REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU-BEKR BELKAID - TLEMCEN

Laboratoire de Recherche en Informatique de Tlemcen

THÈSE

Présentée à :

FACULTE DES SCIENCES – DEPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

DOCTORAT EN SCIENCES

Spécialité : Informatique

Par :

Mr ABDELHAK ETCHIALI

Sur le thème

Sélection des services web avec prise en compte de la qualité de service dynamique et incertaine

---

Soutenue publiquement le 20 Avril 2024 à Tlemcen devant le jury composé de :

Mr BENMAMMAR Badr	Professeur	Université de Tlemcen	Président
Mr HADJILA Fethallah	Maître de Conférences A	Université de Tlemcen	Directeur de thèse
Mr MERZOUG Mohammed	Maître de Conférences A	Université de Tlemcen	Examinateur
Mr DENNOUNI Nassim	Maître de Conférences A	Ecole Supérieure de Management de Tlemcen	Examinateur
Mr BRAHAMI Mostefa Anwar	Maître de Conférences A	Ecole Supérieure en Sciences Appliquées de Tlemcen	Examinateur

Laboratoire de recherche en informatique de Tlemcen.
BP 119, 13000 Tlemcen - Algérie

Remerciements

Louange à Allah qui m'a donné patience et courage pour mener à bien ce travail de thèse malgré les difficultés rencontrées. Je tiens à remercier en premier lieu Mr Hadjila Fethallah, mon directeur de thèse pour ses conseils précieux.

Je remercie également les membres du jury qui ont accepter l'évaluation de notre travail, et je cite en l'occurence Pr Badr Bemammar, Dr Merzoug Mohammed, Dr Denouni Nassim, et Dr Brahami Mostefa Anwar.

Je remercie aussi ma famille pour son soutien, son écoute et ses encouragements tout au long de cette thèse.

Je dédie cet humble travail
A mes très chers parents, que Dieu tout puissant les protège
A ma trés chère femme, mes enfants et mes fréres
A mes collègues

Résumé

La technologie des services web constitue une implémentation idéale du paradigme du calcul orienté services (SOC). Étant donné que l'objectif principal du SOC est d'assurer l'interopérabilité des applications et la création de compositions d'applications (ou de services) avec valeurs ajoutées, il conviendra de concevoir et de mettre en œuvre des modèles permettant de combiner des services web individuels dans des workflows satisfaisants des critères de performance objectifs. Il convient de noter que les services web courants sont caractérisés par différents attributs de QoS qui jouent un rôle majeur dans la spécification des compositions de services désirés. Il est utile de souligner que les attributs de QoS dépendent largement des fluctuations de l'environnement (par exemple, la surcharge des réseaux, ou la fluctuation des coûts en raison des saisons ou des événements socioculturels) et par conséquent, leur incertitude créera des difficultés supplémentaires dans la modélisation mathématique du problème de composition. Dans cette thèse, nous adressons la composition des services avec incertitude de QoS en proposant deux contributions principales, toutes les deux exploitent une recherche locale et globale pour alléger la complexité temporelle du problème. La première contribution exploite l'heuristique des intervalles majoritaires pour effectuer la recherche locale, en outre, la recherche globale est effectuée à l'aide d'une recherche exhaustive qui exploite les contraintes globales. Dans la deuxième contribution, nous adoptons une version discrète de la méta-heuristique de l'algorithme des chauves-souris (bat algorithm) en plus d'un ensemble d'heuristiques (telles que la dominance floue et la dominance stochastique d'ordre zéro) pour effectuer à la fois la recherche locale et globale. Les résultats obtenus confirment l'efficacité de nos contributions, et en particulier, les performances étaient satisfaisantes pour les workflows qui ont une taille variant entre 2 et 10 composants.

Mots clés : calcul orienté services, services web, composition de services web, qualité de service incertaine, algorithmes des chauves souris, méta-heuristique, dominance floue, dominance stochastique, intervalle majoritaire, recherche exhaustive, services IoT.

Abstract

The web service technology constitutes the golden standard of the service-oriented computing paradigm (SOC). Since the ultimate objective of SOC is how to ensure interoperability of applications and how to create pertinent compositions of services, it will be appropriate to design and implement models for combining individual web services and creating sophisticated workflows. It is worth noting that the actual web services are characterized by different QoS attributes that constitute a major role in specifying the desired service compositions. It should be underlined that the QoS attributes are largely dependent on the environment fluctuations (e.g., network overload), and therefore their inherent uncertainty will create additional difficulties in the mathematical modelling of the problem. In this thesis, we address the service composition with QoS uncertainty by proposing two main contributions; both of them leverage a local and global search to alleviate the complexity of the problem. The first contribution leverages the majority interval heuristic to perform the local search, while the global search is performed using exhaustive search. In the second contribution, we adopt a discrete version of the bat algorithm meta-heuristic, in addition to a set of heuristics (such as fuzzy dominance and zero-order stochastic dominance) for performing both the local and global search. The obtained results confirm the effectiveness of the contributions, especially for workflows with a size ranging from 2 up to 10 components.

Keywords: service oriented computing, web services, web service composition, uncertain quality of service, bat algorithm, meta-heuristics, fuzzy dominance, stochastic dominance, majority interval, exhaustive search, IoT services.

الخلاصة

تعد تقنية خدمات الويب تطبيقًا مثاليًا لنموذج الحوسبة الموجهة نحو الخدمة (SOC). نظرًا لأن الهدف الرئيسي لـ SOC هو ضمان قابلية التشغيل البيني للتطبيقات وإنشاء مجموعات من التطبيقات (أو الخدمات) ذات القيم المضافة، سيكون من الضروري تصميم وتنفيذ نماذج تسمح بدمج خدمات الويب الفردية في سير العمل بما يحقق الهدف معايير الأداء. تجدر الإشارة إلى أن خدمات الويب المشتركة تتميز بسمات جودة الخدمة المختلفة التي تلعب دورًا رئيسيًا في تحديد تركيبات الخدمة المطلوبة. سيكون من المفيد التأكيد على أن سمات جودة الخدمة تعتمد إلى حد كبير على التقلبات البيئية (مثل الحمل الزائد على الشبكة، أو تقلب التكلفة بسبب المواسم أو الأحداث الاجتماعية والثقافية)، وبالتالي فإن عدم اليقين المتأصل فيها سيخلق صعوبات إضافية في النمذجة الرياضية لمشكلة التكوين. في هذه الأطروحة، نتناول تكوين الخدمة مع عدم اليقين في جودة الخدمة من خلال اقتراح مساهمتين رئيسيتين، كلاهما يستغلان البحث المحلي والعالمي للتخفيف من التعقيد الزمني للمشكلة. تستغل المساهمة الأولى إرشادية الفاصل للأغلبية لإجراء البحث المحلي، علاوة على ذلك، يتم إجراء البحث الشامل باستخدام بحث شامل يستغل القيود الكلية. في المساهمة الثانية، نعتمد نسخة منفصلة من الاستدلال الفوقي لخوارزمية الخفافيش بالإضافة إلى مجموعة من الاستدلالات (مثل الهيمنة الغامضة والهيمنة العشوائية ذات الترتيب الصفري.) لإجراء البحث المحلي والكلي. تؤكد النتائج التي تم الحصول عليها فعالية مساهماتنا، وعلى وجه الخصوص، كان الأداء مرضيا بالنسبة لسير العمل الذي يتراوح حجمه بين 2 و 10 مكونات.

الكلمات الرئيسية : الحوسبة الموجهة نحو الخدمة، خدمات الويب، تركيب خدمات الويب، جودة الخدمة غير المؤكدة، خوارزميات الخفافيش، الفوقية الإرشادية,الهيمنة الغامضة، الهيمنة العشوائية، فاصل الأغلبية، البحث الشامل، خدمات إنترنت الأشياء.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte

Le calcul orienté service est un paradigme qui offre un ensemble de principes (tels que le couplage faible, la réutilisabilité, la composabilité des applications/services) pour batir une application distribuée [Bieberstein, 2006]. Ces mêmes principes peuvent aussi être concrétisés et raffinés avec l'architecture orientée service (SOA). Cette dernière ajoute des principes tels que le respect des formats d'échange standardisés , la transparence par rapport à la localisation, l'autonomie, et l'encapsulation. SOA définit aussi un cycle de vie basique pour le concept de service (publication, recherche, liaison, invocation) en plus d'un ensemble d'intervenants dans ce cycle, tels que le fournisseur de services, l'annuaire des services et le consommateur des services (voir la figure 1.1).

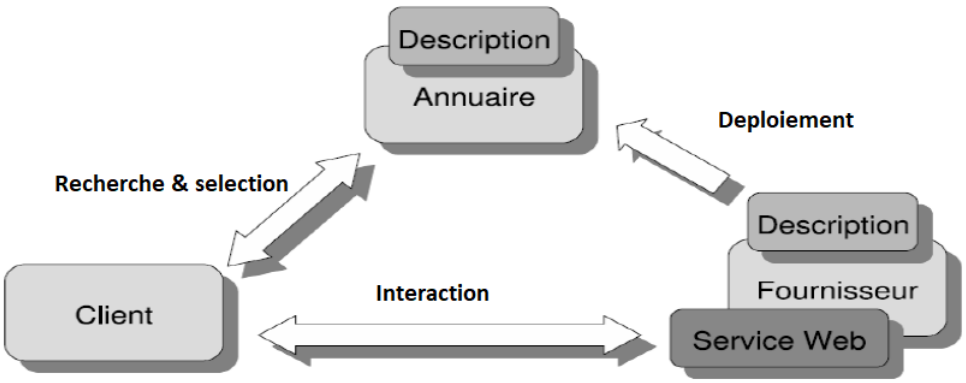

Figure 1.1 – *Les intervenants dans l'architecture orientée service [Papazoglou et al., 2008].*

Dans le contexte de cette architecture, les services web apparaissent comme une technologie idéale qui concrétise tous les principes cités précédemment. En effet le glossaire des services web du W3C : définit cette technologie comme suit :“ A web

service is a software system designed to support interopérable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards". De même IBM definit les services web comme suit : " Web services are a new breed of Web applications. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web".

À partir de ces deux définitions il s'avère que les services web sont des modules logiciels faiblement couplés (l'application cliente n'est pas obligée de suivre les choix technologiques adoptés par le service- en termes de systèmes d'exploitation, architecture matérielle, libraires et langage de programmation) qui utilisent les standards du web (XML, HTTP) pour communiquer entre eux. Il est aussi possible de créer des applications ayant une valeur ajoutée en composant plusieurs services web existants. L'amélioration des performances des services Web est due au fait que les interactions entre les services modulaires sont réalisées sans connaissance préalable des détails de l'implémentation interne. En plus l'adoption du style architectural REST (Representational state transfer) créera des services encore plus flexibles et plus performants en termes de temps, en raison d'utilisation des principes du caching et la diversification des formats textuels des données échangés.

Etant donné la prolifération rapide de cette technologie de services web, les entreprises seront invitées à développer des algorithmes et des solutions pour assurer l'intégration et l'échange de données entre les applications (services) appartenant à la même entreprise (Enterprise Application Integration ou EAI) et même les applications appartenant à des entreprises différentes (nous parlons alors de business to business ou B2B). En outre, les entreprises seront aussi invités à développer des services web pour faciliter à leurs clients l'accès aux produits et aux utilités désirées (nous parlons alors de business-to-consumer ou B2C). Les modèles de communication cités précédemment (EAI, B2B et B2C) réduisent le coût et le temps nécessaire au développement d'une application web fiable et agile.

L'internet des objets (IOT) est aussi un domaine important dans lequel l'architecture SOA peut être appliquée. Dans cette optique, les service IoT peuvent implémentés et déployées dans les villes intelligentes (smart cities) pour assurer des services aux utilisateurs tels que la recherche et la réservation des places de parking (les plus proches par rapport à la position du client), la recherche, la mesure du taux d'encombrement et/ou des degrés de pollution de certaine routes de la ville, la recherche des stations d'énergies les plus proches au client, la réduction de la pollution et de l'encombrement du trafic en ajustant les durées des feux tricolores, etc. La majorité de ces besoins peuvent être assurés en composant les bons services IoT, et pour cela, il sera judicieux de développer des modèles et des algorithmes efficaces pour créer et gérer les compositions de services.

Dans le contexte de l'architecture SOA, il s'avère que les services SOAP (simple object access protocol) du W3C (développés avec des librairies telles que Axis [1] ou ASP.NET [2]) et les services REST (restful services) sont les meilleurs choix pour implémenter les logiques d'affaires des entreprises. Avec ces deux variantes (qui sont peu coûteuses) les decideurs (chef d'entreprises, gouvernements) peuvent etablir des

1. https ://axis.apache.org/
2. https ://dotnet.microsoft.com/en-us/apps/aspnet

services complexes qui procurent une valeur ajoutée à la société.

1.2 Problématique et motivations

Actuellement, Il convient de noter que beaucoup d'entreprises implémentent des services web ayant des fonctionnalités équivalentes, tout en offrant des niveaux de QoS différents (cad, il est probable que l'on rencontre une même fonctionnalité implémentée avec des aspects ou propriétés non fonctionnelles différentes). En plus, dans les cas pratiques, les clients ont besoin d'applications complexes (ou exigence complexe) qui sont rarement satisfaites par un seul service élémentaire. Pour répondre à ce besoin, il sera judicieux de développer des procédures de composition de services web. En composant les services web (atomiques/complexes) nécessaires, nous créerons une application ayant valeur ajoutée, et cette dernière peut assurer tous besoins requis par l'utilisateur final, et éventuellement même des propriétés émergentes. La figure 1.2 illustre une requête d'un utilisateur pour rédiger un article de recherche tout en assurant des niveaux de QoS adéquats. En particulier, le client a besoin d'un éditeur pour rédiger le contenu textuel de son papier de recherche ; en plus il a besoin d'un éditeur de diagrammes et courbes(avec des QoS spécifiques telles que la haute résolution et/ou la diversification des formats graphiques, possibilité de créer les diagrammes avec des formats vectoriels) ; il a aussi besoin d'éditeur de références bibliographiques ; enfin l'utilisateur a aussi besoin d'un service de correction de la langue (language editing service) pour bien présenter la forme de son papier de recherche.

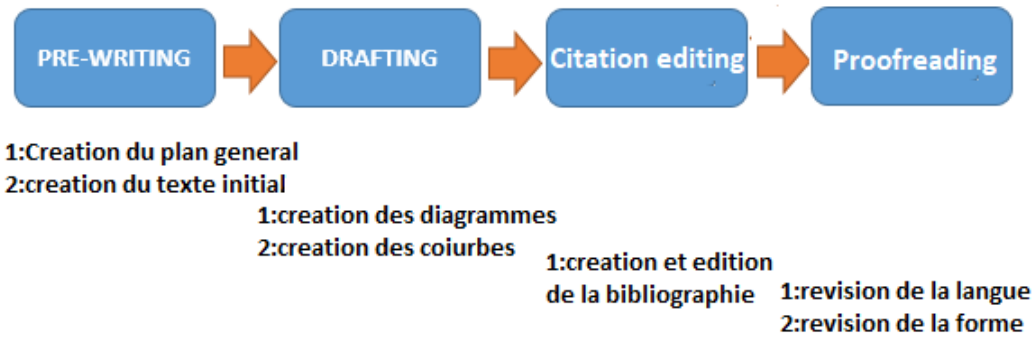

Figure 1.2 – *Besoin complexe d'un client.*

Il est clair que cette requête ne peut pas être satisfaite par un seul service, et de même, il est évident que chaque tâche de l'ensemble cité précédemment peut être réalisée par plusieurs services web (offerts par plusieurs entreprises) avec des critères de QoS différents. Le client recherchera les meilleurs services de chaque tâche (en termes de QoS), tout en satisfaisant ses contraintes globales telles que le respect de son budget monétaire/ ou respect d'un niveau de réputation minimal prédéfini (voir la figure 1.3). Le client a donc besoin de modèles et d'algorithmes pour bien gérer cette situation de recherche/composition.

