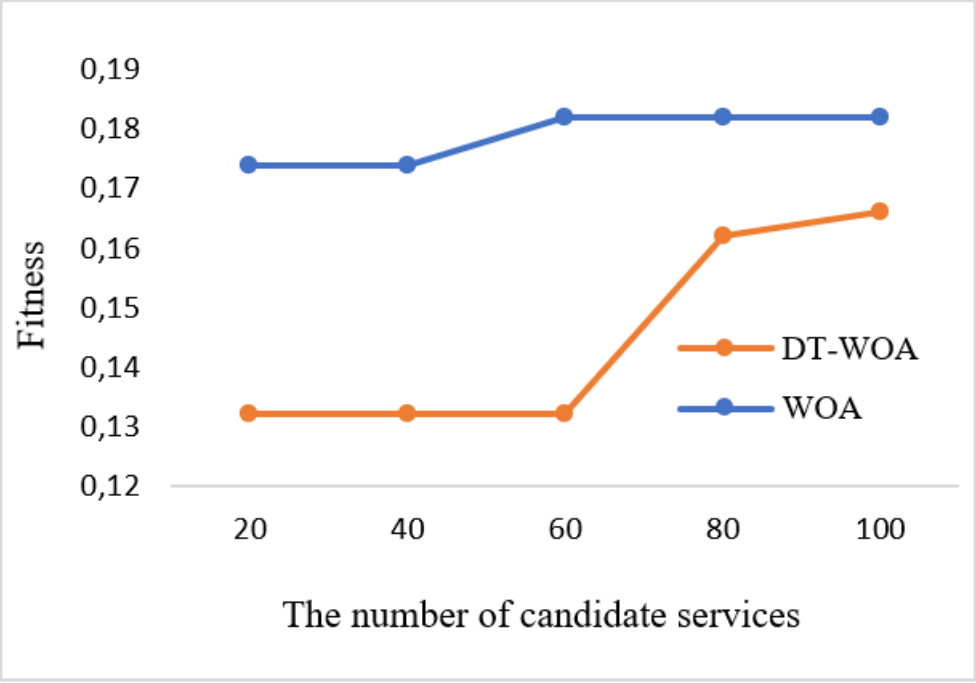

Figure 4.3: Variation de la fitness en fonction du nombre croissant de services candidats

La figure 4.3 présente l'évolution de la fitness en fonction du nombre croissant de services candidats. Les résultats obtenus montrent que, pour l'ensemble des cas considérés, les valeurs

de fitness des solutions générées par DT-WOA sont meilleures comparées à celles obtenues avec l'algorithme WOA original. Ceci met en évidence l'efficacité de DT-WOA pour une exploration plus performante de l'espace de recherche. L'intégration du modèle d'arbre de décision favorise ce niveau de performance en permettant une recherche plus précise, grâce à la réduction du nombre de solutions à explorer.

4.3 Étude des performances de DALOA

Cette section présente le scénario de composition de services IoT adopté pour évaluer notre seconde contribution DALOA. Elle rapporte également les résultats des simulations réalisées pour évaluer les performances de DALOA en comparaison avec ESWOA [41] et à une adaptation de l'algorithme génétique nommée AGA. ESWOA a été choisi pour les raisons principales suivantes : il prend en compte les contraintes globales, il utilise le même dataset QWS [4] que celui choisi pour l'évaluation de DALOA, et il montre de bonnes performances [41]. De plus, sachant que l'algorithme génétique fait partie des algorithmes les plus efficaces et les plus utilisés dans la littérature [1], nous avons implémenté une adaptation de l'approche GA proposée dans [95] en ajoutant les contraintes globales de QoS. Ces algorithmes sont implémentés sous Matlab version R2014b sur un ordinateur personnel HP équipé d'un processeur Intel Core i5-6200U à 2,4 GHz, de 8 Go de RAM et de Windows 10 Professionnel (système 64 bits).

4.3.1 Scénario du cas d'étude de DALOA

Pour démontrer les performances de DALOA, nous avons choisi de l'appliquer au problème de composition de services IoT dans le domaine médical. Considérons une maison de retraite intelligente intégrant des innovations en matière de bâtiments intelligents et de santé intelligente. Les innovations des bâtiments intelligents permettent de gérer l'interactivité entre les maisons intelligentes, leurs collaborateurs (personnel) et leurs résidents, tandis que les innovations en matière de santé intelligente simplifient la vie des résidents âgés et surveillent les constantes de santé quotidiennes des résidents atteints de maladies chroniques. Nous nous concentrons sur le cas des résidents ayant des problèmes auditifs (malentendants) et parlant une langue étrangère. Ces résidents sont équipés d'une prothèse auditive intelligente connectée à leur smartphone et à une station centrale de traitement des données via Internet. Pour répondre aux exigences fonctionnelles de ce scénario, considérons le plan de composition comprenant huit services atomiques, présenté dans la figure 4.4.

Une description de la fonctionnalité de chaque tâche du plan de composition de la figure 4.4 est fournie ci-après :

1. **Service de détection de son :** Ce premier service est utilisé pour détecter les différents sons environnants.

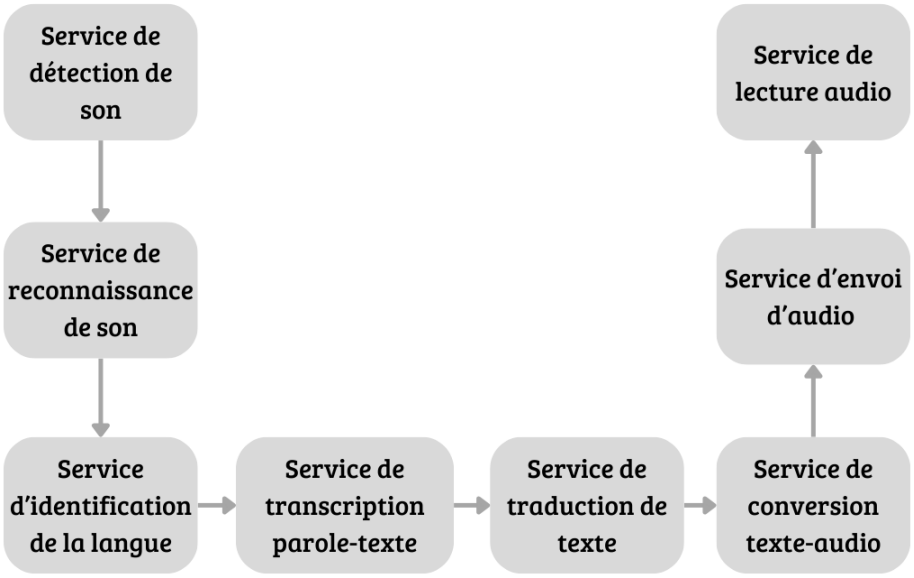

Figure 4.4: Le plan de composition correspondant au scénario de DALOA

2. **Service de reconnaissance de son :** Ce service reconnaît la voix de l'utilisateur afin de l'isoler des divers sons environnants, tels que le bruit et les voix des autres personnes. Il empêche également l'utilisateur de percevoir un retour de sa voix.
3. **Service d'identification de la langue :** Ce service identifie la langue utilisée pour communiquer avec l'utilisateur. L'utilisateur peut communiquer avec d'autres résidents ou avec le personnel médical de la maison de retraite.
4. **Service de transcription parole-texte :** Une fois la langue détectée, une transcription de la parole en texte est effectuée afin de permettre sa traduction.
5. **Service de traduction de texte :** À cette étape, la traduction du texte généré est réalisée.
6. **Service de conversion texte-audio :** Le texte produit est converti en un fichier audio pour permettre la communication via le smartphone de l'utilisateur.
7. **Service d'envoi d'audio :** À cette étape, le fichier audio précédemment généré par le service de conversion texte-audio est envoyé au smartphone de l'utilisateur.
8. **Service de lecture audio :** Le dernier service de ce plan d'exécution est utilisé pour lire le fichier audio généré à l'aide du smartphone.

Après avoir défini le plan de composition, nous procéderons à la réalisation de nos cas de simulation.

4.3.2 Test des performances de DALOA en comparaison avec ES-WOA et AGA

Afin d'évaluer les performances de notre approche, nous avons utilisé le dataset QWS 2.0 [4], largement utilisé dans les travaux traitant le problème de composition de services IoT [1, 18, 21]. Pour notre étude, nous avons retenu les cinq premiers paramètres QoS du dataset, à savoir le temps de réponse (RT), la disponibilité (AV), le débit (TH), le succès (SC) et la fiabilité (RL). Notons que, pour ces tests, nous visons à maximiser la fonction d'utilité, contrairement au test de DT-WOA. La table 4.3 fournit la définition des cinq paramètres tels qu'ils sont décrits dans les datasets.