En général, le nombre croissant de services Web disponibles sur le Web a des impacts positifs et négatifs sur le processus de composition. En ce qui concerne les impacts positifs, nous pouvons noter que cette prolifération assurera plus d'alternatives pour une même fonctionnalité donnée et, par conséquent, plus d'avantages pour

l'utilisateur. Mais d'autre part, le nombre élevés de services offerts par classe augmente de façon exponentielle le nombre de compositions possibles, et de même, le temps de recherche de ces compositions de service augmente et il dépassera les délais tolérés par les décideurs ou clients finaux. Selon certains travaux de la littérature ([Alrifai et al., 2012, Dahan et al., 2019, Benouaret et al., 2012a]) la qualité de service (telle que le coût, temps d'exécution, fiabilité, disponibilité, débit, etc.) est considérées comme statique et elle ne change pas en fonction du temps. Malheureusement cette hypothèse est rarement valide, et nous observons presque toujours que la QoS est fluctuante, non déterministe et dynamique. Par exemple, le coût de la réservation d'une chambre d'hôtel ou d'un billet d'avion est incertain et dépend de la période (comme la saison ou le mois), des événements sociaux, et d'autres aspects contextuels ; le temps d'exécution d'une application distribuée dépend de la charge du réseau et les heures de pointes. Par conséquent, l'incertitude de la qualité de service changera la manière de mesurer la pertinence d'une composition de services ainsi que les algorithmes de sélection/composition [Hadjila et al., 2020]. Pour répondre aux lacunes existantes dans les approches classiques de composition de services (qui considère que la QoS est certaine), nous considérons dans cette thèse que la QoS est incertaine, et que même la loi de probabilité suivie est inconnue. De ce fait, il sera judicieux de proposer des algorithmes et modèles qui tiennent compte de cette incertitude pour évaluer la pertinence d'un service individuel et même les compositions de services.

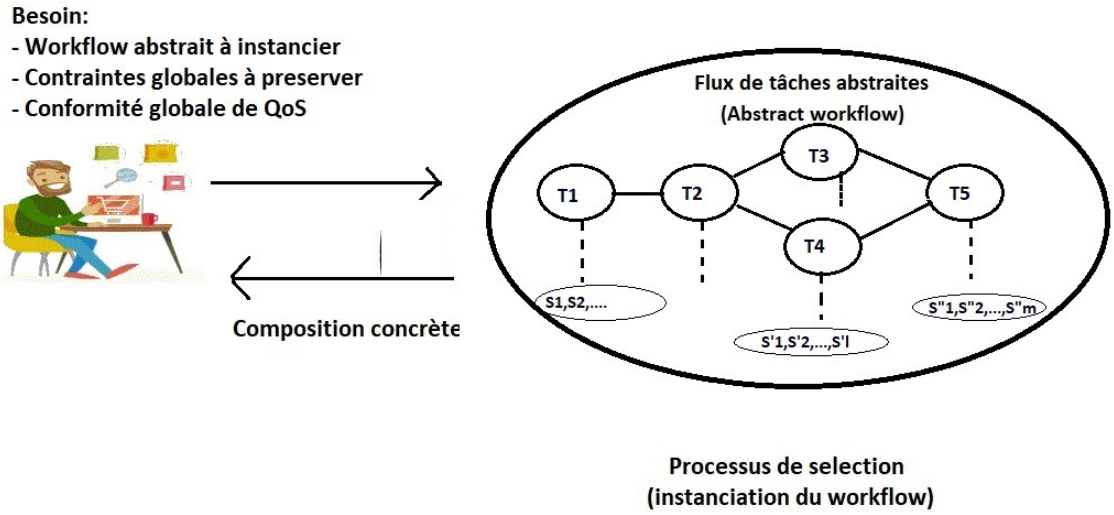

Figure 1.3 – *Workflow abstrait.*

Pour traiter cette problématique de composition avec QoS incertaine, nous supposons que la requête de l'utilisateur est modélisée comme un flux de tâches abstraites (abstract workflow) qui doivent être instanciées avec des services concrets (voir la figure 1.3).

La combinaison de ces services concrets doit répondre à un ensemble d'exigences de qualité de service qui expriment les limites supérieures/inférieures des critères de qualité de service (par exemple, le coût total des services concrets est inférieur au budget de l'utilisateur). Nous admettons que les valeurs des attributs de QoS de chaque service fluctuent en fonction du temps/l'espace ou un autre critère contextuel. En résumé, notre tâche consiste à rechercher les compositions de services Top-k qui répondent le mieux aux exigences de l'utilisateur en matière de qualité de service. la

satisfaction des exigences de qualité de service (également appelée conformité globale de la QoS dans le chapitre 5) se traduit en un processus de maximisation de la probabilité de préserver les contraintes globales. Cette fonction objectif tient compte de la fluctuation des variables de QoS, ainsi que les contraintes globales exprimées par les utilisateurs finaux.

1.3 Contribution

Nous notons que la majorité des travaux adressant la composition de services assume que la QoS est statique, et par conséquent la fonction objectif qui en résulte peut être optimisée avec des méthodes exactes telles que la programmation entière [Ardagna and Pernici, 2007], ou des heuristiques ou meta-heuristiques [Fethallah et al., 2012, Halfaoui et al., 2018]. Dans notre thèse, nous supposons que la QoS est incertaine, et par conséquent, les échantillons des variables des attributs de QoS mesurés sont pris en compte dans les algorithmes de sélection et de compositions de services. Pour adresser la grande taille de l'espace de recherche des compositions, nous proposons une procédure de composition à deux étapes : une recherche locale (pour faire une pré-sélection des services appartenant à une tâche donnée) et une recherche globale (pour trouver une composition de services qui maximise la fonction objectif nommée conformité globale de la QoS (GQC) mentionée dans l'equation 5.1). Ces deux grandes étapes sont illustrées dans la figure 1.4. La première étape permet de

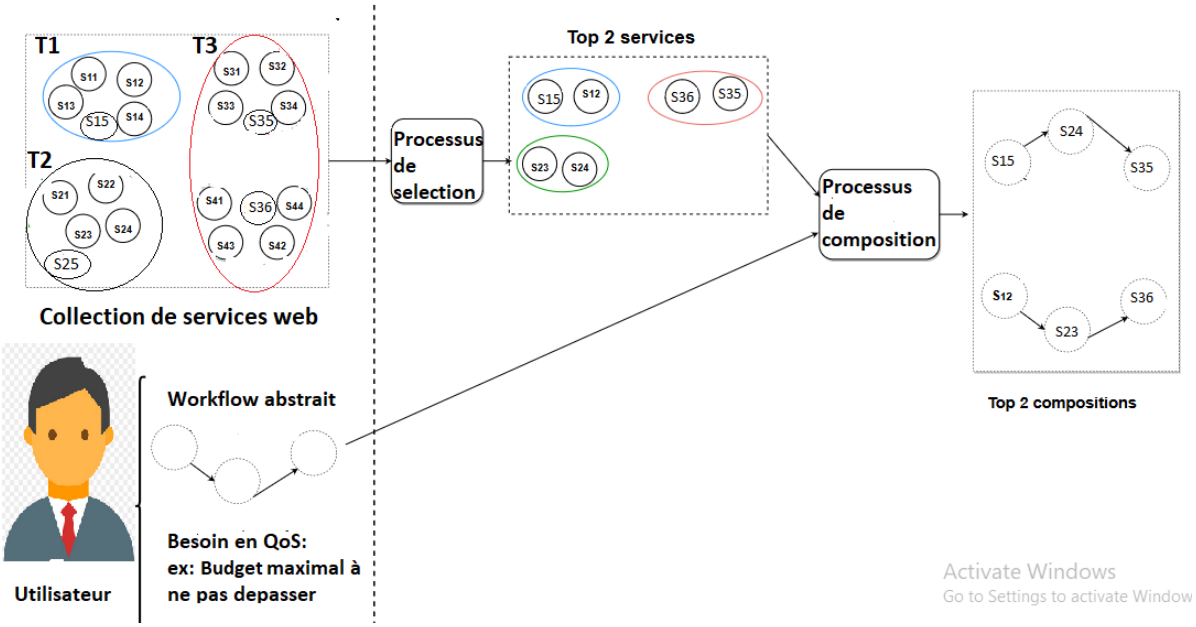

Figure 1.4 – *Processus de séléction à deux phases.*

choisir les services appropriés pour chaque classe abstraite du flux de tâches (workflow abstrait) à l'aide d'une heuristique donnée. L'intuition de base derrière cette présélection est la réduction de l'espace de recherche en filtrant les services les moins pertinents. Notons aussi que cette recherche heuristique assure un compromis entre la pertinence (en QoS) des services retenus et le temps de calcul.

D'autre part, la deuxième étape permet de retenir les meilleures compositions qui optimisent une fonction objectif nommée conformité globale de la QoS (GQC). Les

meilleures compositions qui optimisent les exigences de qualité de service (la fonction objectif GQC mentionnée précédemment) ; pour ce faire, nous effectuons une autre recherche heuristique qui sélectionne les compositions qui satisfont les contraintes globales au sens de la fonction objectif GQC. Dans ce qui suit nous détaillons nos trois contributions :

1. *La première contribution [Abdelhak et al., 2019] :* cette approche est composée de deux étapes : La première étape se base sur l'heuristique d'intervalles majoritaires pour comparer et trier les services de chaque tâche par ordre de mérite (la QoS). Cette étape retourne les Top-K services pertinent en termes d'heuristique d'intervalles majoritaires. La deuxième étape adopte la recherche exhaustive (ou recherche avec retour arrière) pour sélectionner les compositions qui satisfont les contraintes globales et maximisent au mieux la conformité globale de QoS.
2. *La deuxième contribution [Etchiali et al., 2023] :* cette approche est basée aussi sur les deux étapes précédentes (recherche locale et globale) : La première étape (recherche locale des Top-k services) est implementée à l'aide de 03 heuristiques :
   - L'heuristique de dominance floue (implementée avec les fonctions minimum et maximum).
   - L'heuristique de dominance stochastique d'ordre 0.
   - L'heuristique de dominance stochastique d'ordre 1.

   La deuxième étape adopte la meta-heuristique d'algorithme des chauves-souris (ou Bat algorithm) pour sélectionner les compositions qui maximisent au mieux la conformité globale de QoS GQC. Cet algorithme est choisi en raison de sa capacité de mixer les opérateurs globaux (qui s'inspirent de la décente de gradient) et les opérateurs de recherche locale (que nous retrouvons dans des algorithmes tels que Hill Climbing). Les expérimentations montrent que la combinaison de cette méta-heuristique avec les heuristiques de dominance floue ou de dominance stochastique d'ordre 0 ont donné des performances impressionnantes par rapport aux résultats obtenus en état de l'art.
3. *La troisième contribution [Abdelhak et al., 2021] :* cette approche adresse la sélection des services IoT tout en utilisant les attributs contextuels et les attributs de QoS. Plus précisément, la première phase de cette approche sélectionne les services IoT spécifiques à une tâche donnée en utilisant les attributs de couverture temporelle et spatiale. Cela veut dire que si la fenêtre spatiale et temporelle de la requête du client correspond (totalement ou partiellement) avec la fenêtre de couverture spatiale et temporelle du service IoT, alors le service est retenu. Dans la deuxième phase, les services retenus dans l'étape de recherche contextuelle sont filtrés grâce aux critères de QoS. Ces critères de QoS non-déterministes sont comparés grâce à la dominance stochastique d'ordre 0.

1.4 Organisation du document

Cette thèse est constituée de sept chapitres qui couvrent la specification du problème ainsi que les solutions apportées. Nous notons que les chapitres 2, 3, 4 couvrent la partie thèorique ainsi que l'état d l'art de la composition de services. En contre partie, les chapitres 5 et 6 couvrent les contributions et la partie pratique de la thèse. Dans ce qui suit nous donnons un survol sur le contenu de chaque chapitre.

- **Chapitre 2 : Généralités sur les services Web**

  Dans ce chapitre, nous présentons les principes de l'architecture SOA, ainsi que les différents standards relatifs aux services web. Nous presentons aussi les deux variantes des services web et en particulier les services SOAP et les services REST.

- **Chapitre 3 : État de l'art sur la composition de services web**

  Le troisième chapitre présente les travaux qui traitent la composition de façon générale (en tenant compte des attributs fonctionnels et non-fonctionnels). Il présente aussi un aperçu sur les méthodes mono-objectives et multiobjectives qui adressent la composition des services web.

- **Chapitre 4 : Sélection de services web avec QoS incertaine**

  Ce chapitre présente les travaux connexes qui tiennent compte de la QoS non déterministe ou incertaine. Il donne un survol sur des méthodes qui adressent le problème de la sélection avec QoS incertaine (méthodes de recherche globale, locale, approximative et exacte).

- **Chapitre 5 : Approches proposées pour la composition et sélection de services**

  Ce chapitre présente les approches proposées pour résoudre le problème de la composition des services avec prise en compte de la QoS incertaine. Il présente la spécification formelle du problème, ainsi que les algorithmes et les heuristiques développées pour trouver les Top-K compositions optimales en termes de conformité globale de QoS.

- **Chapitre 6 : Implémentation et expérimentation**

  Ce chapitre couvre l'évaluation des approches proposées tout en décrivant les collections de données. Nous détaillons aussi les raisons de succès ou échec de chaque heuristique.

- **Chapitre 7 : Conclusion et perspectives**

  Le dernier chapitre résume les principales contributions de la thèse et les objectifs atteints. Il discute aussi les futures pistes pour adresser la sélection et composition de services web traditionnels (software as a service ou SaaS) et même IoT ou IaaS (infrastructure as a service).

Chapitre 2

Généralités sur les services web

Sommaire

2.1 Introduction

Le paradigme émergeant de l'Informatique Orientée Services (SOA – Service Oriented Architecture) a donné naissance à un nouveau mode de développement logiciel pour les systèmes d'information distribués. En effet, il permet de créer des systèmes d'information intra-entreprises et inter-entreprises de manière relativement aisée et en assemblant des services existants dans des workflows plus sphistiqués. SOA

possède plusieurs implémentations ; dans notre travail nous nous intéressons uniquement aux services web, qui représentent la concrétisation la plus répandue sur le réseau internet. Dans ce chapitre, nous décrivons les concepts de base du paradigme orienté service en mettant en évidence la définition, les acteurs, l'architecture et les technologies des services web. Nous terminons l'étude en donnant une vue générale de la composition des services web et son cycle de vie.

2.2 Généralités sur le paradigme orienté service

2.2.1 Évolution vers les méthodes orientées composants et services

Le domaine d'ingénierie du logiciel a connu plusieurs types de d'approches (paradigmes) pour le développement et le déploiment de logiciels. Chaque paradigme créé a ajouté une couche ou une solution pour pallier aux lacunes relatives aux paradigmes précédents. Plus précisement, l'evolution de développement du logiciel a passé par les approches orientées objets [Taylor, 1997], les approches orientées composants [Szyperski et al., 1999] et dernièrement les approches orientées services.

Pour adresser la complexité d'intégration et d'interopérabilité d'applications hétérogènes, plusieurs architectures à base de composants ont été proposées : [Vinoski, 2004], [Schantz and Schmidt, 2001] : EJB (Enterprise Java Beans), CORBA (Common Object Request Broker Architecture), et .NET. Mais, le couplage entre les objets était assez élevé dans ces technologies à base de composants distribués. Par exemple, le standard CORBA qui permet l'accés à des objets distants avec n'importe quel langage, impose aussi la connaissance de la structure des objets par les applications clientes et par leur systèmes fournisseurs. Ceci impose aux différents partenaires la spécification préalable des règles de transformation objets/ messages. En outre, nous ne pouvons faire interopérer que des objets CORBA (ou COM), puisque chaque architecture utilise sa propre infrastructure. En conséquence, la mise en œuvre de ces technologies dans un réseau ouvert tel qu'internet releve beaucoup de défis. Pour pallier aux lacunes de toutes ces architectures, les efforts de la communauté de recherche a donné naissance au concept d'architecture orientée service (Service Oriented Architecture ou SOA) [Papazoglou et al., 2008].

Le paradigme orienté services (Service-Oriented Computing) est un modèle de développement d'applications distribuées sur des réseaux à grande échelle. Sa conception a introduit une nouvelle manière de concevoir, de déployer, d'intégrer, et de manipuler les logiciels. Les approches orientées services [Huhns and Singh, 2005] favorisent le concept de composition de services indépendants pour batir des applications logicielles dynamiques et faiblement couplées. Ces approches considèrent le concept de service comme élément clé qui permet le développement d'applications complexes. L'idée de base est de réécrire les fonctionnalités offertes par les entreprises sous forme de services. Par conséquent, la gestion des applications à base de services adresse davantage le niveau fonctionnel au lieu du niveau implémentation ou technique des applications.