Paramètre QoS	**Description**
Temps de réponse	Temps nécessaire pour envoyer une requête et recevoir une réponse.
Disponibilité	Pourcentage d'invocations réussies.
Débit	Nombre total d'invocations par seconde.
Taux de succès	Nombre d'invocations répondues/nombre d'invocations total.
Fiabilité	Ratio du nombre de messages d'erreur sur le nombre total de messages.

Tableau 4.3: Description des paramètres QoS selon [4]

Afin d'assurer la diversité des scénarios expérimentaux, les services candidats ont été extraits aléatoirement du dataset. Le tableau 4.4 présente un échantillon de services candidats extraits du QWS 2.0. La population initiale de taille $nPop$ a été générée aléatoirement à travers l'espace de recherche, permettant ainsi d'explorer une large gamme de solutions potentielles. De même, les poids associés aux critères QoS lors du calcul de la valeur d'utilité (équation 2.8) ont également été générés aléatoirement à chaque stimulation, afin de considérer différents cas de préférences d'utilisateur.

ServiceAbstrait	CS	RT	AV	TH	SC	RL
Détection de son	CS_{11}	251.5	78	8.5	78	73
	CS_{12}	170	91	28.6	97	80
	CS_{13}	321.4	88	2.8	96	73
Reconnaissance de son	CS_{21}	126.17	98	12	100	67
	CS_{22}	440.06	91	7	97	60
	CS_{23}	138	85	33	95	73
Identification de la langue	CS_{31}	260.6	94	1.8	98	67
	CS_{32}	486	94	3.1	98	73
	CS_{33}	120	83	19.3	84	73
Transcription parole-texte	CS_{41}	299.83	85	2.2	86	53
	CS_{42}	141	92	1.4	97	73
	CS_{43}	89	83	40.4	84	78
Traduction de texte	CS_{51}	269.38	99	9	100	67
	CS_{52}	138.25	98	14.5	100	73
	CS_{53}	269	96	21.5	99	58
Conversion texte-audio	CS_{61}	102	91	9.2	97	67
	CS_{62}	3163.2	93	2.3	98	73
	CS_{63}	199.17	56	7.3	56	67
Envoi d'audio	CS_{71}	415.93	95	7.3	95	67
	CS_{72}	463.6	94	2.1	98	73
	CS_{73}	201.6	91	1.5	97	73
Lecture audio	CS_{81}	146.83	85	2.1	95	83
	CS_{82}	164.5	72	12.5	72	73
	CS_{83}	3273.75	84	1.6	85	73

Tableau 4.4: Échantillon de trois services extrait du dataset pour chaque tâche du plan de composition

Dans ce qui suit, nous détaillons les différents scénarios de test mis en œuvre afin d'évaluer les performances de notre approche. Ces scénarios, récapitulés dans le tableau 4.5, ont été conçus en faisant varier les paramètres suivants : le nombre d'itérations, le nombre de tâches dans le plan de composition AS et le nombre de services candidats CS dans chaque AS . Rappelons que la taille de la population $nPop$ est de 30, tandis que le nombre de critères de QoS est de 5.

Scénario de test	Nombre de CS	Nombre de AS	Nombre de simulations	Nombre d'itérations
Scénario 1	250	8	1	1-200
Scénario 2	250-1500	8	50	50
Scénario 3	1500	10, 20, 30, 40, 50	50	50
Scénario 4	250-1500	25	50	50

Tableau 4.5: Paramètres des simulations de DALOA

4.3.2.1 Scénario de test 1

Dans ce scénario, nous avons évalué l'utilité de DALOA en fonction du nombre d'itérations. Pour cela, nous avons fixé le nombre de AS à 8 (le nombre de tâches dans le plan de composition présenté dans la figure 4.4), le nombre de CS dans chaque AS est égal à 250, tandis que le nombre d'itérations varie de 1 à 200. L'objectif de ce test est de déterminer le nombre d'itérations à partir duquel DALOA atteint un niveau de performance satisfaisant, afin de l'utiliser dans les prochains tests de cette section.*

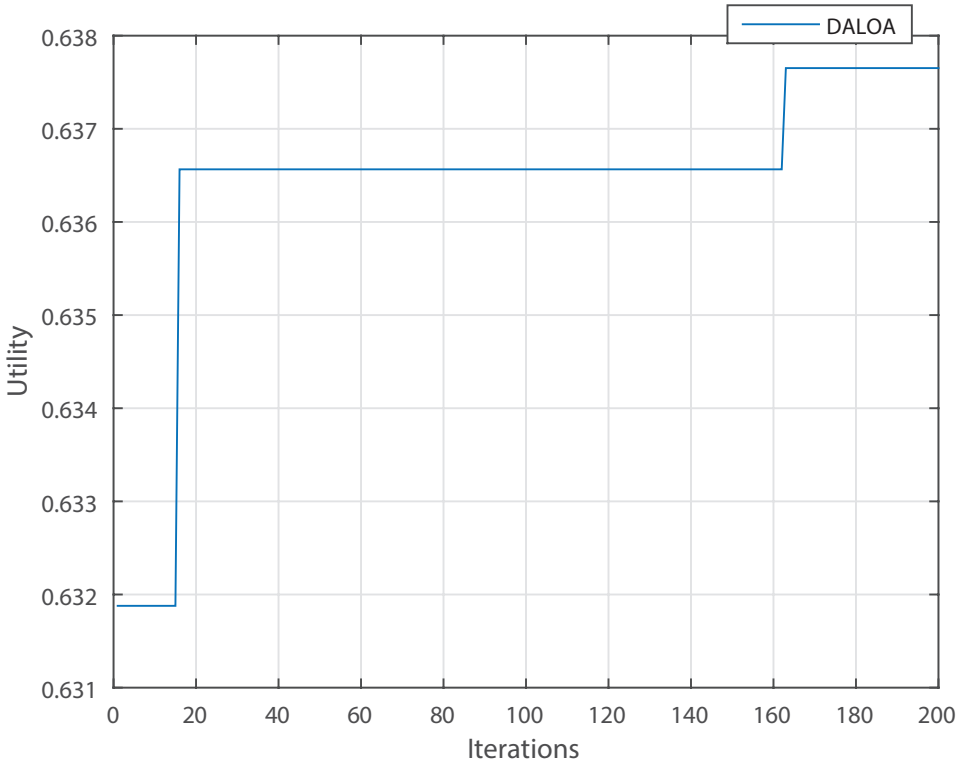

Figure 4.5: Évolution de l'utilité en fonction du nombre d'itérations

Les résultats présentés dans la figure 4.5 indiquent que l'utilité atteint une valeur stable à partir de la 20ème itération. Nous remarquons également une amélioration insignifiante de l'utilité (entre 0.636 et 0.637) après la 160ème itération. Par conséquent, nous avons fixé le nombre d'itérations à 50 dans les prochains tests.

4.3.2.2 Scénario de test 2

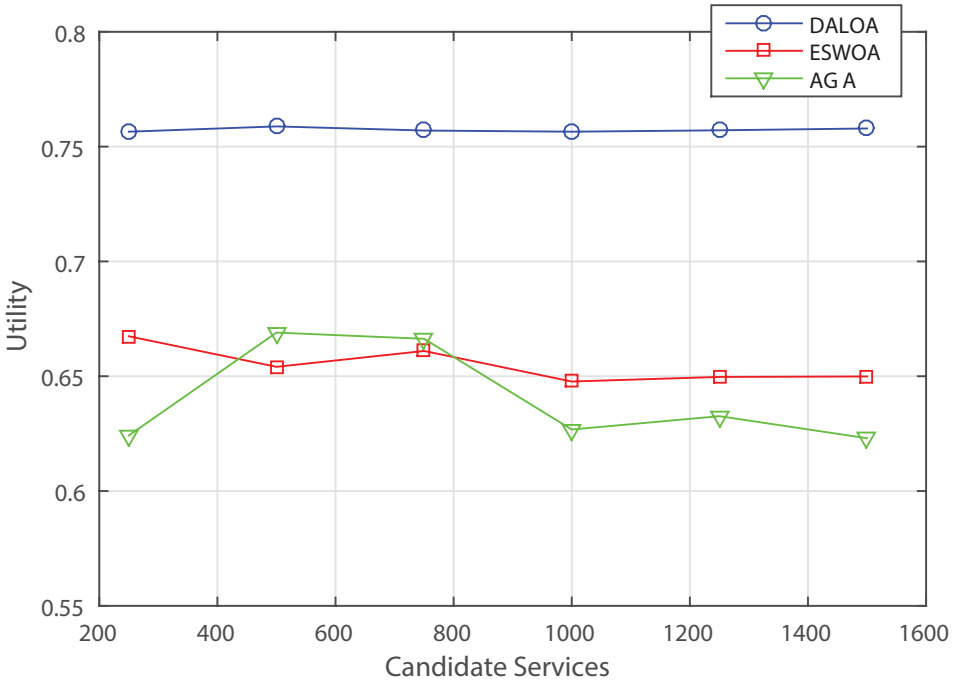

Figure 4.6: Évolution de l'utilité en fonction du nombre de CS

Ce scénario évalue les performances de DALOA en comparaison avec ESWOA et AGA pour le cas du patient malentendant. Aussi, nous avons fixé le nombre de AS à 8 tout en faisant varier le nombre de CS de 250 à 1500, afin d'évaluer la sensibilité des algorithmes à l'augmentation de la taille de l'espace de recherche.

Les résultats présentés en figure 4.6 montrent que la qualité de la solution renvoyée par DALOA est meilleure que celle retournée par ESWOA et AGA. En effet, DALOA parvient à maintenir une bonne valeur d'utilité, même lorsque l'espace de recherche s'élargit. Cela s'explique par la capacité de DALOA à équilibrer efficacement l'exploration de nouvelles régions de l'espace de recherche et l'exploitation de solutions prometteuses, ce qui lui permet de converger vers des optimums globaux.

La figure 4.7 présente la variation du temps d'exécution des algorithmes en fonction du nombre de CS. Dans la plupart des cas, DALOA a un temps d'exécution inférieur à celui des algorithmes

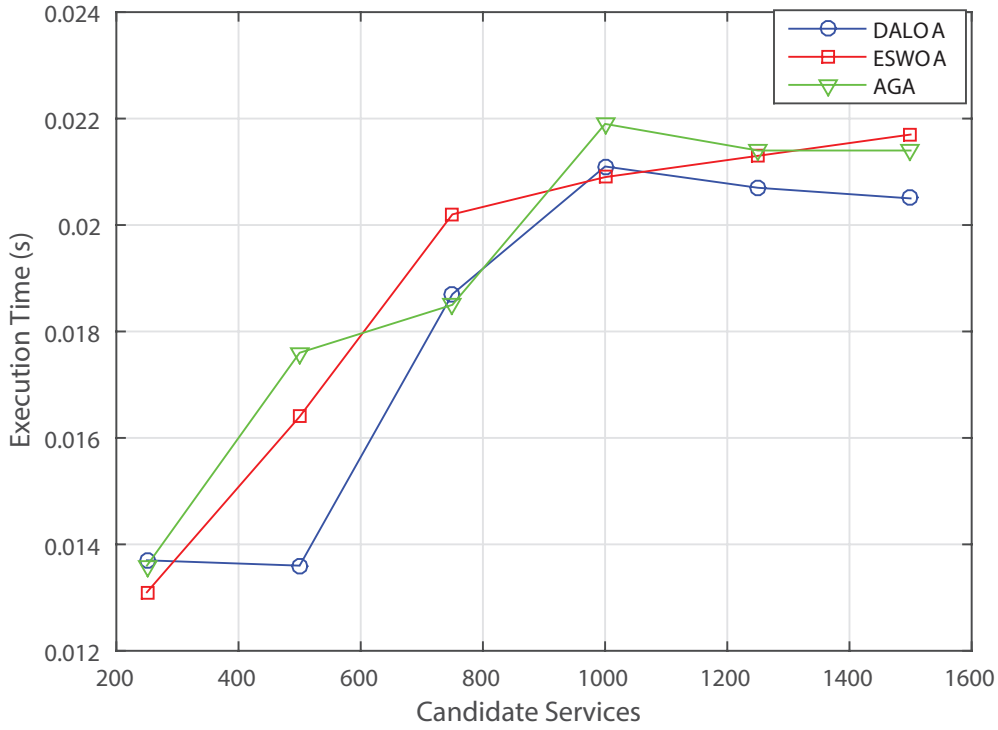

Figure 4.7: Évolution du temps d'exécution en fonction du nombre de CS

ESWOA et AGA. Comme le montre la figure 4.7, lorsque le nombre de services candidats augmente, le temps d'exécution de DALOA n'augmente pas de manière significative. Cela est dû à la définition de deux algorithmes différents pour les opérateurs de roaming et de reproduction, permettant à DALOA d'être efficace en moins de temps. La valeur minimale de 0,0136 s est obtenue lorsque le nombre de CS est égal à 500, ce qui représente le meilleur temps d'exécution de DALOA pour cette expérience. Le temps d'exécution le plus élevé pour DALOA est de 0,0211 s contre 0,0217 s pour ESWOA et 0,0219 s pour AGA.

4.3.2.3 Scénario de test 3

Ce test a pour objectif d'évaluer l'impact de l'augmentation du nombre de AS, qui représente le nombre de tâches du plan de composition, sur les performances de DALOA par rapport à ESWOA et AGA. Pour ce faire, nous avons fixé le nombre de CS à 1500 tout en faisant varier le nombre de AS de 10 à 50.

La figure 4.8 montre que DALOA surpasse ESWOA et AGA en termes de qualité de la composition retournée. Nous observons également que l'augmentation du nombre de AS affecte la qualité de la solution (diminution de la valeur de l'utilité). En effet, l'utilité de DALOA diminue légèrement de 0,6679 à 0,6018, l'utilité de ESWOA varie entre 0,6380 et 0,4956, tandis que l'utilité de AGA fluctue entre 0,63900 et 0,4666. Par conséquent, la valeur minimale de 0,60188

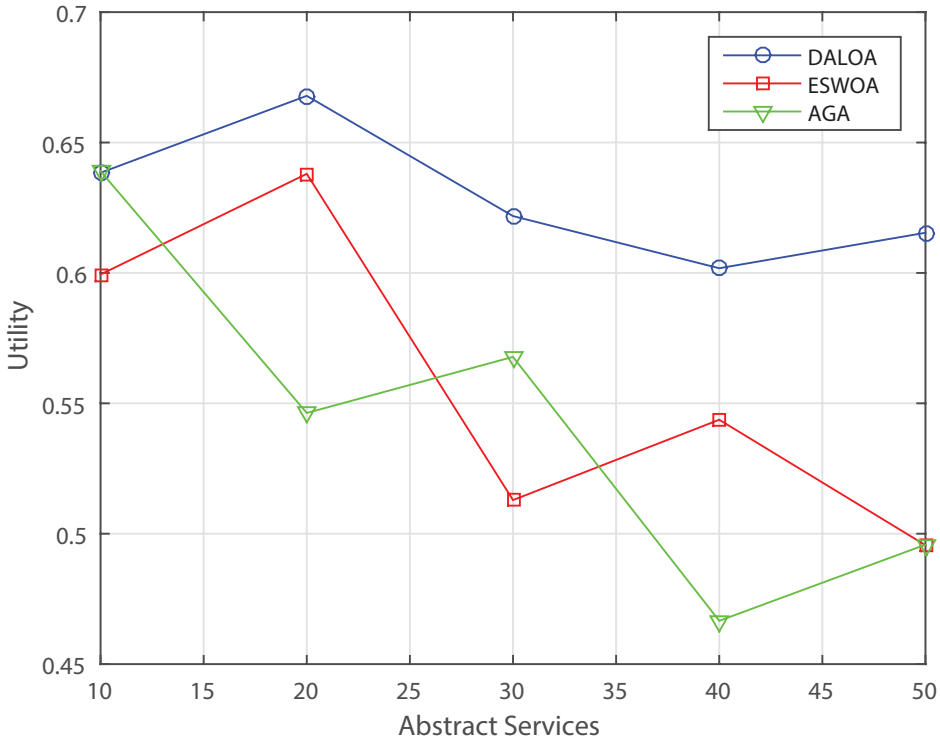

Figure 4.8: Évolution de l'utilité en fonction du nombre de *AS*

est obtenue lorsque le nombre de *AS* est égal à 40, ce qui représente la pire valeur d'utilité pour DALOA contre 0,54377 pour ESWOA et 0,46666 pour AGA. Les variations observées dans les graphiques sont dues au comportement probabiliste et à l'utilisation de nombres aléatoires, caractérisant les approches basées sur les métaheuristiques. Les résultats de ce test démontrent que DALOA est plus apte que ESWOA et AGA pour le traitement de requêtes utilisateur complexes, pouvant nécessiter jusqu'à 50 tâches pour les satisfaire.