L'une des forces majeures du paradigme orienté service est "le couplage faible" entre le service et l'application cliente. De même, le couplage peut exister aussi entre les différents services de la même application complexe. De façon générale, le couplage faible indique que le consommateur d'un service n'a pas besoin de connaitre les

détails et informations techniques du service (et vice versa) tels que la technologie de réalisation, la plateforme d'exécution d'un service [Dustdar and Papazoglou, 2008], et les librairies utilisées. Ce faible couplage permet la création d'applications de façon plus systématique, et ceci entraîne une atténuation des coûts de développement, de déploiement, d'interopération et d'intégration d'applications [Fki, 2015].

2.2.2 Les services

Le concept de service représente un élément clé dans l'architecture orientée services. Bien qu'il sera difficile de le définir formellement, nous pouvons dire qu'il représente une abstraction des fonctionnalités d'une entité logicielle. Ce service a aussi des caractéristiques non-fonctionnelles comme sa performance, son coût ou sa disponibilité. Le cervice est décrit et publié, et ensuite découvert par d'autres entités (logiciels ou utilisateurs) qui peuvent l'exécuter ou le composer avec d'autres entités.

Nous survolons maintenant certaines définitions présentes dans l'état de l'art. Selon [Dustdar and Papazoglou, 2008], *« Services are self-contained processes deployed over standard middleware platforms, e.g., J2EE, or .NET that can be described, published, located (discovered), and invoked over a network... Services are most often built in a way that is independent of the context in which they are used. This means that the service provider and the consumers are loosely coupled. »*. Cette définition assume que le service est une entité autonome qui est décrite, e, découverte et invoquée à travers un réseau. Le service est conceptuellement indépendant de la technologie sous-jacente. Ceci favorise l'implementation du service avec des technologies variées et des infrastructure de déploiment hétérogènes.

OASIS, le consortium international qui travaille sur le développement, la convergence et l'adoption de standards pour les applications informatiques, propose la définition suivante : *« A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description... A service is accessed by means of a service interface where the interface comprises the specifics of how to access the underlying capabilities. »* [MacKenzie et al., 2006]. Cette définition spécifie le service comme un moyen permettant l'accès à plusieurs fonctionnalités et dont l'accès est donné grâce à une interface précise et connue. Cette interface montre les fonctionnalités offertes par le service et un ensemble de restrictions et même des mécanismes d'accès aux fonctionnalités offertes.

[Han et al., 2009] spécifie un service comme une abstraction d'une logique métier implémentant les processus métiers. Selon [Cauvet and Guzelian, 2008] , "le service est vu comme une unité réutilisable qui englobe un ou plusieurs fragments d'un processus métier et vise à satisfaire des buts métiers". A partir de ces définitions, nous pouvons consolider les idées suivantes :

- Le service permet d'établir et exposer une ou plusieurs fonctionnalités décrites avec des formats spécifiques.
- Ce service peut etre recherché, sélectionné, invoqué grâce à une déscription normalisée.

.

2.2.3 Architecture orientée services (SOA)

Dans ce qui suit, nous listons les principes [Huhns and Singh, 2005], [Erl, 2008] les plus importants du paradigme SOA.

- **Couplage faible :** En général, le couplage mesure la quantité d'échange et la dépendance entre deux systèmes. Si cet échange ou même la dépendance est réduit, alors le couplage est faible. Dans le cadre su SOA, le couplage faible implique aussi l'indépendance totale de l'application cliente et l'application services en termes des choix technologiques utilisés, tels que le langage de programmation, la plateforme, les librairies de programmation et les architectures matérielles. Notons aussi que les services web permettent la coopération d'applications tout en garantissant un faible taux de couplage, car la communication avec les services web est réalisée via des messages décrits par le standard XML caractérisé par sa pérvasivité et son haut niveau d'abstraction. Par conséquent, une modification d'un service peut être effectuée sans altérer sa compatibilité avec les autres services de l'application.
- **Interopérabilité :** L'interopérabilité est la capacité de deux systèmes à échanger les données en dépit des hétérogenietés matérielles et logicielles. Dans le contexte des services web, l'intéropérabilité est assurée avec un haut degré, puisque l'emploi des formats standards tels que XML, et les protocoles d'internets (par exemple HTTP), permet un échange uniforme entre des applications clientes et serveurs (et ceci indépendamment de leurs choix technologiques).
- **Réutilisabilité :** Le principe de réutilisation permet la réduction des coûts de développement des logiciels (car nous ré-exploitons des composants déjà existants). Dans le contexte des services web, ce principe est satisfait grâce à l'emploi des descriptions standards qui exposent les fonctionalités à réutiliser.
- **Découverte :** C'est une phase qui permet le matching entre les interfaces des services web et ceux du besoin du client (entrées, sorties, pré-conditions, post-conditions, etc.) [Hadjila et al., 2019]. Grace à la découverte des services, la réutilisation peut réalisée, en assurant un minimum d'intervention de l'humain dans ce processus d'interaction.
- **Composition :** La collaboration d'un ensemble de services peut être réalisée en assemblant les élements adéquats, et en tirant profit des principes précédents tels que le mécanisme de découverte, et l'emploi des formats d'échanges standards. Notons que l'architecture SOA permet la création d'applications avec valeurs ajoutées en mettant en oeuvre ce principe de composition.

En plus des principes cités précédemment, L'architecture orientée services ajoute trois acteurs principaux dans la gestion du cycle de vie du service : le fournisseur de service, le client de service, le registre de services [O'sullivan et al., 2002]. Les interactions entre ces trois élements sont résumées avec les primitives suivantes :

- La publication.
- La découverte.
- l'interaction ou l'invocation.

Les services sont implémentés est déployés par des entreprises (nommées fournisseurs). Les fournisseurs de service publient la description des services créés dans un registre prédéfini (i.e. ; étape 1 : la publication). Les clients ou les consommateurs de

services effectuent des recherches pour trouver un service qui répond à leurs spécifications (i.e. ; étape 2 : la découverte). Une fois le service est séléctionné, le client effectue une liaison (via une interface de déscription) avec le service et commence l'interaction (i.e. ; étape 3 : l'invocation). Ce scénario d'interaction est montré dans la Figure 2.1.

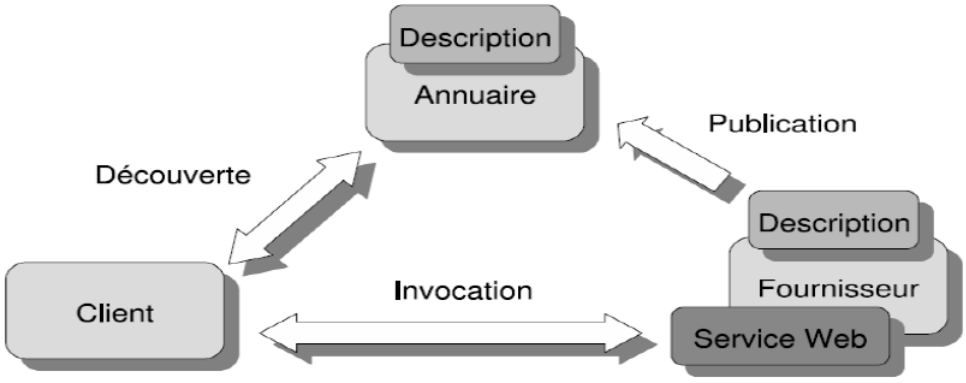

Figure 2.1 – *Le modèle SOA (architecture orientée service)*

Puisque le but de paradigme SOA est d'assurer un accès efficace aux fonctionnalités offertes par les systèmes tiers, il sera évident de consolider toute implémentation de ce paradigme avec un environnement (standards, normes, et protocoles) qui facilite la découverte et la composition de services. Dans ce sens, il sera utile de créer des normes pour bien specifier les aspects fonctionnels des services (celles qui décrivent les principales tâches liées à l'utilisation des services à savoir, la publication, la découverte, la composition, et l'invocation), et les aspects non-fonctionnels (celles qui décrivent les besoins en QoS, les besoins de sécurité, les besoins transactionnels).

2.3 Architecture en couches des services web

Il existe deux niveaux de détails pour specifier l'architecture des services web. La première est une architecture basique (ou plus abstraite) qui permet de décrire les interactions et relations entre les trois acteurs principaux comme celle représentée dans la Figure 2.1. Cette architecture est traditionnellement utilisée pour les services web isolés et reste insuffisante pour permettre une utilisation effective des services web dans les domaines dont les exigences vont au-delà de la capacité d'interactions simples via des protocoles standards. La seconde architecture (la plus détaillé) vient compléter et étendre la première pour prendre en considération des processus plus complexes comme celui de la composition. Elle est constituée de plusieurs couches superposées, d'où son nom. Différentes extensions de l'architecture de base ont été proposées dans la littérature. Le groupe architecture du W3C travaille activement à l'élaboration d'une architecture étendue standard. Nous présentons dans la figure 2.2 une vision globale de cette architecture en couches. Chaque couche de cette architecture s'appuie sur des langages, des protocoles et des modèles, nous distinguons des couches horizontales qui traitent les mécanismes fonctionnels et d'autres verticales pour les mécanismes non fonctionnels. Nous allons détailler dans ce qui suit les couches fonctionnelles ainsi que la couche QoS.

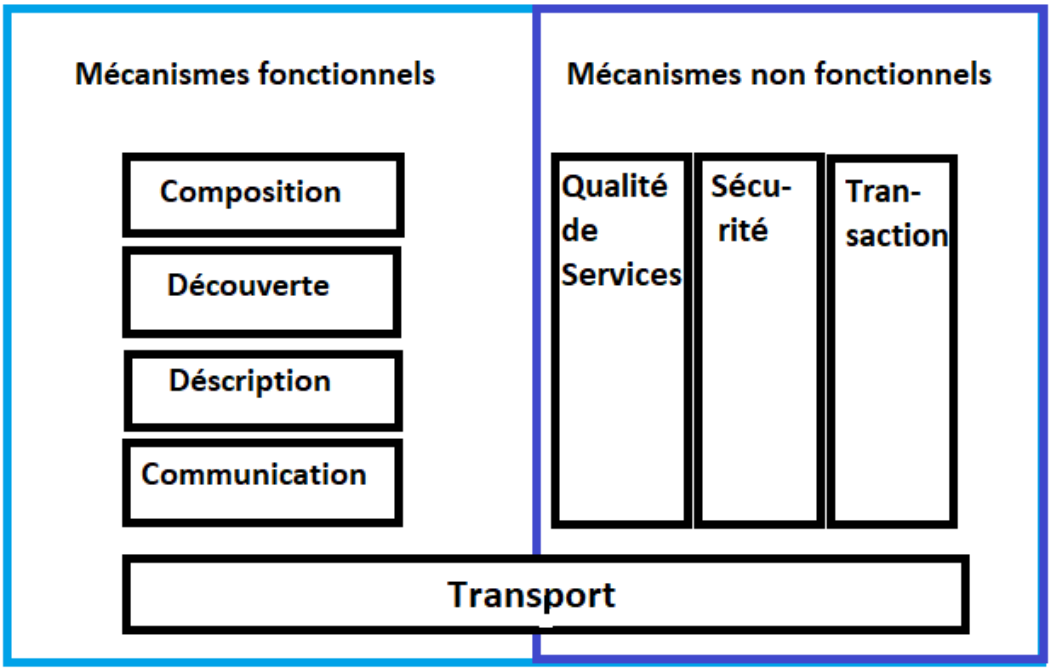

Figure 2.2 – *La pile des couches des Services web [Papazoglou et al., 2008].*

2.3.1 Couche de transport des services

Cette couche regroupe les protocoles de transport de bas niveau, ces derniers servent à transporter les requêtes et les réponses échangées entre services. Le protocole le plus utilisé est http, il est d'ailleurs recommandé par le consortium « Web Service Interoperability»), néanmoins d'autres implémentations peuvent utiliser d'autres protocoles comme HTTPS, SMTP, FTP, etc.

2.3.2 Couche de communication des services

Cette couche spécifie les protocoles d'échanges de documents XML entre le service web et ses clients, elle caractérise aussi le mode d'échange (s'il est bloquant ou non). Nous notons que selon le mode d'invocation, deux styles de services peuvent être distingués REST ou SOAP (SOAP supporte aussi le mode RPC ou DOC). Les services web REST sont actuellement les services les plus répandus dans dans le web (et ceci est dû à leur flexibilité), comme nous le noterons dans la partie comparaison. Tout au long de cette thèse, nous considerions les services independamment de leurs implementations (REST ou SOAP). Nous présentons la premiere implémentation basée sur le protocole SOAP. SOAP est adopté comme un standard pour la messagerie entre les services web.

- **SOAP (Simple Object Acces Protocol)** SOAP [Gudgin et al., 2003] a été développé par Microsoft et IBM. C'est un standard accepté par le W3C en 2000. SOAP est un protocole d'échange et de communication basé sur XML qui permet à un client (navigateur ou programme) d'appeler une operartion à distance. Les services web sont capables d'échanger des informations dans un environnement décentralisé et distribué en utilisant HTTP/SMPT/POP comme

protocole de communication, tout en étant indépendant des plateformes de programmation utilisées. SOAP définit un ensemble de règles pour structurer les messages envoyés. Notons que la spécification du protocole SOAP ne donne aucune indication sur le mécanisme d'accès aux services web. Cette indication est du ressort du langage WSDL qui sera défini dans la section suivante.

SOAP est composé des éléments suivants :

- Entête des protocoles : les entêtes de protocoles (e.g. HTTP, SMTP, etc).
- Enveloppe : il spécifie un cadre général pour exprimer le contenu d'un message. Elle est aussi composée de deux autres éléments imbriqués.
  - Entête : c'est une structure optionnelle. elle représente un mécanisme d'extension qui fournit une manière de passer les informations en messages SOAP qui ne sont pas prises en compte directement par les applications. L'entête peut contenir, par exemple, des informations de sécurité (telles que les signatures électroniques), des informations transactionnelles, des informations de traçabilités, etc.
  - Corps : il représente un élément obligatoire. Il contient les informations principales échangée dans un message SOAP. Il peut s'agir soit du nom de la méthode, avec les données correspondantes, ou d'un simple document XML pour le cas du style DOC. Le corps doit contenir aussi les valeurs de retour ou un message d'erreur pour les messae de réponse.

- **REST (Representational State Transfer)** ce concpet est un style architectural et non pas un protocole. En d'autres termes, c'est un ensemble de principes que nous respecterons pour batir des applications client-serveurs flexibles et performantes en termes de temps et bande passante. L'idée de base est que le service REST expose un ensemble fixé d'operations (interface standardisée) qui sont les verbes du protocole HTTP : GET,POST, PUT, DELETE, HEAD ; ces operations manipulent des resources (présentes sur le serveur) identifiées par un URI. Une ressource peut être une base de donnée, une ligne d'une base de données, une image, une vidéo, une séquence audio, etc (voir l'exemple de la figure 2.3). Après traitement de la requête, le service retourne une représentation de l'état de la resource sous forme d'un contenu écrit en XML, JSON, CSV, ou d'autres formats textes. Les services REST possèdent des avantages de performance (puisque leur charge de données est en moyenne inferieure à celle de l'enveloppe SOAP) et même de flexibilité (puisqu'ils supportent plusieurs formats de données) [Wagh and Thool, 2012]. La table (2.1) présente les principales différénces entre le modèle SOAP et le modèle REST des services web.

Notons que le choix de l'implémentation SOAP ou REST depend de certains paramètres qui déterminent le contexte d'utilisation, par exemple, si le fournisseur du service estime que la sécurité est trés prioritaire que la performance (temps ou bande passante), dans ce cas, le modèle SOAP sera plus appropriés, par contre si la performance est plus prioritaire que d'autres aspects, alors le modèle REST sera plus approprié.

Tableau 2.1 – *Services SOAP vs. REST*

Critère	**SOAP**	**REST**
Protocole de transport	SOAP est indépendant du protocole de bas niveau et peut fonctionner avec n'importe quel protocole de transport (SMTP, FTP, HTTP, etc.)	Travaille uniquement avec HTTP(S)
Format de données	XML	XML, JSON, CSV, HTML, Texte brut
Performance	La taille des messages est assez grande en moyenne (et donc la performance peut etre faible)	La taille des messages est moins grande en moyenne, avec l'adoption du caching , la performance sera amelio-rée
Sécurité	Elle est trés renforcée avec l'adoption de la specifica-tion WS-security	elle est moins forte que celle de SOAP, elle adopte la cryptographie offerte par SSL ou HTTPS
Passage à l'échelle (scalability)	Il est difficile de s'adap-ter à l'échelle, les services SOAP implémentent les ser-vices avec états et sans états	il peut s'adapater avec le passage à l'échelle, les ser-vices REST sont tous sans états

2.3.3 Couche de description des services

Cette couche permet de décrire le service web. Cette description concerne le profil fonctionnel/non fonctionnel du service web. Comme nous l'avons mentionné plus haut, les services web peuvent etre décrits de manière syntaxique ou sémantique ; ils sont aussi soit atomiques ou composites.

La description est liée au type de service qui peut être soit syntaxique ou sémantique. dans ce qui suit, nous expliquons les deux cas :

1. Une description syntaxique qui utilise des structures syntaxiques de type XML pour décrire les propriétés des services web. Elle repose principalement sur le langage WSDL pour décrire la structure du service atomique. Elle repose aussi sur WS*-spécifications pour décrire le comportement au sein d'une composition. Nous citons la recommandation WS-CDL (Web Services Choreography Description Language) [Kavantzas et al., 2005] pour couvrir la description comportementale de la chorégraphie des services. Concernant le comportement du service lié à l'orchestration des services, OASIS (Organization for the Advancement of Structured Information Standards) propose le standard WS-BPEL (Web Services Business Process Execution Language) [Alves et al., 2006] L'orchestration et la chorégraphie sont des modèles d'exécution de la composition. Ces deux modèles seront présentés dans la section composition des services web.

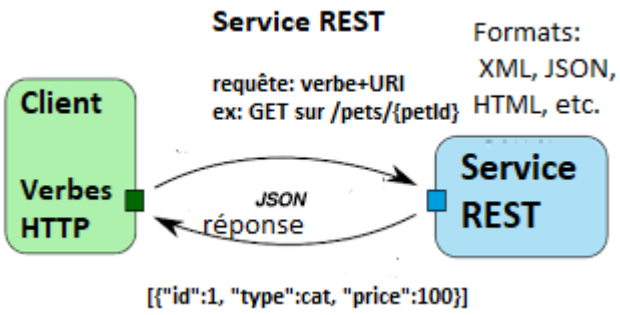

Figure 2.3 – *Exemple d'un service REST*

Nous présentons dans ce qui suit brièvement le standard WSDL.

- Web Services Description Language (WSDL) WSDL [Chinnici et al., 2007] fournit un modèle et un format XML pour décrire des wervices web. WSDL permet de faire une séparation entre la description abstraite et concrète d'un service. Le niveau abstrait permet de répondre à la question : 'quelle' fonctionnalité est fournie ?. Tandis que la partie concrète permet de répondre à la question : 'comment' et 'ou' celle-ci est offerte ? Par conséquent, le niveau abstrait regroupe les informations pouvant être réutilisées (non spécifiques à un service), tandis que le niveau concret est constitué de la description des protocoles d'accès au service web (information particulière à un service). Le niveau abstrait est utilisé principalement lors du processus de sélection. Il est constitué de types de données, des messages et des interfaces du document WSDL. Tandis que le niveau concret est seulement utilisé lors de l'invocation des méthodes du service web. Il est constitué d'informations relatives aux ports d'accès et aux protocoles de communication ainsi que les liaisons de services (bindings).

2. Une description sémantique qui est réalisée grâce à des annotations sémantiques de wsdl comme WSDL-S [Akkiraju et al., 2005] , SAWDL ou des ontologies de services comme l'ontologie OWLS [Martin et al., 2005] ou WSMO.

Nous définissons dans ce qui suit la facette structurelle et comportementale relatives à la description syntaxique.

2.3.4 Couche de découverte des services

Un service web doit être d'abord référencé pour qu'il puisse être par la suite découvert et utilisé par des clients ou organisations. Pour cela, il existe des annuaires pouvant être soit internes à l'organisation, soit universels. Nous décrivons dans ce qui suit le registre universel UDDI.

- UDDI (Universal Description Discovery and Integration) UDDI [Bellwood et al., 2002] est un standard pour décrire un annuaire de services web. UDDI est initialement adopté par OASIS pour publier et de découvrir des services web. Un registre UDDI est une base de donnée orientée XML qui contient des informations et des métadonnées concernant les services web. Trois types d'informations peuvent être trouvées dans ces registres XML :

- Les pages blanches : décrivent les informations de contacts sur les entreprises. Elles sont utilisées pour trouver les services par contact car elles contiennent toutes les informations jugées pertinentes pour identifier l'organisation (telles que son nom, son adresse physique).
- Les pages jaunes : décrivent des informations de classification de services. Elles sont utilisées pour trouver les services à l'aide des taxonomies car ces pages jaunes décrivent les services offerts par l'organisation, le type de servicesf et les conventions d'utilisation de ces services.
- Les pages vertes : donnent des informations techniques (telles que l'accès) sur les services. Elles sont utilisées pour trouver la localisation ou la sémantique formelle du service (en l'occurrence, nous pouvons trouver des URLs pour les documents WSDL ou OWLS/SAWDL.

2.3.5 Couche de composition des services

Cette couche sert a traiter le processus de composition qui permet d'agréger des services web. La composition permet de construire de nouveaux services appelés services composites ou agrégats par assemblage de services déjà existants. Les services participant à la composition sont nommés services basiques ou élémentaires ou encore atomiques. En industrie, le concept de composition des services web est réduit aux concepts d'orchestration et de chorégraphie. Dans ce contexte, les efforts industriels se sont concentrés sur la description de l'aspect comportemental des services web composites, en développant des langages de composition spécifiques tels que WS-BPEL et WSCDL. Ces langages permettent de décrire des plans de composition (traduisant des comportements ou encore la manière dont les composants doivent être agrégés) pour qu'ils puissent, par la suite, être exécutés par des serveurs d'applications. Nous décrivons dans ce qui suit de manière brève les deux modèles de composition à savoir l'orchestration et la chorégraphie et nous donnerons les principes du langage BPEL (Business Process Execution Language) .

2.3.5.1 Modèle comportemental du processus de la composition

Les propriétés comportementales décrivent la manière avec laquelle un service composite est construit. Ce comportement est lié au processus de la composition où l'information sur le comportement interne du service composite (la chorégraphie) et le comportement externe des services qui le composent (l'orchestration) est primordiale afin d'interagir avec le service composite.

- Modèle d'orchestration

  L'orchestration décrit le comportement externe d'un service composite. Elle désigne la manière dont les services invoqués sont agrégés au niveau de la composition afin de fournir une fonctionnalité plus complexe. L'orchestration permet de décrire l'enchaînement des services selon un modèle prédéfini [Sadiq and Racca, 2003], et de les exécuter comme étant un ensemble d'actions à réaliser par l'intermédiaire de services web. Dans ce type de modèle, toutes les communications sont controlées et routées par un processus central appelé moteur d'orchestration ou moteur de workflow (voir figure 2.4).

  Le moteur d'orchestration aussi appelé 'coordinateur' prend le contrôle de tous les services web impliqués et coordonne l'exécution des différentes actions des

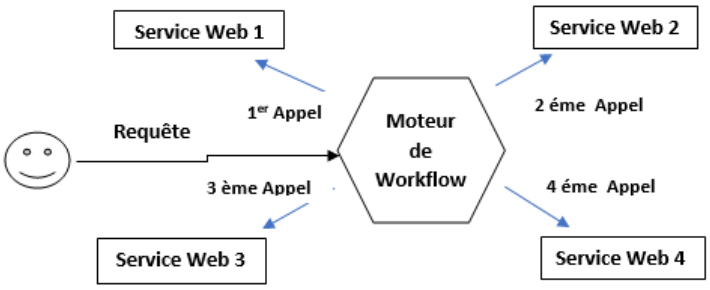

Figure 2.4 – *L'orchestration des services*

services composants qui participent dans le processus. Les services composants ne doivent pas être modifiés pour exécuter le workflow. Un exemple typique de langage d'orchestration est WS-BPEL [Jordan et al., 2007].

- Modèle de chorégraphie
  Contrairement à l'orchestration, il n'existe pas un coordinateur dans la chorégraphie, elle est basée sur la collaboration entre les services web (voir figure 2.5) où chaque partie de la composition possède un contrat ou des accords avec le reste des services qui dictent la manière de collaborer.

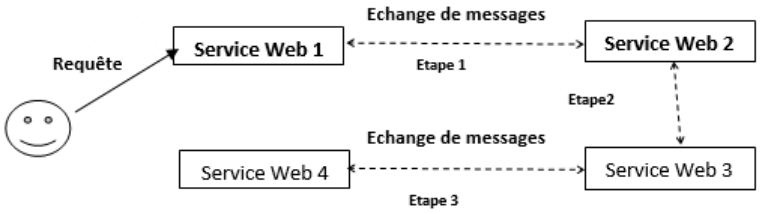

Figure 2.5 – *Vue générale de la chorégraphie*

La chorégraphie peut être considérée comme une composition dynamique [Peltz, 2003]. En effet, à chaque étape du processus de la composition, un service web choisit le service web qui lui succède et implémente ainsi une partie de la chorégraphie. Un exemple typique d'un langage de chorégraphie est la spécification web Services Choreography Description Language (WSCDL) [Ross-Talbot and Fletcher, 2006], [Kavantzas et al., 2005].

2.3.5.2 Langage BPEL

Plusieurs langages de composition de services ont été proposés dans la littérature, mais un seul d'entre eux est devenu le formalisme le plus utilisé, c'est le standard BPEL ou BPEL4WS [Andrews et al., 2003] (pour la version 1.0) ou [Jordan et al., 2007] (pour la version 2.0). BPEL est basé sur XML et sur les Workflows. Il permet de modéliser les processus métiers en terme d'orchestration, en décrivant le flux de données (les variables) et le flux de contrôle (les activités simples et composées, les partenaires...), son but est de créer une fonctionnalité complexe qui réutilise les services existants. Il permet aussi de donner une vue centralisée de l'exécution de la composition. Ce langage distingue les processus abstraits des processus exécutables.

Le processus abstrait décrit les interactions publiques de messages entre les services web composants sans obligation d'indiquer le comportement interne de chacun d'eux. Par opposition, le processus exécutable montre l'ordre d'exécution des activités, des services invoqués, des messages échangés et des mécanismes de gestion des erreurs potentielles. En d'autres termes, il s'agit du moteur de l'orchestration donnant une représentation indépendante des interactions entre les services composants. Deux catégories d'activités peuvent être distinguées dans bpel, les activités primitives et les activités structurées. Les activités primitives (Receive, Reply, Assign, Throw et Wait) sont des activités simples utilisées pour décrire comment le processus effectué par le service Web composite doit réagir vis-à vis des messages échangés. Par ailleurs, les activités structurées (Sequence, Flow, while Switch ou Split) sont des activités complexes qui combinent plusieurs activités primitives pour décrire différents types de branchement dans un processus.

2.3.6 Couche de qualité de services

Il n'existe pas de langage ou annuaire dédié exclusivement à la spécification de la QoS des services web. Cependant, il est nécessaire d'avoir une spécification claire et non ambigüe de la QoS du service afin de permettre leur évaluation. Deux possibilités sont offertes pour faire cette spécification.

2.3.6.1 L'extension des langages de description des services par la QoS

Les langages de description de service comme UDDI et WSDL ne concernent que les aspects fonctionnels du service, de nombreuses propositions ont été faites pour renforcer ces spécifications afin de permettre la description des paramètres QoS du service. Une des contributions les plus en avant, vient du groupe ouvert OASIS. Le groupe a proposé la prolongation du WSDL par la QoS appelée WSQDL (Web Service Quality Description Language). Nous trouvons aussi, l'extension de WSDL nommée Performance-enabled WSDL (P-WSDL) [D'Ambrogio and Bocciarelli, 2007] et l'extension d'UDDI nommée (UX) [Zhou et al., 2004]. Dans ces deux formalismes, la QoS d'un service est modélisée comme un tuple de paramètres de QoS, en spécifiant une valeur (ou un intervalle de valeurs) pour chaque paramètre.

2.3.6.2 Utilisation des contrats de QoS

Nous parlons aussi de Service Level Agreements (SLA), les contrats sont des accords conclus entre le fournisseur et le client d'un service concernant la QoS du service. Les contrats peuvent préciser les obligations du fournisseur et du client. Par exemple, un contrat peut dire que "à condition que le client fasse au plus cinq demandes par seconde, le fournisseur assure que ces demandes soient traitées en moins de 100 millisecondes". La première partie de cette clause est une obligation que le client doit respecter et la seconde est une obligation du fournisseur. Un contrat peut avoir plusieurs clauses de ce genre, qui décrivent ensemble la QoS du service. Les contrats sont généralement négociés en off-line, avant que le service ne soit appelé par le client. Toute méthode de gestion de QoS impliquant des contrats est accompagnée par des techniques de monitoring, pour veiller à ce que les obligations dans le contrat soient respectées. WSLA [Keller and Ludwig, 2003] et WS-Agreement [Andrieux et al., 2007] sont deux cadres répandus pour la spécification des contrats pour

les services web.

2.4 Composition des services web

La composition de services permet de répondre aux besoins complexes des utilisateurs qui ne peuvent etre satisfaits par des services simples [Benatallah et al., 2005]. Dans ce qui suit, nous citons quelques définitions pour le concept de composition de services.

Selon [Casati and Shan, 2002] "la composition de services est la capacité d'offrir des services à valeur ajoutée en combinant des services existants potentiellement offerts par différentes organisations". Elle est aussi définie par [Kellert and Toumani, 2003] comme étant "une technique permettant de combiner des services pour réaliser un objectif particulier, par l'intermédiaire de primitives de contrôles (boucles, tests, traitement d'exception, etc.) et d'échanges (envoi et réception de messages)". En d'autres mots, la composition de services spécifie quels services seront en mesure d'être invoqués, dans quel ordre, quelles sont leurs données à consommer, et comment gérer les cas d'exceptions. La composition de services peut être considérée comme un mécanisme permettant l'intégration des services dans une application [Benatallah et al., 2005]. La définition la plus répandue et la plus référencée dans la littérature est celle de [Benatallah et al., 2005]. Les auteurs estiment que la composition des services web est "un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent les uns des autres". Nous constatons que ces définitions s'accordent sur l'hypothèse que la composition de services a pour objectif la création de nouveaux services, donnant de nouvelles fonctionnalités, à partir des services existants. Le nouveau service qui en résulte est communément appelé *service composite*. Son exécution requiert l'invocation de plusieurs autres services afin de tirer profit de leurs fonctionnalités. Les services appelés lors d'une composition de services sont qualifiés comme étant *services composants*. La Figure 2.6 montre le principe de la composition de services ; à partir d'un ensemble de services existants (et faisant partie d'un registre, nous construisons un service composite [Chollet, 2009].