D'après les résultats présentés dans la figure 4.9, il est clair que le nombre de AS a un impact direct sur le temps d'exécution des trois algorithmes. Ces résultats montrent que DALOA offre généralement de meilleures performances en termes de temps d'exécution par rapport à AGA, tout en étant, dans la majorité des cas, supérieur ou proche de ESWOA. En effet, le temps d'exécution maximal atteint par DALOA est de 0,2234 s, contre 0,2857 s pour ESWOA et 0,4564 s pour AGA.

4.3.2.4 Scénario de test 4

Dans ce test, nous avons étendu le scénario 2 en augmentant la valeur du nombre de tâches afin d'évaluer la capacité des algorithmes à gérer des requêtes plus complexes.

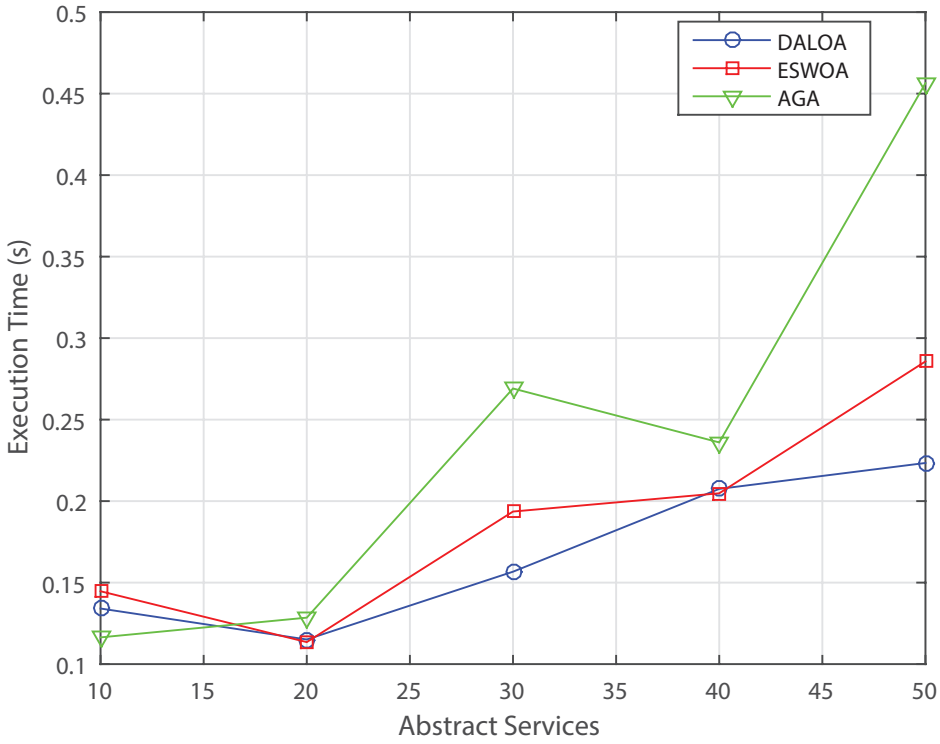

Figure 4.9: Évolution du temps d'exécution en fonction du nombre de *AS*

Les résultats, présentés dans la figure 4.10, démontrent que DALOA conserve un niveau d'utilité élevé par rapport à AGA et ESWOA, même face à une augmentation significative du nombre de AS qui passe de 8 à 25.

D'après la figure 4.11, nous observons, d'une part, que le temps d'exécution de DALOA est meilleur que celui de AGA et, d'autre part, que DALOA est, dans la plupart des cas, plus rapide qu'ESWOA pour trouver une composition proche de l'optimale. En effet, le meilleur temps d'exécution est de 0,0194 s pour DALOA, 0,0351 s pour ESWOA et 0,0157 s pour AGA, tandis que le pire temps d'exécution est de 0,0434 s pour DALOA, 0,0454 s pour ESWOA et 0,043 s pour AGA.

L'analyse comparative des résultats présentés dans cette section démontre que DALOA se distingue par un meilleur compromis entre le temps d'exécution et la qualité de ses solutions en comparaison avec ESWOA et AGA.

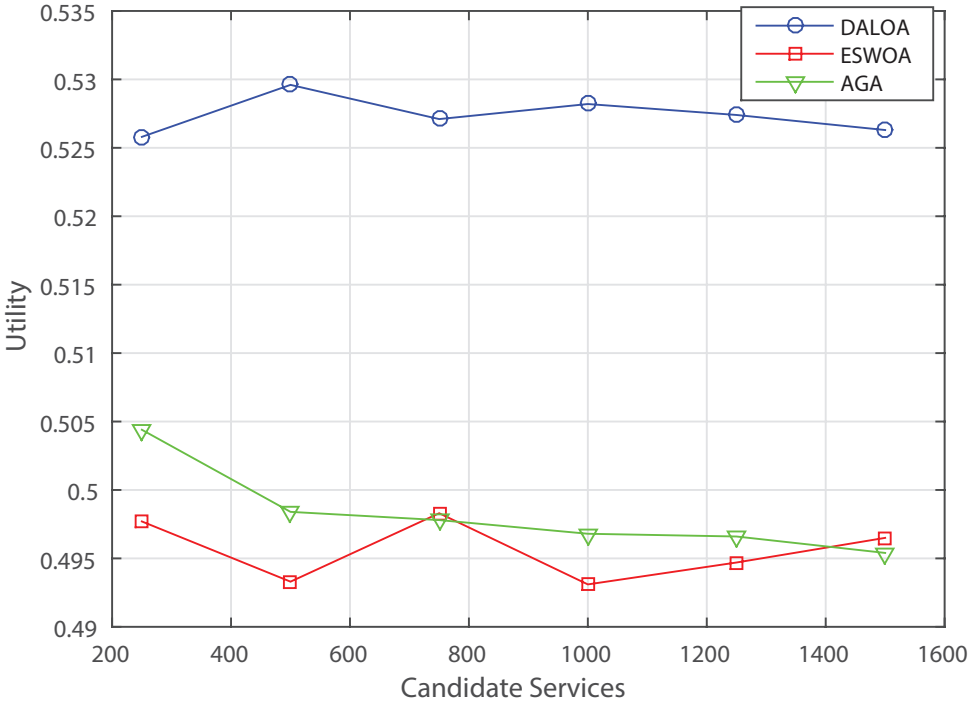

Figure 4.10: Évolution de l'utilité en fonction du nombre de CS

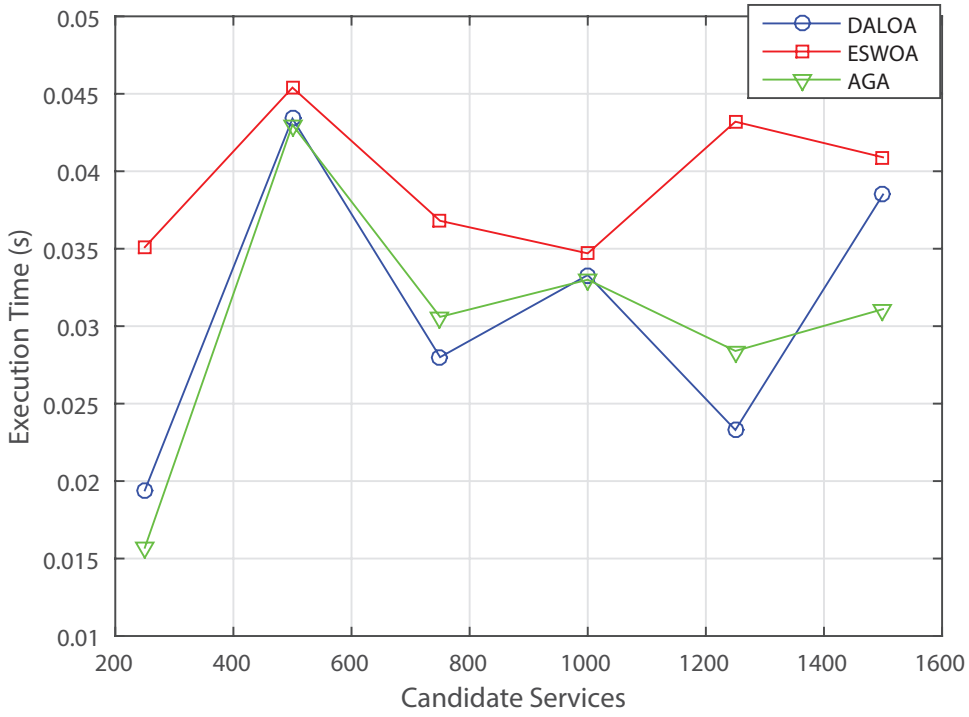

Figure 4.11: Évolution du temps d'exécution en fonction du nombre de CS

4.3.3 Complexité temporelle

Pour calculer la complexité des algorithmes implémentés, nous supposons que n représente le nombre de services abstraits (AS) et p la taille de la population initiale de DALOA, divisée en deux sous-populations : nomade et pride. Lorsque n tend vers l'infini, nous supposons que la taille de la sous-population est significativement plus petite que n. Ainsi, les pourcentages de la sous-population pride (FNbr(100-S)), de la sous-population pride participant au roaming (FNbr(R)), de la sous-population nomade (FNbr(N)), de la sous-population participant à la reproduction (FNbr(Ma)), et de la sous-population pride participant à la migration (FNbr(Mi)) sont tous significativement plus petits que n.