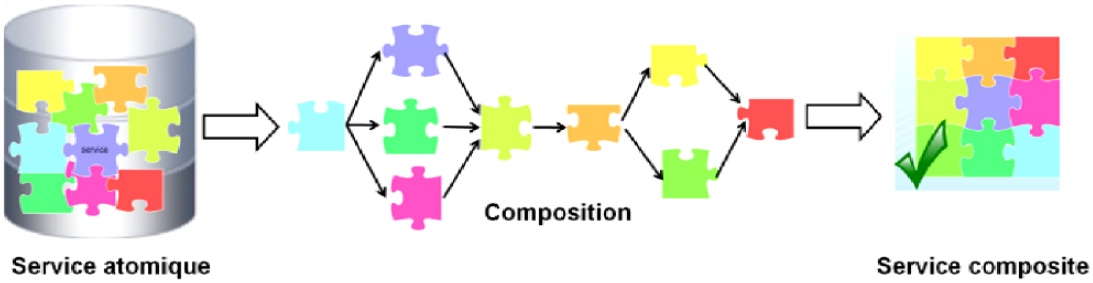

Figure 2.6 – *Composition de services*

2.4.1 Cycle de vie d'une composition de services

Le processus de raffinement d'une composition de services depuis son stade de définition vers son stade opérationnel passe par plusieurs étapes qui incluent : la

phase de *définition*, la phase de *découverte et de sélection*, la phase de *déploiement et d'exécution*, et la phase de *monitoring* [Sheng et al., 2014] (Figure 2.7)

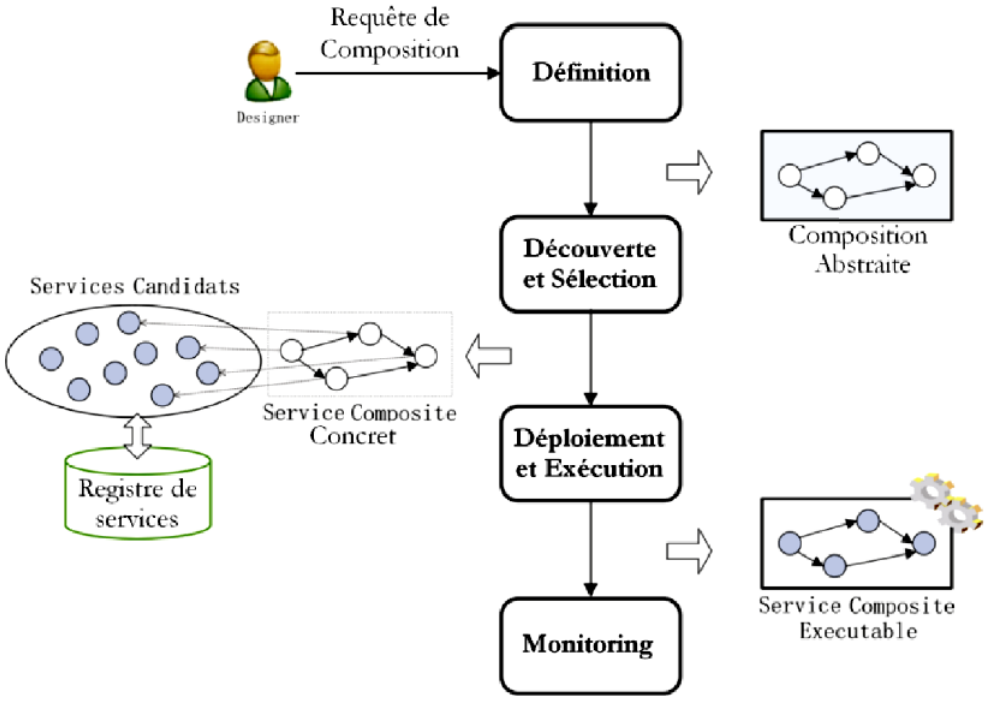

Figure 2.7 – *Les différentes phases d'une composition de services*

- **La phase de définition :** cette étape spécifie les les fonctionnalités requises dans une composition. Elles peuvent être décrites avec des taches abstraites, avec des interfaces d'entrées/sorites/pré-conditions/post-conditions, et même avec des restrictions les critères de QoS.
- **La phase de découverte et de sélection :** cette étape met en oeuvre une recherche dans les annuaires et les portails des services en comparant la spécification fonctionnelle et non fonctionnelle de la requête avec celles des services publiés. Le but final de cette phase est de fournir une liste services classés selon leur ordre de pertinence.
- **La phase de déploiement et d'exécution :** les services retenus dans la phase précédente (et qui sont déja déployés dans leurs plateformes) sont invoqués par le moteur du workflow.
- **La phase de monitoring :** l'exécution d'une application composée est surveillée par un moteur du workflow. Ce dernier assure les activités suivantes : (1) la journalisation et la visualisation des détails d'exécution de la composition, (2) le calcul des statistiques de QoS à partir de l'analyse des données et des fichiers LOG, et (3) la vérification des contraintes fonctionnelles, ainsi que l'évaluation des attributs non-fonctionnels du service composé.

2.4.2 Typologie des méthodes de composition

Les méthodes proposées pour la composition des services peuvent être classées en fonction de plusieurs facteurs, tels que le degré ou niveau d'automatisation, le type

de contrôle des services, et la gestion des services. [Khanouche et al., 2016], [Yachir, 2014] (Figure2.8).

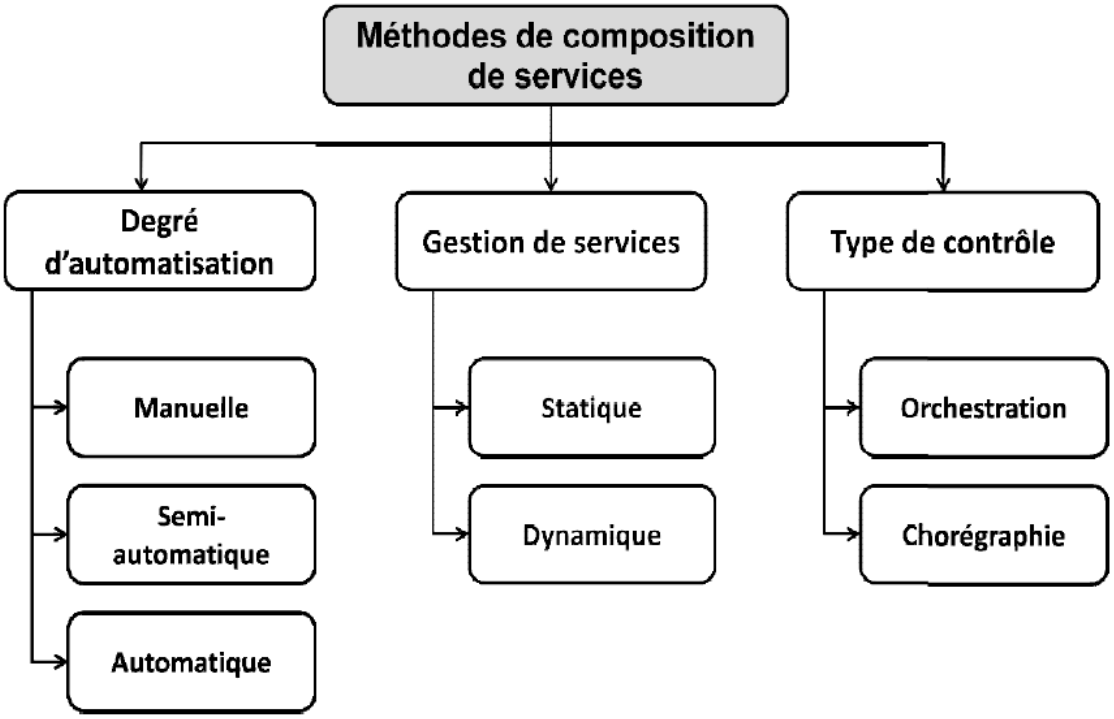

Figure 2.8 – *Les différentes classes de composition de services*

2.4.2.1 Type de contrôle

Nous distinguons deux types de compositions : l'orchestration ou la chorégraphie des services [Chollet, 2009]., nous les survolons brièvement par la suite :

L'orchestration : une orchestration implique que les services du processus composite sont exécutés dans un ordre prédéfini et de manière centralisée. Le moteur qui assure l'invocation de tous les services est appelé *orchestrateur* ou moeur de workflow [Zribi, 2014]. L'orchestration est définie dans [Barros et al., 2005], [Benatallah et al., 2005].

La figure 2.9 montre un exemple d'orchestration de quatre services (et dont le flux de controle est prédéfini).

Nous notons que le langage BPEL (Business Process Execution Language) est le standard de facto pour définir les orchestrations [Standard, 2007].

La chorégraphie : Contrairement à l'orchestration qui a un caractère statique, la chorégraphie est un assemblage de services de nature dynamique [Peltz, 2003]. Une composition de type chorégraphie est construite progressivement (non connue à l'avance). A chaque étape, un service sélectionne le prochain élement qui le suit et implémente ainsi un segment de la chorégraphie. Selon [Barros et al., 2005], la chorégraphie permet de décrire la composition comme une collaboration de services (contrairement à l'orchestration qui impose un orchestrateur central) pour la réalisation d'un objectif explicite.

La figure 2.10 montre un exemple de composition de 04 services (service1, service 2, service 3 et service 4) de type chorégraphie. L'utilisateur (humain ou application)

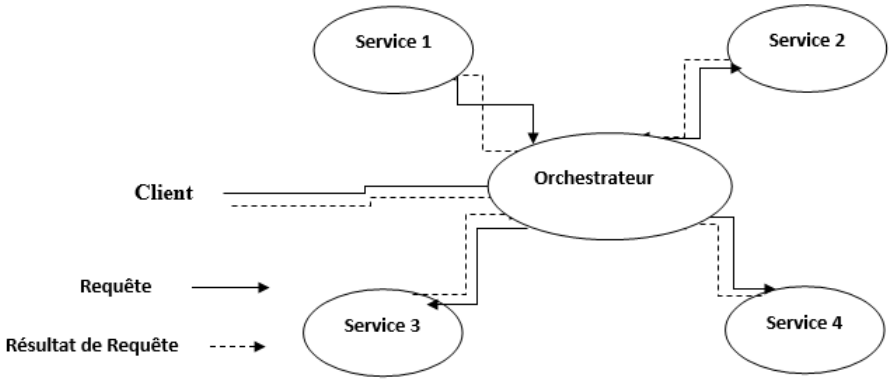

Figure 2.9 – *Orchestration de services*

envoie une requête qui est communiquée en premier lieu au premier service (service 1). Ce dernier découvre ensuite le service qui le suit. Une fois le service 2 découvert, les deux services (service 1 et service 2) valident la faisabilité de leur communication. Si les échanges sont concluants et parfaits, le résultat de l'exécution du service 1 est communiqué au service 2. Ce processus est réitéré jusqu'à ce que le service 4 achève son exécution.

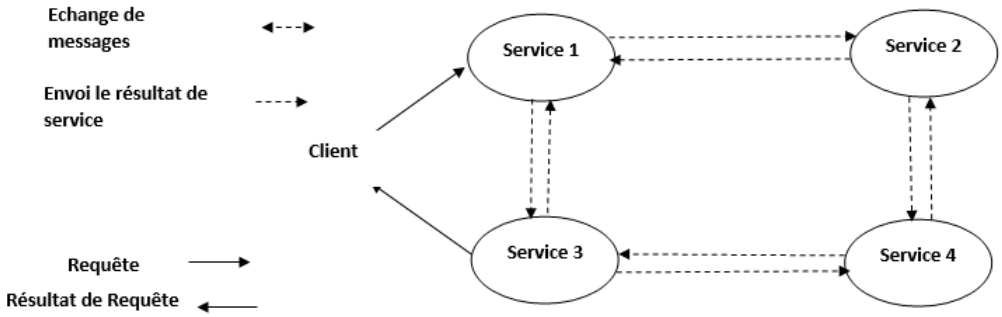

Figure 2.10 – *Chorégraphie des services.*

Nous notons que le langage WS-CDL est le standard de facto (du (W3C) pour la chorégraphie des services [Ross-Talbot and Fletcher, 2006], [Kavantzas et al., 2005].

2.4.2.2 niveau d'automatisation

Nous distinguons trois types de degrés d'automatisation des compositions : *la composition manuelle, la composition semi-automatique et la composition automatique.*

La composition manuelle : dans ce cas, un expert humain s'en charge de spécifier les services ainsi que le flux de controle et de données qui les connectent. Les limites de la composition manuelle se résument comme suit : le passage à l'échelle n'est plus garanti pour la découverte et la sélection de services. En second lieu, l'utilisateur doit posséder une compétence et un savoir faire dans la composition, et ceci n'est pas toujours faisable.

La composition semi-automatique : les approches semi-automatiques permettant d'assurer une aide pour les utilisateurs durant les phases de construction du

workflow et meme durant l'instanciation (sélection) des activités abstraites. En effet, des éditeurs graphiques permettent d'aider l'utilisateur à concevoir flux de tâches (workFlow). Nous notons que ce type d'approches a aussi des problèmes pour le passage à l'échelle.

La composition automatique : dans ce cas de figure, l'intervention de l'humain n'est plus requise durant la séléction des services ou leur enchainement.Ce genre de composition est trés souhaitable dans les environnements pervasifs, mobiles, volatiles, ou meme agiles.

2.4.2.3 Gestion de services

En ce qui concerne l'instant de creation du workflow, nous pouvons distinguer deux cas de figures : les compositions statiques et celles qui sont dynamiques.

Composition statique : une composition statique implique que le workflow est réalisé au moment de la conception de l'application. Les services à composer sont sélectionnés et fixés à l'avance (flux de controle et flux de données sont figés) [Dustdar and Schreiner, 2005]. Les compositions statiques, sont aussi appelées compositions proactives ou offline, elles ne sont pas adaptées aux environnements dynamiques ou pervasives.

Composition dynamique : une composition de services est considérée comme dynamique si les services sont sélectionnés et composés à la volée (et aprés la présentation de la requête) [Osman et al., 2005]. Ce type de compositions est aussi appelé composition réactive ou on-line. Les compositions dynamiques sont convenables pour les environnements pervasives et agiles.

2.5 Conclusion

Nous avons présenté dans ce chapitre, un survol sur l'architecture SOA ainsi que la technologie des services web. Nous avons présenté les différents standards qui assurent l'intégration et l'interopérabilité des services.

Nous avons défini par la suite la phase de composition de services web et cité les différentes techniques et modèles qui permettent son achèvement.

Le chapitre suivant présentera un état de l'art sur les approches systèmatiques qui assurent la composition des services web.

Chapitre 3

Etat de l'art sur la composition de services

Sommaire

3.1 Introduction

Dans la plupart des cas, le fournisseur de services propose un service web intéressant garantissant une fonctionnalité spécifique. Par ailleurs, les utilisateurs demandent généralement des applications complexes qui ne peuvent pas être réalisées avec un service web élémentaire.Par exemple, si un utilisateur veut acheter un produit, il ne suffit pas de le rechercher, mais il est nécessaire de s'occuper du paiement, de la livraison, etc. L'invocation de plusieurs services coopérant ensemble implique le processus de composition des services. Nous remarquons que le choix des composants de service appropriés est compliqué pour l'utilisateur,en raison de la prolifération de services similaires qui proposent des fonctionnalités identiques.

Malgré la similarité fonctionnelle, chaque service est caractérisé par des propriétés non fonctionnelles appelées qualité de service (QoS). Essentiellement, la QoS représente l'aspect clé pour traiter le problème de la composition des services. En fait, il s'agit de choisir les services et/ou les compositions les plus appropriés ayant la meilleure QoS et respectant les exigences de l'utilisateur. Néanmoins, la fluctuation de l'environnement SOA en général, et des services web en particulier, entraîne une instabilité de la QoS au fil du temps. Par conséquent, elle perturbe le processus de composition, puisqu'il n'est pas approprié de traiter la QoS comme une valeur statique.

Dans ce chapitre, nous décrivons les concepts de base de la composition de services web. Nous commençons par l'introduction de quelques concepts de base. Nous expliquons ensuite le rôle de la QoS dans le processus de composition. Ensuite, nous présentons la classification principale des approches de composition de services et les travaux connexes de chaque catégorie. Enfin, nous abordons les recherches basées sur la QoS incertaine.

3.2 Composition des services web

De nombreuses définitions ont été introduites pour la composition de services web. Dans [Papazoglou et al., 2008], la composition de services est définie comme une tâche consistant à développer des services complexes en composant d'autres services élémentaires ou composites. Un service élémentaire fournit un point d'accès à des applications Internet qui ne dépendent pas d'autres services web pour répondre à des requetes externes. Un service composite regroupe d'autres services élémentaires et/ou composites qui coopérent pour mettre en œuvre un ensemble d'opérations.

Une autre définition de [Paik et al., 2017], stipule que "l'activité d'agrégation de services web pour construire un nouveau service est connue sous le nom de composition de services web". Par ailleurs, la définition la plus simple est que la composition de services web se réfère au processus de combinaison de plusieurs services pour fournir un service à valeur ajoutée [Luo et al., 2010].

3.2.1 Eléments de la composition des services web

1. **Activités** : Une activité indique le travail qui doit être effectué. Les activités peuvent être atomiques, auquel cas elles sont appelées tâches, ou non atomiques, auquel cas elles sont appelées sous-processus [Paik et al., 2017].
2. **Flow de données** : Il s'agit du flux de messages entre les activités, pour le service web composite. Il représente les échanges de messages (appelés flux de messages) qui joignent les services.
3. **Flow de contrôle** : Il représente l'ordre d'exécution des services web atomiques dans la composition.
4. **Workflow** : Un workflow contient généralement un ensemble d'activités et la séquence entre elles [Tan and Zhou, 2013]. Une activité peut être soit une activité manuelle qui nécessite une intervention humaine, soit une activité automatisée qui sera exécutée par une application logicielle. En ce qui concerne les services web, le workflow pour la composition de services web est généralement composé de tâches et de transitions qui dénotent les dépendances entre les tâches et elles sont associées à une probabilité d'activation (enabling probability) [Cardoso et al., 2004]. Selon des études récentes,les structures élémentaires du modèle de workflow comprennent la séquence, le parallèle, le choix et la boucle.
5. **Contraintes globales** : Il s'agit de l'un des concepts les plus importants pour filtrer les alternatives. Il représente les exigences sur une certaine séquence de variables.
6. **Client** : Organisation ou personne qui reçoit un produit ou un service.
7. **Utilisateur final** : Personne(s) qui utilisera(ont) en fin de compte le système dans le but pour lequel il(s) a(ont) été conçu(s).

8. **Fonction objectif** : La fonction objectif générale visée par la communauté des chercheurs pour la sélection des services est de maximiser la satisfaction du demandeur de service par rapport à l'exécution du service composite [Moghaddam and Davis, 2013].

3.2.2 Rôles de la QoS dans la sélection des services web

La QoS a été largement abordée au cours des dernières années, depuis la garantie de la meilleure qualité pour les télécommunications et les réseaux informatiques jusqu'à l'extension de ce concept à la sélection des services web, les services cloud et les systèmes IoT. La norme internationale ISO 8402 décrit la qualité comme l'ensemble des caractéristiques d'un produit ou d'un service qui influent sur son aptitude à satisfaire des besoins exprimés ou implicites.

En fait, la QoS est un ensemble de caractéristiques (c'est-à-dire de critères) décrivant les attributs non fonctionnels. Chacune d'entre elles est représentée sous forme de valeurs quantitatives et limitée par une fourchette et des frontières. Dans le contexte des services web, les valeurs de ces attributs de la QoS peuvent être soit directement collectées auprès des fournisseurs de services (par exemple, le prix), enregistrées à partir d'exécutions précédentes (par exemple, le temps de réponse), soit collectées à partir des commentaires des utilisateurs (par exemple, réputation) [Liu et al., 2004].

Les attributs de la Qualité de Service (QoS) peuvent être classés en attributs positifs et négatifs. Les attributs positifs, tels que le débit, la réputation et la disponibilité, visent à être maximisés, tandis que les attributs négatifs, tels que le coût et le temps de réponse, doivent être réduits au minimum [Alrifai et al., 2010], [Alrifai and Risse, 2009]. Selon [Zheng et al., 2010], il existe seulement deux catégories : les attributs dépendants de l'utilisateur et les attributs indépendants de l'utilisateur.

Au préalable, les catégories dépendantes de l'utilisateur se composent de tous les attributs de la QoS qui sont différents pour chaque utilisateur, tels que le débit et le temps de réponse. La catégorie indépendante de l'utilisateur comprend les attributs qui sont similaires pour tous les utilisateurs. Les propriétés de la QoS de cette catégorie comprennent la popularité, le prix, etc. Selon [Wang et al., 2012], les attributs de QoS peuvent être classés en QoS mesurables ou non mesurables. Les attributs de QoS mesurables sont quantifiables à l'aide d'une mesure de QoS, telle que le temps d'exécution moyen,tandis que les attributs de QoS non mesurables sont naturellement qualitatifs (par exemple, la flexibilité, la réputation, etc.).

Dans [Lee et al., 2003], la classification est différente, les attributs de la QoS pouvant être considérés comme des attributs d'application et de réseau. Les catégories de réseau comprennent les attributs de QoS réseau tels que le délai du réseau, la variation du délai, la perte de paquets. Ces derniers attributs sont ceux qui indiquent la communication entre un service web et le monde extérieur. Néanmoins, les attributs de la QoS de l'application sont descriptifs de l'application des services web, tels que la disponibilité, la fiabilité, le coût, etc. D'autre part, la QoS peut être considérée comme déterministe ou non déterministe [Liu et al., 2004]. Un attribut est considéré comme déterministe si sa valeur est reconnue avant l'invocation du service (par exemple, le prix, la sécurité). Toute fois, si l'attribut est inconnu au moment de l'invocation du service (par exemple, le temps de réponse ou la disponibilité), cet attribut est alors considéré comme non déterministe.

Souvent, chaque demande d'utilisateur comprend deux spécifications, la description fonctionnelle et les exigences de la QoS qui représentent les contraintes de la

QoS sur les services demandés. D'autre part, de nombreux fournisseurs de services peuvent répondre de manière fonctionnelle à la demande de l'utilisateur, chacun s'efforce d'offrir la meilleure QoS ou au moins d'atteindre l'objectif de QoS qu'il a spécifié. La gestion de la concurrence entre les fournisseurs au cours du processus de composition peut être réalisée en sélectionnant entre plusieurs services candidats sur la base de la QoS offerte. Cette étape représente la sélection locale. De même, pour contrôler le nombre de compositions possibles, la similarité entre la QoS de la composition et la QoS demandée par l'utilisateur est prise en compte, ce qui représente la sélection globale. En fait, le processus de composition peut inclure la sélection locale ou globale. Cependant, s'il inclut les deux, il est considéré comme une sélection hybride (voir la section 3.4 pour plus de détails).Intégrer a QoS dans le processus de composition est connue sous le nom de QoS-aware service composition.

3.2.2.1 Fonctions d'agrégation de la QoS

En pratique, le passage de la QoS des services atomiques à la QoS des services composites est traité selon le modèle de workflow et une fonction d'agrégation de la QoS. Cette dernière est un mécanisme utilisé pour calculer la QoS globale de la composition sur la base d'un ensemble de fonctions. Elle dépend des structures du workflow. D'une manière générale, l'agrégation de la QoS dans les workflow séquentiels est différente de celle des workflow parallèles ou en boucle. Dans cette thèse, nous utilisons les fonctions d'agrégation proposées dans [Hwang et al., 2014] (voir la table 3.1).

Tableau 3.1 – *Modèles d'agrégation des attributs de QoS [Hwang et al., 2014], [Alrifai et al., 2012]*

Attribut de QoS	**Séquence**	**Parallèlisme**	**boucle (k itérations)**	**Choix (if)**
Latence	$\sum_{i=1}^{n} q(S_i)$	$MAX_{i=1}^{n} q(S_i)$	$\sum_{i=1}^{k} q(S)$	$MAX_{i=1}^{n} q(S_i)$
coût	$\sum_{i=1}^{n} q(S_i)$	$\sum_{i=1}^{n} q(S_i)$	$\sum_{i=1}^{k} q(S)$	$MAX_{i=1}^{n} q(S_i)$
Disponibilité	$\prod_{i=1}^{n} q(S_i)$	$\prod_{i=1}^{n} q(S_i)$	$\prod_{i=1}^{k} q(S)$	$MIN_{i=1}^{n} q(S_i)$
Fiabilité	$\prod_{i=1}^{n} q(S_i)$	$\prod_{i=1}^{n} q(S_i)$	$\prod_{i=1}^{k} q(S)$	$MIN_{i=1}^{n} q(S_i)$
Réputation	$\frac{1}{n} \sum_{i=1}^{n} q(S_i)$	$\frac{1}{n} \sum_{i=1}^{n} q(S_i)$	$q(S)$	$MIN_{i=1}^{n} q(S_i)$
Débit	$MIN_{i=1}^{n} q(S_i)$	$MIN_{i=1}^{n} q(S_i)$	$q(S)$	$MIN_{i=1}^{n} q(S_i)$

3.2.2.2 Poids des attributs de QoS

Généralement, l'utilisateur a des préférences ou une importance concernant chaque attribut de QoS pendant le processus de composition. Néanmoins, de nombreux chercheurs affirment que l'utilisateur n'a qu'une idée incomplète de ses préférences en matière de QoS. C'est pourquoi de nombreux travaux ont proposé des solutions pour calculer les pondérations de QoS. En conséquence, nous pouvons distinguer quatre approches principales en fonction des poids de QoS exploités :

- **Pondérations subjectives :** Ces approches utilisent les poids appliqués par les utilisateurs. Ces poids sont extraits de leurs données de rétroaction [Karim et al., 2011].
- **Pondérations objectives :** Dans ces approches, certaines méthodes systématiques sont adoptées pour calculer les poids de QoS pour ajuster les préférences de l'utilisateur, telles que l'entropie proposée dans [Sun et al., 2016].
- **Pondérations subjectives et objectives combinées :** Ces approches intègrent à la fois des poids subjectifs et objectifs, notamment le travail proposé dans [Ouadah et al., 2019].
- **Pondérations équitables pour la QoS :** Dans ces méthodes (telles que [Yasmina et al., 2018]), des poids équitables sont attribués à chaque QoS.

3.2.3 Catégories de composition

La composition des services web peut être classée selon l'implication de l'utilisateur dans le processus de composition en trois catégories : manuelle, automatique et semi-automatique. Elle peut également être classée en fonction de la flexibilité et de la dynamicité du processus de composition, où nous distinguons la composition statique et la composition dynamique.

- **Composition statique :** Une composition statique signifie que le processus de composition est statiquement formé, déployé et fixé en construisant le modèle de processus comprenant les services atomiques et les dépendances entre eux. Dans ce cas, il n'y a aucune possibilité de modification après l'exécution, même si les fonctionnalités des services ou les exigences de composition changent.
- **Composition dynamique :** Étant donné que l'environnement est dynamique au cours de l'exécution et que le processus de composition doit s'adapter à cette flexibilité, le processus de composition doit être dynamique. Selon [Rao and Su, 2004], la composition dynamique est réalisée en créant un modèle abstrait de tâches et en sélectionnant automatiquement les services web atomiques sans que le demandeur de service n'intervienne dans le processus de composition.
- **Composition manuelle :** Dans une composition de services web manuelle, nous créeons un assemblage où un workflow avec un environnement graphique et l'utilisateur intervient dans la sélection des services web. En plus, l'exécution de la composition se fait en exécutant les services un par un. Par conséquent, la modification ou l'extension de la composition est compliquée. Ce type de composition est généralement géré par le programmeur en utilisant des langages de processus d'entreprise, tels que Web Ontology Language for Web Services (OWL-S), le Web Services Flow Language (WSFL) d'IBM, et Business Process Execution Language for Web Services (BPEL4WS), pour spécifier le schéma de composition. La présence de nombreux outils (tels que ZenFlow, Sedna et Tavern), et de techniques n'empêche pas que la création d'une composition manuelle de services web est une tâche trés difficile, et sujette aux erreurs.
- **Composition semi-automatique :** La composition semi-automatique est une méthode améliorée et plus rapide pour la composition de services, par rapport à la méthode manuelle où le système et les utilisateurs sont tous deux impliqués dans le processus de recherche, de sélection et de composition automatique des

services. Dans cette approche, le workflow est conçu avant de commencer le processus de composition.

- **Composition automatique :** La composition automatique de services est un terrain en expansion continuelle, il permet de surmonter les limites de la composition manuelle. La composition automatique de services tend à éliminer ou à atténuer l'implication de l'utilisateur dans le processus de composition, ce qui aura un impact positif sur le processus de composition. La composition automatique est le processus dans lequel le workflow est automatiquement généré par des techniques de raisonnement ou de recherche combinatoire (algorithmes articulés sur des ensembles de contraintes et de préférences).En effet, le modèle de processus sera créé automatiquement et la composition peut changer au cours de l'exécution.

La composition automatique de services web se scinde en plusieurs catégories d'approches, elles sont résumées comme suit :

- **Approches basées sur les workflows**
- **Approches basées sur des modèles formels (automates/réseaux de petri)**
- **Approches de planification qui s'articulent sur l'intelligence artificielle (IA)**

Dans cette section, nous nous concentrerons sur les approches basées sur le workflow et sur les approches de planification en I.A.

- **Approches basées sur les workflows :** L'une des classes principales d'approches de composition automatique de services est la planification orientée workflow. En effet, le besoin de l'utilisateur est formalisé sous forme d'un graphe (ou workflow) montrant les activités requises et les connexions entre elles. Le workflow comprenant aussi le flux de données et le flux de contrôle. L'élaboration des workflows peut être statique ou dynamique. Dans les approches statiques, le modèle abstrait des activités est spécifié à l'avance par l'utilisateur. En revanche, le modèle de processus est généré automatiquement dans les approches dynamiques. L'utilisateur est résponsable sur la définition les contraintes et les préférences. **Cette classe de composition par workflow fera l'objet de nos contributions** (voir les chapitres 5 et 6).
- **Approches de planificationen I.A :** Ces méthodes sont exploitées pour la composition totalement automatisée de services web. Le problème de la composition des services est considéré comme un problème de planification (raisonnement). L'objectif du moteur planificateur est de générer un processus de composition basé sur un ensemble d'actions possibles ; ces actions permettent le déplacement depuis l'état initial vers un état final. Les travaux connexes se scindent en plusieurs catégories : la planification classique, (ou planification basée sur l'éspace d'états), la planification orientée graphe, la planification basée sur les algorithmes de satisfaisabilité.

3.3 Cycle de vie d'une composition de services

Dans cette section, nous présentons le processus de composition des services web. D'une façon générale, le processus de composition des services passe par quatre phases successives (voir la figure 2.7).

1. **Demande de service et planification** : Dans cette étape, l'utilisateur décrit le service composite requis en spécifiant ses contraintes et ses préférences. la requête peut etre aussi spécifiée (ou générée) sous forme d'un ensembles de tâches abstraites (avec les flux de données et controle.
2. **Découverte de services :** Ce processus de recherche de service est effectué sur les registres. il met en correspondance les bes besoins fonctionnels/non-fonctionnels de l'utilisateur avec ceux des services publiés.
3. **Composition des services :** Cette phase est constituée de deux sous-phases. En effet, la phase précédente renvoie plusieurs services similaires donnant des fonctionnalités similaires. En conséquence, la première sous-phase, nommée sélection de services, est adoptée pour sélectionner localement les services optimaux (ou quasi-optimaux) en fonction des propriétés non fonctionnelles (QoS). La deuxième sous-phase est la génération systèmatique de la composition possible en fonction du workflow initial, puis la retention de la composition optimale.
4. **Exécution et monitoring :** La dernière étape du cycle de vie est l'exécution de la composition. Les composants du service complexe sont invoqués dans un ordre connu à l'avance et exécutés (échange de messages). Cette composition est surveillée pour relancer les alternatives en cas de pannes ou échec.

3.4 Optimisation mono-objectif et multi-objectifs dans les compositions de services web

L'augmentation des services web fonctionnellement similaires représente un défi majeur pour la composition de services web. D'une manière générale, pour offrir une application à valeur ajoutée, il faut générer m^n combinaisons (n est le nombre de tâches et m le nombre de services par tâche). Ensuite, nous sélectionnons les solutions optimales ($TopK$ compositions). Dans ce cas, la sélection locale des services Web est bénéfique pour réduire le nombre de services liés à chaque tâche. La principale clé de caractérisation et différenciation entre les services est la QoS (par exemple, la QoS peut représenter le temps de réponse, la disponibilité ou la réputation).

Jusqu'à présent, "QoS-aware web service composition" est largement traitée par plusieurs chercheurs, où elle est considérée comme une optimisation à objectif unique ou à objectifs multiples :

- **Optimisation à objectif unique (SOO)(Single-Objective Optimization (SOO)) :** L'optimisation mono-objectif peut être décrite comme une technique d'optimisation basée sur une fonction objective unique où cette dernière est maximisée ou minimisée. La fonction objective comprend une combinaison de variables avec ou sans contraintes.
- **Optimisation multi-objectifs (MOO)(Multi-Objective Optimization (MOO)) :** (également appelée optimisation multicritères, multi-performances ou problème d'optimisation vectorielle) est une technique d'optimisation qui implique un ensemble de contraintes à satisfaire et plusieurs objectifs à maximiser ou à minimiser. La résolution des problèmes MOO consiste à résoudre de multiples sous-problèmes. Les solutions proposées connues sous le nom de solutions Pareto-optimales qui offrent un bon compromis entre les objectifs.