4.3.3.1 Complexité de DALOA

La complexité de l'algorithme 4 dépend de FNbr(100-S) et des opérations de mises à jour et d'évaluation des nouvelles compositions en utilisant la fonction d'utilité. Cette fonction utilise le vecteur QoS présenté dans l'équation 2.7 pour calculer la QoS d'une composition. Ce vecteur, basé sur le nombre de AS, est calculé en O(n). Par conséquent, la complexité de la fonction d'utilité et donc de l'algorithme 4 est O(n).

Dans l'algorithme 5, les temps nécessaires pour calculer la probabilité selon l'équation 3.15 et mettre à jour la composition nomade selon l'équation 3.14 sont constants à l'intérieur de la boucle. Par conséquent, la complexité de l'algorithme 5 est O(n) puisque la meilleure composition de la population sera mise à jour en utilisant la fonction d'utilité.

En outre, l'algorithme 7 dépend de la taille du vecteur d'entrée, qui représente le nombre de AS, donc la complexité de l'algorithme 7 est O(n).

La boucle de l'algorithme 6 s'exécute FNbr(Ma) fois tout en appelant deux fois l'algorithme 7 à chaque itération pour appliquer les équations 3.16 et 3.17. Ainsi, la complexité de cette boucle est O(2n). De plus, l'algorithme 6 appelle la fonction d'utilité, donc la complexité totale de cet algorithme est O(2n) + O(n). Concernant l'opérateur mating nomade, il suit les mêmes étapes que l'algorithme 6, sauf qu'une composition nomade se reproduit avec une seule autre composition nomade. Par conséquent, cet opérateur a la même complexité que l'algorithme 6 qui est O(2n) + O(n). La complexité de l'algorithme 8 est O(FNbr(Mi)) car la migration à l'intérieur de la boucle est effectuée en un temps constant, et l'instruction de mise à jour n'appelle pas la fonction d'utilité.

La complexité de l'algorithme 9 de DALOA correspond à la somme des complexités des algorithmes 4,5, 6 et 8. L'algorithme 7 n'est pas pris en compte dans le calcul, car il est appelé par l'algorithme 6. En outre, la complexité de l'algorithme 6 est prise en compte deux fois (pour les sous-populations pride et nomade). Notez que la complexité de la vérification des contraintes globales de QoS est constante. En conséquence, la complexité de DALOA est O(n).

4.3.3.2 Complexité de ESWOA

L'algorithme ESWOA inclut deux stratégies de recherche : locale et globale. La complexité de chaque stratégie est O(pn) car elles utilisent la même fonction d'utilité donnée par l'équation 2.8 et s'appliquent à l'ensemble de la population. Par conséquent, la complexité de l'algorithme ESWOA est O(n), car p est considéré comme insignifiant par rapport à n et la complexité de la vérification des contraintes globales de QoS est constante.

4.3.3.3 Complexité de AGA

En ce qui concerne l'algorithme AGA, qui est une adaptation de GA prenant en compte les contraintes globales, il se compose de deux opérateurs : croisement et mutation. Ces opérateurs sont appliqués à un pourcentage de la population et appellent la fonction d'utilité pour évaluer les compositions. Étant donné que la complexité de la vérification des contraintes globales est constante, la complexité de AGA est également O(n).

4.4 Conclusion

Dans le cadre de la validation des approches proposées, celles-ci ont été appliquées à des scénarios réalistes représentatifs du domaine de l'IoT. Plus précisément, DT-WOA a été mis en œuvre pour la détection d'intrusion au sein d'une maison intelligente. Les résultats des simulations effectuées mettent en évidence l'efficacité de DT-WOA par rapport à WOA, notamment lors de la recherche de compositions satisfaisant les préférences utilisateur en termes de QoS. Par ailleurs, DALOA a été évalué à travers une série de tests menés dans le cadre d'un scénario médical réel, déployé au sein d'un environnement IoT. Les tests menés démontrent que DALOA offre un meilleur rapport qualité/temps d'exécution que les algorithmes ESWOA et AGA, lors de la recherche d'une composition satisfaisant les préférences et les contraintes globales de QoS de l'utilisateur.

Conclusion Générale

Pour clôturer ce rapport, nous allons récapituler les défis et les enjeux initiaux, résumer nos contributions, citer les apports des approches proposées, et offrir un aperçu sur nos perspectives futures.

Les défis de la sélection de services IoT lors du processus de composition

L'Internet des objets est de plus en plus répandu dans des secteurs cruciaux tels que la santé, le marketing, l'agriculture et l'industrie. La technologie IoT relie les objets du quotidien à Internet en leur attribuant une identité unique. Cela permet à ces objets de fournir des fonctionnalités sous forme de services aux utilisateurs. Face à des requêtes utilisateurs complexes, un service IoT individuel s'avère insuffisant. La composition de plusieurs services devient alors indispensable pour répondre à ce type de demande. Le processus de composition de services permet de combiner les fonctionnalités des objets IoT pour créer un service IoT composite répondant à la fois aux exigences fonctionnelles et de QoS [5, 10, 11].

L'essor des environnements intelligents a entraîné une explosion du nombre de services IoT présentant des fonctionnalités similaires, mais se distinguant par leurs performances en termes de QoS. Cela augmente la complexité du problème de sélection des services, en soulevant des défis majeurs : D'une part, la nécessité de comparer un grand nombre de compositions candidates aux niveaux de QoS variables ; d'autre part, la difficulté de garantir que les services sélectionnés répondent non seulement aux exigences fonctionnelles, mais également aux contraintes de QoS imposées par l'utilisateur.

Nos contributions

Nous avons présenté dans cette thèse deux approches proposées pour la résolution du problème de composition de services basé sur les critères de QoS.

Les contributions de l'approche DT-WOA sont :

— La réduction de l'espace de recherche en éliminant les services les moins pertinents via un

modèle d'arbre de décision.

— La proposition d'une version discrète de l'algorithme WOA, connu pour offrir un bon équilibre entre l'exploration et l'exploitation de l'espace de recherche.

Les principales contributions de l'approche DALOA sont :

— Proposition d'un modèle de composition de services IoT axé sur la QoS, prenant en compte les préférences de QoS de l'utilisateur et les contraintes globales de QoS utilisées comme limites supérieure et inférieure lors de la sélection des services IoT composites.

— Pour optimiser notre algorithme DALOA, nous avons sélectionné et adapté trois opérateurs de LOA à savoir le roaming, le mating et la migration. Cette combinaison assure une diversité de population et un bon équilibre entre l'exploration et l'exploitation de l'espace de recherche.

— Définition d'un nouvel opérateur $\oplus$ pour discrétiser l'algorithme LOA, permettant ainsi de l'adapter au problème de composition de services IoT.

— Évaluation de l'approche proposée sur la base de différents scénarios de simulation appliqués à un cas médical réel de composition de services IoT.

— Intégration des contraintes globales de QoS dans DALOA.

Les apports des approches proposées

L'algorithme DT-WOA, présenté dans cette étude, combine un modèle d'arbre de décision avec une version discrète de WOA. Les résultats de simulations démontrent que DT-WOA surpasse l'algorithme WOA original dans la recherche d'une solution proche de l'optimale qui répond aux préférences QoS des utilisateurs. Cela confirme l'efficacité de l'intégration du modèle DT dans l'approche proposée. L'approche DALOA se distingue par son aptitude à offrir un meilleur compromis entre le temps d'exécution de l'algorithme et la qualité de la composition IoT obtenue par rapport aux approches de comparaison. En effet, les résultats de simulations démontrent que DALOA atteint un meilleur rapport qualité/temps d'exécution comparé aux algorithmes ESWOA et AGA, soulignant son efficacité et sa performance.

Perspectives

Voici une liste de perspectives émergeant de nos travaux :

— Concevoir et implémenter une variante parallèle de l'algorithme DALOA pour améliorer les performances de l'approche proposée en termes de temps de réponses.

— Concevoir et implémenter une variante multi-objective de l'algorithme DALOA .

— Améliorer la prédiction de la QoS et la sélection de services IoT composites en intégrant de nouvelles techniques de l'apprentissage automatique (réseaux de neurones, clustering, etc.).

— Utiliser l'apprentissage par renforcement pour adapter dynamiquement l'algorithme aux changements de l'environnement IoT.

— Valider les performances de l'approche proposée sur des ensembles de données caractérisant des services IoT réels.

Références

[1] Rabah Boucetti, Sofiane Mounine Hemam, and Ouassila Hioual. An approach based on genetic algorithms and neural networks for QoS-aware IoT services composition. *Journal of King Saud University-Computer and Information Sciences*, 2022.

[2] Samia Chibani Sadouki and Abdelkamel Tari. Multi-objective and discrete elephants herding optimization algorithm for QoS aware web service composition. *RAIRO-Operations Research*, 53(2) :445–459, 2019.

[3] Seyedali Mirjalili and Andrew Lewis. The whale optimization algorithm. *Advances in engineering software*, 95 :51–67, 2016.

[4] QWS Dataset Repository, 2007. Consulté le 29 Mars 2024.

[5] Gustavo André Setti Cassel, Vinicius Facco Rodrigues, Rodrigo da Rosa Righi, Marta Rosecler Bez, Andressa Cruz Nepomuceno, and Cristiano André da Costa. Serverless computing for Internet of Things : A systematic literature review. *Future Generation Computer Systems*, 128 :299–316, 2022.

[6] Boucetti Rabah, Hemam Sofiane Mounine, and Hioual Ouassila. QoS-aware IoT services composition : A survey. In *Distributed Sensing and Intelligent Systems*, pages 477–488. Springer, 2022.

[7] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. Service composition approaches in IoT : A systematic review. *Journal of Network and Computer Applications*, 120 :61–77, 2018.

[8] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. Internet of Things applications : A systematic review. *Computer Networks*, 148 :241–261, 2019.

[9] Samia Sadouki Chibani and Abdelkamel Tari. Elephant herding optimization for service selection in QoS-aware web service composition. *International Journal of Computer and Information Engineering*, 11(10) :1124–1128, 2017.

[10] Osama Alsaryrah, Ibrahim Mashal, and Tein-Yaw Chung. Bi-objective optimization for energy aware Internet of Things service composition. *IEEE Access*, 6 :26809–26819, 2018.

[11] Bin Cheng, Jonathan Fuerst, Gurkan Solmaz, and Takuya Sanada. Fog function : Serverless fog computing for data intensive IoT services. In *2019 IEEE International Conference on Services Computing (SCC)*, pages 28–35. IEEE, 2019.

[12] Souhila Ait Hacène Ouhadda, Samia Chibani Sadouki, Achour Achroufene, and Abdelkamel Tari. A discrete adaptive lion optimization algorithm for QoS-driven IoT service composition with global constraints. *Journal of Network and Systems Management*, 32(2) :34, 2024.

[13] Mohammad Alrifai and Thomas Risse. Combining global optimization with local selection for efficient QoS-aware service composition. In *Proceedings of the 18th international conference on World wide web*, pages 881–890, 2009.

[14] Virginie Gabrel, Maude Manouvrier, and Cécile Murat. Web services composition : complexity and models. *Discrete Applied Mathematics*, 196 :100–114, 2015.

[15] Mohammad Alrifai, Thomas Risse, and Wolfgang Nejdl. A hybrid approach for efficient web service composition with end-to-end QoS constraints. *ACM Transactions on the Web (TWEB)*, 6(2) :1–31, 2012.

[16] Virginie Gabrel, Maude Manouvrier, Kamil Moreau, and Cecile Murat. QoS-aware automatic syntactic service composition problem : Complexity and resolution. *Future Generation Computer Systems*, 80 :311–321, 2018.

[17] Mostafa Ghobaei-Arani and Alireza Souri. LP-WSC : a linear programming approach for web service composition in geographically distributed cloud environments. *The Journal of Supercomputing*, 75(5) :2603–2628, 2019.

[18] Mehdi Hosseinzadeh, Hawkar Kamaran Hama, Marwan Yassin Ghafour, Mohammad Masdari, Omed Hassan Ahmed, and Hemn Khezri. Service selection using multi-criteria decision making : a comprehensive overview. *Journal of Network and Systems Management*, 28 :1639–1693, 2020.

[19] V Pavan Kumar, Sudheer Shetty, DR Janardhana, and AP Manu. QoS aware service composition in IoT using heuristic structure and genetic algorithm. *Mathematical Statistician and Engineering Applications*, 71(3) :750–766, 2022.

[20] Ronghan Wang and Junwei Lu. QoS-aware service discovery and selection management for cloud-edge computing using a hybrid meta-heuristic algorithm in IoT. *Wireless Personal Communications*, 126(3) :2269–2282, 2022.

[21] Amal Kouicem, Mohamed Essaid Khanouche, and Abdelkamel Tari. Novel bat algorithm for QoS-aware services composition in large scale Internet of Things. *Cluster Computing*, pages 1–15, 2022.

[22] Souhila Ait Hacene Ouhadda, Samia Chibani Sadouki, and Abdelkamel Tari. Decision tree with whale optimization algorithm (DT-WOA) for QoS-based IoT service composition. In *2024 1st International Conference on Electrical, Computer, Telecommunication and Energy Technologies (ECTE-Tech)*, pages 1–7. IEEE, 2024.

[23] Maziar Yazdani and Fariborz Jolai. Lion optimization algorithm (LOA) : a nature-inspired metaheuristic algorithm. *Journal of computational design and engineering*, 3(1) :24–36, 2016.

[24] Prem Prakash Jayaraman Ampalavanapillai Nirmalathas Anas Dawod, Dimitrios Georgakopoulos. A survey of techniques for discovering, using, and paying for third-party IoT sensors. *Advances in Intelligent Sensors and IoT Solutions*, 24, 2024.

[25] Mark Weiser. The computer for the 21 st century. *Scientific american*, 265(3) :94–105, 1991.

[26] Ashton Kevin. That'Internet of Things' thing. *RFID journal*, 22, 2009.

[27] Idir Aoudia, Saber Benharzallah, Laid Kahloul, and Okba Kazar. A multi-population genetic algorithm for adaptive QoS-aware service composition in fog-IoT healthcare environment. *Int. Arab J. Inf. Technol.*, 18(3A) :464–475, 2021.

[28] Alessandro Bassi and Geir Horn. Internet of Things in 2020 : A roadmap for the future. *European Commission : Information Society and Media*, 22 :97–114, 2008.

[29] M Han and H Zhang. Business intelligence architecture based on Internet of Things. *Journal of Theoretical & Applied Information Technology*, 50(1) :90–95, 2013.

[30] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of Things : Vision, applications and research challenges. *Ad hoc networks*, 10(7) :1497–1516, 2012.

[31] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things : A survey. *Computer networks*, 54(15) :2787–2805, 2010.

[32] Petros Zervoudakis, Nikolaos Karamolegkos, Eleftheria Plevridi, Pavlos Charalampidis, and Alexandros Fragkiadakis. Epoptis : A monitoring-as-a-service platform for internet-of-things applications. *Sensors*, 24(7) :2208, 2024.

[33] Dustin Chen and Qibing Pei. Electronic muscles and skins : a review of soft sensors and actuators. *Chemical reviews*, 117(17) :11239–11268, 2017.

[34] Marc Pastre, Hubert Blanchard, and Maher Kayal. Continuously calibrated magnetic field sensor, June 29 2010. US Patent 7,746,065.

[35] Jochen Wirtz, Paul G Patterson, Werner H Kunz, Thorsten Gruber, Vinh Nhat Lu, Stefanie Paluch, and Antje Martins. Brave new world : service robots in the frontline. *Journal of Service Management*, 29(5) :907–931, 2018.

[36] Wang Zhiliang, Yang Yi, Wang Lu, and Wang Wei. A SOA based IoT communication middleware. In *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pages 2555–2558. IEEE, 2011.

[37] Bruno Costa, Paulo F Pires, and Flávia C Delicato. Modeling SOA-based IoT applications with soaml4IoT. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 496–501. IEEE, 2019.

[38] Poonam Gupta, Trupti P Mokal, DD Shah, and KVV Satyanarayana. Event-driven SOA-based IoT architecture. In *International Conference on Intelligent Computing and Applications : ICICA 2016*, pages 247–258. Springer, 2018.