En général, il existe quatre notions fondamentales dans la MOO : dominance de Pareto, front de Pareto,l'optimum de Pareto, la skyline :

- **Dominance de Pareto :** Si nous avons le choix entre deux objets, l'un d'eux étant meilleur en ce qui concerne au moins un attribut mais au moins égal en ce qui concerne tous les autres attributs, les utilisateurs préféreront toujours le premier objet au second (le premier objet domine le second). En général, on peut dire que la solution i domine une autre solution j ; si la solution $i$est meilleure ou égale à la solution j pour tous les objectifs et est strictement meilleure que la solution j dans au moins un objectif. La définition des concepts suivants est basée sur la dominance de Pareto.
- **Solution optimale de Pareto (ensemble non dominé) :** Il s'agit de la meilleure solution ayant au moins un objectif optimal tout en préservant la qualité des autres objectifs. En général, cette solution n'est dominée par aucune autre solution (elle a nomme aussi Skyline). Une solution optimale de Pareto signifie qu'il n'existe pas d'autre solution qui puisse augmenter la qualité d'un objectif donné sans diminuer la qualité d'au moins un autre objectif [Talbi, 2009].
- **Front de Pareto :** Il représente l'ensemble de toutes les solutions Pareto-optimales équivalentes pour lesquelles il n'existe pas d'autre solution meilleure.

Le concept de dominance de Pareto peut être utilisé pour mettre en œuvre une extraction intuitive puisque les objets dominés peuvent être extraits de la collection de données ou des données finales de l'ensemble Skyline. Par exemple, un utilisateur demande à trouver des produits qui sont à la fois à la mode et pas chers. Étant donné qu'un produit à la mode est généralement plus cher qu'un autre, il est donc difficile de définir un produit "optimal" satisfaisant aux deux conditions. Cependant, les produits chers et non tendance seront éliminés de la liste des produits souhaités. Le reste des produits représente l'ensemble de Skyline.

En fait, la dominance de Pareto est adoptée pour établir des préférences parmi un ensemble donné de solutions. Par conséquent, elle est utilisée pour sélectionner le service web le plus approprié en fonction de la demande de l'utilisateur. La figure 3.1 montre l'ensemble dominé, l'ensemble non dominé et le front de Pareto (ou le skyline) dans le cas d'une demande de service web à faible coût et faible temps d'exécution (minimisation).

Différentes méthodes sont proposées dans la littérature pour adresser le problème MOO ; ces méthodes sont scindées en plusieurs classes :

1. **Les méthodes classiques :** Également appelées méthodes agrégatives, elles consistent à transformer le MOO en SOO en agrégeant les objectifs en une fonction scalaire.

   De nombreuses techniques sont proposées, telles que :

   - **L'agrégation pondérée :** Cette technique consiste à utiliser un poids pour chaque objectif, puis à transformer tous les objectifs en une seule fonction objective résolue par une méthode SOO.
   - **Contrainte :** L'idée de base de cette technique est de trouver des solutions optimales de Paretoen considérant un seul objectif et en traitant les autres objectifs comme des contraintes limitées par un vecteur.

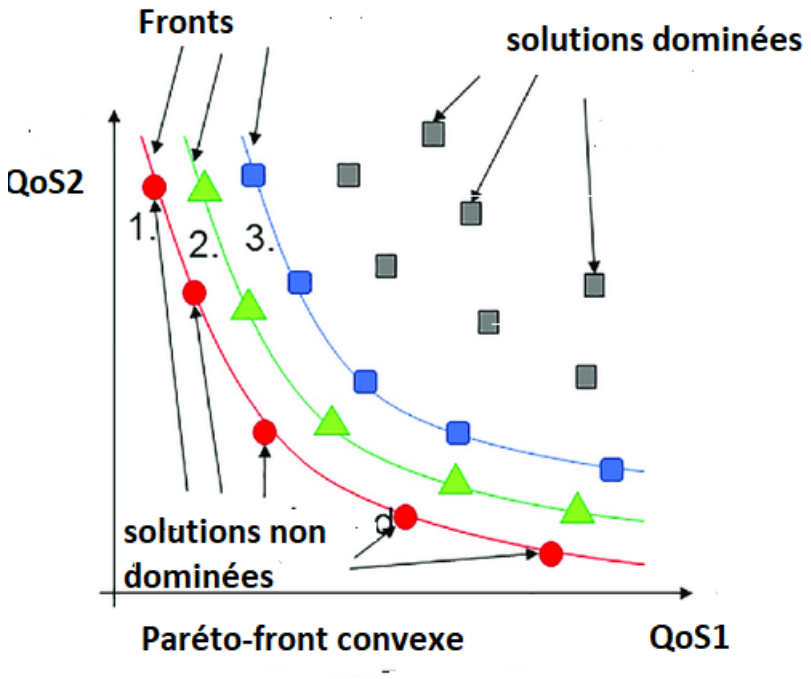

Figure 3.1 – *Front de paréto (solutions non dominées).*

- **La méthode de programmation des objectifs :** Il s'agit d'un programme d'optimisation. Elle peut être comme une extension ou une généralisation de la programmation linéaire pour traiter des mesures objectives multiples, normalement contradictoires. Chacune de ces mesures se voit attribuer un but ou une valeur cible à atteindre [Bhattacharya and Chatterjee, 2014].

2. **Approches basées sur Pareto :** Ces méthodes sont classées comme des méthodes a posteriori [Branke, 2008]. Dans cette catégorie, tous les objectifs sont pris en compte lors du traitement des problèmes de MOO. Cependant, il n'y a pas de préférences sur aucun objectif, cela signifie que la recherche d'un compromis entre les objectifs est primordiale. Ce compromis est traité par le biais du concept de dominance. Les techniques de résolution visent à trouver toutes les solutions non dominées possibles, telles que l'algorithme génétique de tri non dominé-II (Non-dominated sorting genetic algorithm-II) (NSGAII) [Deb et al., 2002].

3.4.1 Stratégies de composition

Nous pouvons mentionner trois stratégies de composition principales pour réaliser le processus de composition des services web :

- **Sélection globale :** La sélection globale vise à sélectionner des compositions quasi-optimales qui garantissent la satisfaction des exigences de l'utilisateur, appelées contraintes globales de QoS. Néanmoins, elle présente une complexité exponentielle et le coût de calcul est NP-hard (c.-à-d., Non-deterministic Polynomial-time Hard).
- **Sélection locale :** Une stratégie de sélection locale se concentre sur la sélection du service le plus pertinent pour chaque activité du processus abstrait, qui sera

combiné pour former une composition. Les services sélectionnés présentent le meilleur compromis entre les attributs de la QoS requis. Le sapproches concernées ont une complexité linéaire et un faible coût de calcul. Cependant, il n'y a aucune garantie que la contrainte globale soit satisfaite puisqu'elles ne traitent que les contraintes locales.

- **Sélection hybride :** Cette classe représente un compromis entre les deux premières approches. Elle est moins complexe par rapport à l'approche globale, et elle peut également tenir compte des contraintes globales.

3.4.2 Classification des techniques de composition de services web

Selon [FethAllah, 2014], les techniques de résolution de la composition de services à base de QoS (QoS-aware web service composition) peuvent être scindées en quatre classes principales :

1. **Méthodes exactes** :

   La propriète majeure de ces méthodes est la garantie de l'optimalité de la solution recherchée. En général, les méthodes exactes sont utilisées pour des instances de problème de taille peu élevée. Dans le cas d'instances de problèmes à grande échelle, le temps de calcul de ces approches est exagéré.

   Plusieurs méthodes exactes ont été proposées et ont assuré l'obtention des solutions optimales, nous citons : la programmation en nombres entiers (PI), la programmation dynamique (PD), la programmation linéaire (PL) ou les algorithmes de graphes (Integer Programming (IP), Dynamic Programming (DP), LinearProgramming (LP), or graph algorithm).

   Concernant les méthodes exactes, [Gabrel et al., 2014] s'appuie sur un nouveau modèle ILP pour tenir compte des transactions dans la composition de services. L'ensemble des services est aussi modélisé par un graphe de dépendance de services (SDG). À partir de ce graphe, l'approche de programmation linéaire 0-1 est mise en oeuvre pour trouver la composition optimale.

   Dans la même ligne d'idées, la proposition de [Liu et al., 2012] présente une méthode basée sur la technique évaluation-separation (branch and bound) appelée (BB4EPS). Cette approche est utilisée pour résoudre la composition de services avec prise en compte des préférences et des contraintes des utilisateurs. Le travail cité dans [Alrifai et al., 2012], introduit une approche centralisée pour la composition de services (QoS-aware service composition) en tenant compte d'un modèle de composition complexe (séquentiel, itératif, parallèle et conditionnel). L'approche proposée commence par l'utilisation des techniques de résolution MIP pour trouver la décomposition des contraintes globales de QoS en contraintes locales. Ensuite, la sélection locale vise à trouver les meilleurs services web qui satisfont à ces contraintes locales. Le programme MIP proposé a besoin de quelques données de QoS des services pour atténuerla charge temporelle du processus de composition.

   Shetty et D'Mello [Shetty and D'Mello, 2018] ont utilisé à la fois la recherche locale et globale pour sélectionner des services cloud. L'optimisation locale permet de réduire l'éspace de recherche, tandis que l' optimisation globale permet de sélectionner des solutions optimales grace à l'algorithme de Dijkstra.

Dans [Wang et al., 2017], les auteurs proposent une nouvelle approche appelée SCORE-QoS on Correlated QoS Requirements. Dans cette approche, le problème de sélection de service est modélisé comme un problème d'optimisation de contraintes (Constraint Optimization Problem (COP)) en intégrant deux types d'exigences de QoS (avec ou sans les objectifs d'optimisation des utilisateurs). Pour trouver la solution optimale pour le COP, la programmation en nombres entiers (Integer Programming (IP)) est employée.

Dans [Fan et al., 2018], les auteurs s'appuient sur un mécanisme efficace pour traiter le problème de composition de services web. Le graphe de dépendance des services est généré pour décrire les services et les relations entre eux. En plus, le problème de la composition représente la recherche d'un chemin atteignable dans le graphe de dépendance des services, tel que knapsack-variant-algorithm. En outre, une stratégie d'optimisation est proposée pour réduire la complexité de knapsack-variant-algorithm.

Dans [Ghobaei-Arani and Souri, 2019], les auteurs proposent une approche basée sur la programmation linéaire, appelée "LP-WSC", pour gérer la QoS-aware web service composition dans un environnement de cloud distribué géographiquement. Tout d'abord, la méthode de normalisation Z-score est appliquée pour normaliser les données de QoS, car le Z-score est adapté pour traiter les données aberrantes. Ensuite, le centre le plus proche est sélectionné en fonction de la distance géographique de l'utilisateur. Enfin, la technique de programmation linéaire est utilisée pour sélectionner la composition la plus appropriée qui répond aux exigences de l'utilisateur sous la forme d'un accord de niveau de service (SLAs). Cependant, l'augmentation du nombre de variables de décision entraîne une complexité exponentielle et des coûts exacerbés. Les méthodes exactes résolvent le problème d'optimisation de manière optimale. Elles utilisent les techniques de la programmation par contraintes ou la programmation dynamique, la programmation linéaire entière (Integer Lineaire Programming, ILP) ou encore les techniques de programmation entière et mixte (Mixed Integer Programming ou MIP). Ces méthodes assurent des solutions optimales mais elles ont une charge temporelle exponentielle.

la programmation linéaire en nombres entiers (ILP) a été utilisée pour résoudre le problème de la sélection optimale de services Plusieurs travaux [Zeng et al., 2004], [Liu et al., 2012], [Ardagna and Pernici, 2007] ont adopté la programmation linéaire en nombres entiers (ILP) pour trouver les services optimaux. L'approche citée dans [Zeng et al., 2004] emploie deux méthodes qui tiennent compte des contraintes de QoS. La première technique effectue une recherche locale de services sans tenir compte des contraintes globales. Pour chaque activité abstraite (cad, classe de services) de la composition, une méthode d'aide à la décision multicritère (MCDM – Multiple Criteria Decision Making) est employée afin de trouver le l'instance optimale. Malgré que la QoS est optimisée au niveau de chaque tâche de services, la méthode n'asssure pas nécessairement à une QoS agrégée optimale (d'une composition). Un autre atout de cette solution est le temps de traitement qui est polynomial. Pour faire face à ces limites (relatifs à la sélection locale), la deuxième méthode suggérée par les auteurs tient compte des restrictions globales de QoS imposées au service composé, et considère la sélection de la composition optimale comme un problème de Programmation Linéaire en Nombres Entiers avec une fonction objectif à maxi-

miser, et un ensemble de restrictions (contraintes) à satisfaire. Cette méthode assure une composition avec une QoS globale optimale, mais elle consomme aussi un temps de traitement exorbitant (exponentiel).

[Ardagna and Pernici, 2007] adoptent la programmation linéaire pour tenir compte des contraintes locales. Dans cette approche, les contraintes globales sont attachée à la composition entière, en outre, les contraintes locales peuvent être définies par le concepteur de la composition (au niveau des tâche). Dans le même scénario, [Gao et al., 2006] adoptent la programmation entière en prenant en compte des contraintes de conflit entre les services (qui appartiennent à des tâche différentes). Les conflits expriment le fait que l'utilisation d'un service i de la tâche X n'est pas compatible (ou en opposition) avec le service j de la tâche Y. L'expérimentation confirme que l'ajout des conflits, cause une élevation de 13% dans le temps d'exécution de la méthode à base de MIP.

Certaines méthodes de composition à base de QoS utilisent des techniques d'optimisation basées sur le decoupage des contraintes de QoS [Alrifai et al., 2012], [Sun and Zhao, 2012] où le problème de recherche globale est divisé en plusieurs sous-problèmes de recherche (optimisation). Des restrictions locales de QoS sont déduites à partir des restrictions globales. Pour chaque tâche abstraite de la composition, une sélection locale est ensuite appliquée pour trouver les instances satisfaisant les contraintes locales de QoS correspondantes. Le travail de [Alrifai et al., 2012] propose une approche pour la composition de services avec QoS qui met en oeuvre deux phases : 1) une décomposition des restrictions globales de QoS et 2) une recherche locale. Dans la première étape, les restrictions globales de QoS sont décomposées en un ensemble de restrictions locales pour chaque tâche abstrait. Dans la deuxième étape, une recherche locale est appliquée pour trouver la meilleure instance en termes de QoS et qui satisfait aussi les contraintes locales. Les restrictions locales sont utilisées comme limites supérieures pour les valeurs de QoS des instances candidats. Les services qui violent ces limites (bornes) sont ignorés lors de la recherche. Une liste de services candidats satisfaisant les restrictions de QoS est alors générée et triée par ordre de pertinence.

Le travail de [Sun and Zhao, 2012], propose une autre méthode de composition de services basée sur la décomposition des restriction de QoS. Elle constituée de trois étapes majeures : une phase de décomposition, une phase de recherche, et une phase d'amélioration. Dans la première phase, l'utilité d'un service composé, ou utilité globale, est calculée à partir de l'utilité des instances de services, ou utilité locale ; les restrictions relatives aux services candidats, ou restrictions locales, sont déduites à partir des restrictions imposées par l'utilisateur sur le service composé. Dans la deuxième phase, le service qui satisfait les restrictions locales est sélectionné pour chaque tâche abstraite. S'il y a au moins une instance qui satisfait les restrictions locales, la tâche abstraite correspondante est marquée comme satisfaite. Dans le cas opposé, la tâche est marquée comme insatisfaite. Dans la troisième phase, l'utilité globale est boostée en adaptant les contraintes locales de toutes les tâche de services. L'algorithme s'achève si l'utilité globale se stagne. L'approche de [Rosenberg et al., 2009] offre un modèlisation qui traite la sélection comme un problème d'optimisation par contraintes relaxées. L'objectif est de trouver une solution qui optimise les contraintes pondérées. L'idée de base dans l'approche est de consider des contraintes souples

(qui peuvent etre ignorées) et d'autres contraintes rigides qui doivent etre satisfaites.

[Gabrel et al., 2014] proposent une approche pour trouver la solution optimale pour la composition de services avec gestion des aspects transactionnels et en employant un graphe de dépendance et la programmation linéaire 0-1.