[39] Fouzia Boudries, Samia Sadouki, and Abdelkamel Tari. A bio-inspired algorithm for dynamic reconfiguration with end-to-end constraints in web services composition. *Service Oriented Computing and Applications*, 13(3) :251–260, 2019.

[40] Liangzhao Zeng, Boualem Benatallah, Anne HH Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. QoS-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5) :311–327, 2004.

[41] Siva Kumar Gavvala, Chandrashekar Jatoth, GR Gangadharan, and Rajkumar Buyya. QoS-aware cloud service composition using eagle strategy. *Future Generation Computer Systems*, 90 :273–290, 2019.

[42] Chandrashekar Jatoth, GR Gangadharan, and Rajkumar Buyya. Computational intelligence based QoS-aware web service composition : a systematic literature review. *IEEE Transactions on Services Computing*, 10(3) :475–492, 2015.

[43] Harald Meyer and Mathias Weske. Automated service composition using heuristic search. In *International Conference on Business Process Management*, pages 81–96. Springer, 2006.

[44] Rainer Berbner, Michael Spahn, Nicolas Repp, Oliver Heckmann, and Ralf Steinmetz. Heuristics for QoS-aware web service composition. In *2006 IEEE International Conference on Web Services (ICWS'06)*, pages 72–82. IEEE, 2006.

[45] Dongmei Liu, Zhiqing Shao, Caizhu Yu, and Guisheng Fan. A heuristic QoS-aware service selection approach to web service composition. In *2009 Eighth IEEE/ACIS International Conference on Computer and Information Science*, pages 1184–1189. IEEE, 2009.

[46] Jing Li, Yongwang Zhao, Min Liu, Hailong Sun, and Dianfu Ma. An adaptive heuristic approach for distributed QoS-based service composition. In *The IEEE symposium on Computers and Communications*, pages 687–694. IEEE, 2010.

[47] Adrian Klein, Fuyuki Ishikawa, and Shinichi Honiden. Efficient heuristic approach with improved time complexity for QoS-aware service composition. In *2011 IEEE International Conference on Web Services*, pages 436–443. IEEE, 2011.

[48] Neeti Kashyap and A Charan Kumari. Hyper-heuristic approach for service composition in Internet of Things. *Electronic Government, an International Journal*, 14(4) :321–339, 2018.

[49] ZhiJun Ding, JunJun Liu, YouQing Sun, ChangJun Jiang, and MengChu Zhou. A transaction and QoS-aware service selection approach based on genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 45(7) :1035–1046, 2015.

[50] Xiaoning Sun, Jiangchuan Chen, Yunni Xia, Qiang He, Yuandou Wang, Xin Luo, Rongqing Zhang, Wuhong Han, and Quanwang Wu. A fluctuation-aware approach for predictive web service composition. In *2018 IEEE International Conference on Services Computing (SCC)*, pages 121–128. IEEE, 2018.

[51] Fadl Dahan, Khalil El Hindi, Ahmed Ghoneim, and Hussain Alsalman. An enhanced ant colony optimization based algorithm to solve QoS-aware web service composition. *IEEE Access*, 9 :34098–34111, 2021.

[52] M Shamim Hossain, Mohd Moniruzzaman, Ghulam Muhammad, Ahmed Ghoneim, and Atif Alamri. Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment. *IEEE Transactions on Services Computing*, 9(5) :806–817, 2016.

[53] Lijuan Wang and Jun Shen. A critical systematic review of service concretization based on bio-inspired approaches. 2014.

[54] Karima Khadir, Nawal Guermouche, Amal Guittoum, and Thierry Monteil. A genetic algorithm-based approach for fluctuating QoS aware selection of IoT services. *IEEE Access*, 10 :17946–17965, 2022.

[55] Qi Yu and Athman Bouguettaya. Efficient service skyline computation for composite service selection. *IEEE Transactions on Knowledge and Data Engineering*, 25(4) :776–789, 2011.

[56] Pablo Rodriguez-Mier, Manuel Mucientes, and Manuel Lama. A hybrid local-global optimization strategy for QoS-aware service composition. In *2015 IEEE International Conference on Web Services*, pages 735–738. IEEE, 2015.

[57] Shangguang Wang, Ao Zhou, Mingzhe Yang, Lei Sun, Ching-Hsien Hsu, and Fangchun Yang. Service composition in cyber-physical-social systems. *IEEE Transactions on Emerging Topics in Computing*, 8(1) :82–91, 2017.

[58] Ying Chen, Jiwei Huang, Chuang Lin, and Jie Hu. A partial selection methodology for efficient qos-aware service composition. *IEEE Transactions on Services Computing*, 8(3) :384–397, 2014.

[59] Aitor Urbieta, Alejandra González-Beltrán, S Ben Mokhtar, M Anwar Hossain, and Licia Capra. Adaptive and context-aware service composition for IoT-based smart cities. *Future Generation Computer Systems*, 76 :262–274, 2017.

[60] Junqin Xu and Jihui Zhang. Exploration-exploitation tradeoffs in metaheuristics : Survey and analysis. In *Proceedings of the 33rd Chinese Control Conference*, pages 8633–8638. IEEE, 2014.

[61] Quanwang Wu and Qingsheng Zhu. Transactional and QoS-aware dynamic service composition based on ant colony optimization. *Future Generation Computer Systems*, 29(5) :1112–1119, 2013.

[62] Godar J Ibrahim, Tarik A Rashid, and Mobayode O Akinsolu. An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment. *Journal of Parallel and Distributed Computing*, 143 :77–87, 2020.

[63] P Thangaraj and P Balasubramanie. Meta heuristic QoS based service composition for service computing. *Journal of Ambient Intelligence and Humanized Computing*, 12(5) :5619–5625, 2021.

[64] Qiang Yu, Ling Chen, and Bin Li. Ant colony optimization applied to web service compositions in cloud computing. *Computers & Electrical Engineering*, 41 :18–27, 2015.

[65] Hashem Alayed, Fadl Dahan, Taha Alfakih, Hassan Mathkour, and Mohammed Arafah. Enhancement of ant colony optimization for QoS-aware web service selection. *IEEE Access*, 7 :97041–97051, 2019.

[66] Hong Xia, Yan Chen, Zengzhi Li, Haichang Gao, and Yanping Chen. Web service selection algorithm based on particle swarm optimization. In *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 467–472. IEEE, 2009.

[67] Wenbin Wang, Qibo Sun, Xinchao Zhao, and Fangchun Yang. An improved particle swarm optimization algorithm for QoS-aware web service selection in service oriented communication. *International Journal of Computational Intelligence Systems*, 3(sup01) :18–30, 2010.

[68] Simone A Ludwig. Applying particle swarm optimization to quality-of-service-driven web service composition. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 613–620. IEEE, 2012.

[69] Alexandre Sawczuk da Silva, Hui Ma, and Mengjie Zhang. A graph-based particle swarm optimisation approach to QoS-aware web service composition and selection. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 3127–3134. IEEE, 2014.

[70] Xiaofei Xu, Quan Z Sheng, Zhongjie Wang, Lina Yao, et al. Novel artificial bee colony algorithms for QoS-aware service selection. *IEEE Transactions on Services Computing*, 12(2) :247–261, 2016.

[71] N Arunachalam and A Amuthan. Improved cosine similarity-based artificial bee colony optimization scheme for reactive and dynamic service composition. *Journal of King Saud University-Computer and Information Sciences*, 32(10) :1218, 2018.

[72] Hamza Yapıcı and Nurettin Çetinkaya. An improved particle swarm optimization algorithm using eagle strategy for power loss minimization. *Mathematical Problems in Engineering*, 2017, 2017.

[73] Bernardo Morales-Castañeda, Daniel Zaldivar, Erik Cuevas, Fernando Fausto, and Alma Rodríguez. A better balance in metaheuristic algorithms : Does it exist ? *Swarm and Evolutionary Computation*, 54 :100671, 2020.

[74] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. Privacy-aware cloud service composition based on QoS optimization in Internet of Things. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–26, 2020.

[75] Mina Emami Khansari, Saeed Sharifian, and Seyed Ahmad Motamedi. Virtual sensor as a service : a new multicriteria QoS-aware cloud service composition for IoT applications. *The Journal of Supercomputing*, 74(10) :5485–5512, 2018.

[76] Zhangbing Zhou, Deng Zhao, Lu Liu, and Patrick CK Hung. Energy-aware composition for wireless sensor networks as a service. *Future Generation Computer Systems*, 80 :299–310, 2018.

[77] KyeongDeok Baek and In-Young Ko. Dynamic and effect-driven output service selection for IoT environments using deep reinforcement learning. *IEEE Internet of Things Journal*, 10(4) :3339–3355, 2022.

[78] Marzieh Hamzei, Saeed Khandagh, and Nima Jafari Navimipour. A quality-of-service-aware service composition method in the Internet of Things using a multi-objective fuzzy-based hybrid algorithm. *Sensors*, 23(16) :7233, 2023.

[79] Heba Kurdi, Fadwa Ezzat, Lina Altoaimy, Syed Hassan Ahmed, and Kamal Youcef-Toumi. Multicuckoo : multi-cloud service composition using a cuckoo-inspired algorithm for the Internet of Things applications. *IEEE Access*, 6 :56737–56749, 2018.

[80] Ahmed Zebouchi and Youcef Aklouf. pRTMNSGA-III : a novel multi-objective algorithm for QoS-aware multi-cloud IoT service selection. *Annals of Telecommunications*, pages 1–22, 2024.

[81] Mehdi Hosseinzadeh, Quan Thanh Tho, Saqib Ali, Amir Masoud Rahmani, Alireza Souri, Monire Norouzi, and Bao Huynh. A hybrid service selection and composition model for cloud-edge computing in the Internet of Things. *IEEE Access*, 8 :85939–85949, 2020.

[82] Zhengyi Tang, Yongbing Wu, Jinshui Wang, and Tianwei Ma. IoT service composition based on improved shuffled frog leaping algorithm. *Heliyon*, 10(7), 2024.

[83] Neeti Kashyap, A Charan Kumari, and Rita Chhikara. Multi-objective optimization using NSGA II for service composition in IoT. *Procedia Computer Science*, 167 :1928–1933, 2020.

[84] Asma Cherifi, Mohamed Essaid Khanouche, Yacine Amirat, and Zoubeyr Farah. A parallel approach for user-centered QoS-aware services composition in the Internet of Things. *Engineering Applications of Artificial Intelligence*, 123 :106277, 2023.

[85] Zainab H Ali and Hesham A Ali. Towards sustainable smart IoT applications architectural elements and design : opportunities, challenges, and open directions. *The Journal of Supercomputing*, 77 :5668–5725, 2021.

[86] Bahzad Charbuty and Adnan Abdulazeez. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, 2(01) :20–28, 2021.

[87] Malika Yaici, Salima Sabri, Wissam Azni, and Faiza Boudjemil. Contextual data classification for a ubiquitous intelligent environment. *SN Applied Sciences*, 2(4) :553, 2020.

[88] Alain Girard. *Exploration d'un algorithme génétique et d'un arbre de décision à des fins de catégorisation.* PhD thesis, Université du Québec à Trois-Rivières, 2007.

[89] Hui Li, Peng Zou, Zhiguo Huang, Chenbo Zeng, and Xiao Liu. Multimodal optimization using whale optimization algorithm enhanced with local search and niching technique. *Mathematical Biosciences and Engineering*, 17(1) :1–27, 2020.

[90] Eyhab Al-Masri and Qusay H Mahmoud. QoS-based discovery and ranking of web services. In *2007 16th international conference on computer communications and networks*, pages 529–534. IEEE, 2007.

[91] Nora Almezeini and Alaaeldin Hafez. Task scheduling in cloud computing using lion optimization algorithm. *International Journal of Advanced Computer Science and Applications*, 8(11), 2017.

[92] Karuppaiah Geetha, Veerasamy Anitha, Mohamed Elhoseny, Shankar Kathiresan, Pourya Shamsolmoali, and Mahmoud M Selim. An evolutionary lion optimization algorithm-based image compression technique for biomedical applications. *Expert Systems*, 38(1) :e12508, 2021.

[93] Sadhan Gope, Subhojit Dawn, Rituparna Mitra, Arup Kr Goswami, and Prashant Kr Tiwari. Transmission congestion relief with integration of photovoltaic power using lion optimization algorithm. In *Soft Computing for Problem Solving : SocProS 2017, Volume 1*, pages 327–338. Springer, 2019.

[94] SS Saranya, Palagati Anusha, S Chandragandhi, O Kiran Kishore, Nakka Phani Kumar, and K Srihari. Enhanced decision-making in healthcare cloud-edge networks using deep reinforcement and lion optimization algorithm. *Biomedical Signal Processing and Control*, 92 :105963, 2024.

[95] Shuiguang Deng, Hongyue Wu, Wei Tan, Zhengzhe Xiang, and Zhaohui Wu. Mobile service selection for composition : An energy consumption perspective. *IEEE Transactions on Automation Science and Engineering*, 14(3) :1478–1490, 2017.

Résumé : L'Internet des objets (IoT) a transformé notre quotidien en connectant des milliards d'appareils, offrant des fonctions spécifiques que l'on appelle services IoT. Pour répondre à des demandes utilisateurs complexes, il est nécessaire de composer plusieurs de ces services. Cependant, la prolifération des services IoT fonctionnellement équivalents, mais présentant des niveaux de qualité de service (QoS) variables, fait du problème de composition de services basé sur la QoS un défi majeur dans le domaine de l'IoT. Cette thèse propose deux nouvelles approches basées sur des méta-heuristiques, pour aborder ce problème. La première combine un arbre de décision (DT) et l'algorithme d'optimisation des baleines (WOA), pour réduire l'espace de recherche et retourner une composition de bonne qualité. La seconde proposition, DALOA, est basée sur l'algorithme d'optimisation des lions et considère les préférences et les contraintes globales de QoS. La validation de ces approches a été effectuée par des scenarios et une série de tests. Parmi ces scénarios une étude de cas médial concernant la prise en charge d'un patient malentendant a permis de prouver la capacité de DALOA à fournir une solution optimisée, adaptée aux besoins du patient et conforme aux exigences de qualité du domaine médical.

Mots-clés : Qualité de Service, Composition de services, Internet des objets, arbre de décision, Algorithme d'optimisation de la baleine, Algorithme d'optimization des lions.

Abstract: The Internet of Things (IoT) has transformed our daily lives by connecting billions of devices, offering specific functions known as IoT services. To meet complex user demands, several IoT services need to be combined. However, the proliferation of IoT services that are functionally equivalent but offer varying levels of quality of service (QoS) makes the problem of QoS-based service composition a major challenge in the field of IoT. This thesis proposes two new meta-heuristic-based approaches to address this problem. The first combines a decision tree (DT) and the whale optimization algorithm (WOA) to reduce the search space and return a high-quality composition. The second proposal, DALOA, is based on the lion optimization algorithm and considers global QoS preferences and constraints. These approaches were validated through scenarios and a series of test cases. Among these scenarios, a medical case study involving the care of a hearing-impaired patient demonstrated DALOA's ability to provide an optimized solution tailored to the patient's needs and compliant with medical quality requirements.

Keywords: Quality of Service, Service Composition, Internet of Things, Decision Tree, Whale Optimization Algorithm, Lion Optimization Algorithm.

ملخص: لقد غيرت تقنية إنترنت الأشياء حياتنا اليومية من خلال ربط مليارات الأجهزة، وتوفير وظائف محددة تسمى خدمات إنترنت الأشياء. ولتلبية متطلبات المستخدمين المعقدة، من الضروري تركيب العديد من هذه الخدمات. ومع ذلك، فإن انتشار خدمات إنترنت الأشياء المتكافئة وظيفيًا، ولكنها تقدم مستويات متفاوتة من جودة الخدمة، يجعل مشكلة تركيب الخدمات على أساس جودة الخدمة تحديًا كبيرًا في مجال إنترنت الأشياء. تقترح هذه الأطروحة نهجين جديدين قائمين على الميتا-الخوارزميات، لمعالجة هذه المشكلة. الأول يجمع بين شجرة القرار وخوارزمية تحسين الحيتان لتقليل مساحة البحث وإرجاع تركيبة جيدة الجودة. أما الاقتراح الثاني، DALOA، فيستند إلى خوارزمية تحسين الأسود ويأخذ في الاعتبار التفضيلات والقيود الشاملة لجودة الخدمة. تم التحقق من صحة هذه المناهج من خلال سيناريوهات وسلسلة من الاختبارات. من بين هذه السيناريوهات، أثبتت دراسة حالة طبية تتعلق برعاية مريض يعاني من ضعف السمع قدرة DALOA على توفير حل محسّن ومناسب لاحتياجات المريض ومتوافق مع متطلبات الجودة في المجال الطبي.

الكلمات المفتاحية: جودة الخدمة، وتكوين الخدمة، وإنترنت الأشياء، وشجرة القرار، وخوارزمية تحسين الحوت، وخوارزمية تحسين الأسد، وخوارزمية تحسين الأسد.