2. **Méthodes heuristiques :**

Il convient de noter que complexité temporelle des approches exactes est exorbitante (elle est exponentielle et elle dépend de la taille du problème). Pour pallier à cette lacune, d'autres approches visualisent le problème de sélection de services composés comme un problème de d'optimisation combinatoire [Yu et al., 2007] ou un problème de recherche dans un graphe [Yu et al., 2007] ; [Llinás and Nagi, 2015]. Dans la spécification combinatoire, la sélection de services est définie comme un problème de sac à dos avec plusieurs dimensions et à choix multiples (MMKP – Multi-dimensional Multi-choice Knapsack Problem) [Martello and Toth, 1987] alors que le modèle de graphe specifie la sélection de services comme un problème du plus court chemin sous restrictions (MCSP – Multi-Constrained Shortest Path problem) [Khanouche et al., 2016]. Les méthodes de cette classe fonctionnent uniquement pour des problèmes d'optimisation spécifiques (puisque l'heuristique proposée est spécifique pour un problème donné), elles assurent des solutions quasi-optimales et un temps de traitement raisonnable.

Le travail de [Llinás and Nagi, 2015] permet de rechercher plusieurs instances de services pouvant être exécutées séquentiellement afin d'améliorer la QoS du service composé. Pour assurer cet objectif, la composition de services est représentée sous forme d'un problème de recherche, dans un graphe acyclique orienté (DAG – Directed Acyclic Graph), du plus court chemin satisfaisant des restrictions spécifiques. L'algorithme MCSP-FP (MCSP-FP – Multiple Constrained Shortest Path-Feasibility Potential) booste l'algorithme du plus court chemin à origine unique (SSSP – Single-Source Shortest Paths) en introduisant le concept de réalisabilité potentielle en termes de besoins de QoS. La technique MCSP-FP est différente de la technique MCSP proposée dans [Yu et al., 2007], dans la mesure où seuls les chemins potentiellement satisfaisables sont stockés pour la recherche du chemin optimal. Le gain apporté par la technique MCSP-FP est dû à l'utilisation de conditions plus serrées sur la faisabilité d'un solution.

[Akbar et al., 2006] suggèrent une autre heuristique, C-UHE, et évaluent sa pertinence et son optimalité contre plusieurs heuristiques candidates, y compris l'algorithme M-UHE de [Yu et al., 2007]. Les résultats montrent que la C-UHE est meilleure par rapport à M-UHE en termes de temps CPU. Par contre, les évaluations montrent que M-UHE est la meilleure en termes de degré d'optimalité, mais l'optimalité de C-HEU se dégrade à mesure que le nombre d'instances par classe s'éleve.

La méthode proposée dans [Xia et al., 2011] offre une sélection globale de composition de services en tenant compte de quatre structures de flux de contrôle : parallélisme, séquence, choix conditionnels, et les boucles, la requête est spécifiée en BPEL, elle possède cinq attributs de QoS. L'algorithme (nommé aussi "qssac") donne un résultat proche de l'optimal. Pour minimiser le temps CPU, ils groupent les services similaires en termes de QoS à l'aide d'un algorithme de clustering appelé optics (qui est peu sensible aux données aberrantes). OP-

TICS fournit M groupes (avec leurs représentants) pour chaque tâche, ensuite l'algorithme scanne toutes les compositions possibles pour 2 ou K tâches de la structure BPEL, et les trie en fonctiion de leur pertinence. Dans la même ligne d'idées, [Luo et al., 2011] ont proposé un algorithme heuristique noté HCE pour la composition de services (satisfaisant les contraintes de QoS). [Comes et al., 2010] ont fourni un modèle de sélection heuristique pour la recherche de compositions de services. [Li et al., 2014] ont proposé une méthode efficace et fiable pour la sélection de compositions optimales des services à base d'attributs de confiance.

[Lecue and Mehandjiev, 2009], [Klein et al., 2011] adoptent l'algorithme *hill climbing* pour baisser la complexité de temporelle du calcul et comparent leur approche avec celle utilisant LIP.

[Feng et al., 2013] adoptent des algorithmes d'apprentissage semi-supervisés (reinforcement learning) pour trouver l'ensemble de solutions Pareto optimales qui tiennent compte des paramètres de QoS et les préférences des clients finaux.

D'autres travaux ont proposé plusieurs techniques heuristiques [Luo et al., 2011], [Comes et al., 2010] et [Do Prado et al., 2013] pour baisser la complexité temporelle de la recherche locale ou globale, lors de l'énumération des compositions de services.

Dans [Baranwal and Vidyarthi, 2016], les auteurs considèrent de multiples critères de QoS et introduisent des métriques de QoS pour mesurer les services en nuage. En outre, la méthode improved ranked voting method(IRVM) est exploitée en tenant compte de l'importance des mesures de QoS pour un cadre de vote afin de sélectionner le meilleur fournisseur avec le score le plus élevé.

Dans [Wang et al., 2017], les auteurs proposent une nouvelle approche appelée SCORE-QoS based on Correlated QoS Requirements. Dans cette approche, le problème de sélection de service est modélisé comme un problème d'optimisation des contraintes (COP) intégrant deux types d'exigences de QoS (avec ou sans optimisation des utilisateurs).(avec ou sans les objectifs d'optimisation des utilisateurs).La programmation en nombres entiers (IP) est utilisée pour trouver la solution optimale pour le COP.

La sélection de services web basée sur la QoS peut être solutionnée avec les méthodes de prise de décision multicritères (Multi-Criteria Decision Making) (MCDM). Dans cette optique, [Purohit and Kumar, 2018] améliorent une méthode MCDM nommée "organisation du classement des préférences pour l'évaluation de l'enrichissement" (Preference Ranking Organization Method for Enrichment Evaluation) (PROMETHE) [Brans et al., 1984]. Cette méthode est adaptée avec succès pour le problème de la sélection et composition de services web.

Les travaux de [Ouadah et al., 2019] introduisent l'approche Skyline-Entropy-Fuzzy-Ahp-Promethee(SEFAP) pour la sélection des services skyline. Cette approche est basée sur l'algotithme Block Nested Loops (BNL) pour générer le service web skyline. En outre, l'entropie et l'AHP flou (the Entropy and Fuzzy AHP) (Analytic Hierarchy Process) sont utilisés pour extraire les poids objectifs et subjectifs. Enfin, la méthode PROMETHEE est proposée pour classer les services web skyline.

Le travail présenté dans [Al-Faifi et al., 2019] utilise une méthode MCDM hybride pour sélectionner le meilleur fournisseur de services cloud (cloud service

provider) (CSP) à partir de données intelligentes. La méthode hybride intègre le clustering en adoptant l'algorithme k-means qui regroupe les fournisseurs de services en k-clusters. Ensuite, DEMATEL est adopté pour obtenir un proxy de chaque cluster et pour déterminer l'interdépendance et les relations entre les critères. Enfin, le processus analytique de réseau (Analytical Network Process) (ANP) est utilisé pour classer les clusters et fournir le CSP pertinent. Le modèle hybride de prise de décision multicritère est proposé dans [Jatoth et al., 2019] pour la sélection des services dans un environnement cloud. Le modèle hybride appelé (EGTOPSIS) se compose à la fois de AHP et des versions étendues de Grey TOPSIS pour calculer les poids des critères ; ce dernier est ensuite utilisé pour classer les services. La recherche de [Serrai et al., 2019], exploite la méthode AHP pour générer des poids normalisés des critères de la QoS dérivés de la demande de l'utilisateur. Ensuite, ils introduisent une nouvelle approche appelée OMRI (Optimized Method of Reference Idea). L'OMRI est une amélioration de la technique de normalisation des données appelée RIM(Reference Ideal Method), car cette dernière ne garantit pas un bon classement en raison de la technique de normalisation des données utilisée dans le processus. En effet, la méthode OMRI est combinée avec différentes méthodes de classement MCDM appropriées (weighted Product Method (WPM),Simple Additive Weighting (SAW), VIKOR, et Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)) pour classer les services beb.

Dans [Youssef, 2020], la méthode du meilleur pire (Best Worst Method) (BWM) est utilisée pour acquérir les poids des critères et les scores relatifs des alternatives. Ces poids et scores sont utilisés dans la méthode TOPSIS pour classer les services cloud.

Dans un travail plus récent [Tiwari and Kumar, 2021], les auteurs adoptent le TOPSIS Gaussien(G-TOPSIS) pour un classement robuste par inversion. Cette méthode est exploitée pour traiter le problème de la sélection des service scloud basée sur la MCDM en fonction de la QoS. Pour évaluer l'approche proposée,les auteurs utilisent l'ensemble de données Cloud Harmony comme fournisseur de services de référence.

3. **Méthodes méta-heuristiques** :

   Le terme "méta-heuristique" a été proposé pour la première fois dans [Glover, 1986], où méta signifie "au-delà" ou "niveau supérieur" et heuristique est le mot grec qui signifie qui signifie "heuriskein" ou "chercher". En général, ces approches sont plus performantes que les simples heuristiques. Elles sont utilisées pour résoudre une grande partie des différents problèmes NP-difficiles car ces derniers nécessitent un temps de calcul élevé. En fait, la plupart des approches méta-heuristiques représentent un processus itératif et aléatoire. En outre, un nombre d'entre elles s'inspirent des systèmes naturels. La principale caractéristique des méta-heuristiques est qu'elles ne sont pas proposées pour un problème particulier, mais elles sont approximatives et recherchent une solution satisfaisante.

   Selon [Canfora et al., 2005], [Chen and Wang, 2007], les méta-heuristiques sont les techniques d'optimisations les plus utilisées pour la résolution des problèmes combinatoires NP-difficiles. Elles ont été très adoptées pour résoudre le problème de composition des services à base de QoS

   Les algorithmes méta-heuristiques imitent les comportements d'exploitation et d'exploration (également appelés intensification et diversification). L'explora-

tion consiste à explorer l'éspace de recherche en générant diverses solutions, tandis que l'exploitation consiste à exploiter la zone locale et la recherche de bonnes solutions dans cette région [Talbi, 2009].

Les méta-heuristiques peuvent être classées en deux catégories ;les méthodes basées sur la population (telles que les algorithmes génétiques) et les méthodes point à point (telles que la recherche taboue ou le récuit simulé [Glover, 1989], [Černỳ, 1985]). Les méthodes orientées la population (telles que les algorithmes génétiques et l'optimisation par essaims de particules -particle swarm optimization-) se caractérisent par l'utilisation de multiples particules. En revanche, les méthodes point à point utilisent un seul agent ou une seule solution et améliorent cette solution à l'aide d'une recherche locale. Par exemple, le recuit simulé et la recherche Tabu (simulated annealing and Tabu search) sont des méthodes point à point.

[Canfora et al., 2005] proposent l'un des premiers algorithmes génétiques déstinés à la résolution du problème de composition à base de QoS. L'approche de composition adopte les caractéristiques fonctionnelles et du non-fonctionnelles (QoS). Le travail cité dans [Yu et al., 2013], a proposé un algorithme génétique dans lequel ils ont modélisé le problème sous forme d'un arbre. De même, [Liu et al., 2010] ont mis en oeuvre un algorithme génétique sophistiqué utilisant l'optimisation par colonies de fourmis (ACO) pour améliorer la convergence de l'algorithme. Dans [Xiangbing et al., 2012], les auteurs expriment le problème de composition des services web en utilisant l'ontologie(WSMO) et en mettant en oeuvre un algorithme génétique pour minimiser le temps de recherche de la solution optimale.

Dans [Elsayed et al., 2017], les auteurs adoptent une nouvelle méthode qui combine l'utilisation de l'algorithme génétique (AG) et l'apprentissage par renforcement (*Q-learning*) pour retenir la composition optimale. La performance des AG est fortement dépendante de la population initiale, et dans ce sens, le *Q-learning* est employé pour générer la population initiale afin de booster l'efficacité de l'AG.

Les algorithmes génétiques ont des difficultés relatives à la scalabilité, (le codage des chromosomes est figé), en plus il ont aussi une difficulté concernant la stagnation dans un optimum local. D'autres travaux ont adopté des algorithmes d'optimisation pour trouver une solution quasi-optimale. Le travail de [Wang et al., 2010] propose l'optimisation par colonies de fourmis (ACO) pour retenir une composition quasi-optimale, ils analysent les performances en variant plusieurs variables (le seuil d'évaporation de la phéromone, la population initiale, etc). Une approche de composition de services (gérant la QoS et les propriétés transactionnelles) est proposée dans Dans [Wu and Zhu, 2013]. L'approche modélisée le problème comme étant une construction de chemin dans un graphe acyclique orienté (DAG). La technique d'optimisation par colonies de fourmis (ACO) [Dorigo et al., 2006] est utilisée pour rechercher le meilleur chemin dans le DAG (celui qui satisfait les contraintes transactionnelles et de QoS globales).

Dans [Xia et al., 2008], les auteurs modélisent la composition de service en termes de graphes tels que les nœuds sont des motifs d'orchestration et les arcs sont des services, ils emploent l'ACO, mais ils stipulent que la présence d'un seul type de phéromone n'est pas convenable pour traiter plusieurs attributs de QOS. [Yang et al., 2010] adoptent une hybridation de l'ACO et les algorithmes

génétiques pour la résolution de composition de services. L'algorithme génétique a pour but d'adapter les paramètres d'ACO, qui sera utilisé dans l'étape de sélection, les auteurs observent une accélération notable au niveau du temps d'exécution de l'algorithme amélioré par rapport à l'algorithme originale. [Hadjila, 2014] introduit une technique de "sélection clonale" pour la résolution du problème de sélection de services. Ils adoptent une fonction mono-objectif qui gére cinq critères de QoS. Les auteurs estiment que la sélection clonale est plus prometteuse que l'algorithme génétique.

L'architecture proposée dans [Zhao et al., 2012] adopte un algorithme d'optimisation par colonies de fourmis amélioré pour la composition multidimensionnelle et multi-objectifs des services. La technique ACO mise en œuvre introduit des changements dans la phéromone et la probabilité de transition. La technique proposée dans [Mohammed et al., 2014] emploie la meta-heuristique d'harmonies pour adresser la composition de services. Dans le même champ d'idées, [Bekkouche et al., 2017] proposent la meta-heuristque d'harmonies qui tient compte de contraintes globales de QoS et de la sémantique des entrées/-sorties des services.

Les algorithmes génétiques et leurs extensions sont largement utilisés pour résoudre le problème de la sélection des services web tenant compte de la QoS (QoS-aware web service sélection). Pour surmonter les limites de l'algorithme génétique de base, le travail de [Li et al., 2018] présente un algorithme génétique évolutif coopératif (CGA)basé sur une méthode de codage réelle. En effet, la méthode de l'entropie est utilisée pour déterminer les poids de chaque attribut de la QoS. [Ranjan and Sahoo, 2020] est un autre travail qui introduit une approche basée sur l'algorithme génétique pour sélectionner la composition optimale dans un environnement de brouillard (fogenvironment). Cette approche tient compte à la fois des préférences de l'utilisateur et des attributs de la QoS mentionnés dans l'accord de niveau de service (SLA). Les travaux de [Thangaraj and Balasubramanie, 2021] introduisent une approche méta-heuristique hybride pour la sélection de services tenant compte de la QoS (QoS-aware sélection of services). Cette approche est basée à la fois sur l'algorithme génétique et la recherche Tabu pour réduire l'éspace de recherche tout en respectant les contraintes de QoS.

Dans [Li et al., 2020], les auteurs proposent une nouvelle approche pour traiter le problème de la QoS-aware web service composition. Tout d'abord, une nouvelle méthode appelée Fuzzy Optimal Continuity Construction (FOCC) est proposée pour représenter les relations de voisinage continues floues entre les services, ce qui représente un prétraitement, afin de rendre les algorithmes méta-heuristiques efficaces dans un éspace discret. Ensuite, l'algorithme Chaos Harris Hawk Optimization (CHHO) est proposé pour retrouver la meilleure composition à l'aide de la stratégie Logistic Chaotic Single-Dimensional Perturbation (LCSDP) et une méta-heuristique appelée Harris Hawk Optimization (HHO). Les auteurs de [Dahan et al., 2019] améliorent les performances de la colonie d'abeilles artificielle (Artificial Bee Colony) (ABC) en adoptant les stratégies de sélection et d'échange de nœuds voisins et de permutation.

Afin de prendre en compte la QoS basée sur les intervalles (interval-based QoS), la distance d'encombrement originale de NSGAII est améliorée. Dans [Dahan, 2021], une méthode d'optimisation par colonies de fourmis basée sur des agents

(MAACS) est exploitée. Ils proposent une décomposition de la recherche et des techniques de réorganisation de la population en temps réel pour réduire la complexité la composition des services dans un cloud.

Le travail de [Hosseinzadeh et al., 2020] traite un modèle de composition de services dans cloud edge computing par le biais d'une méthode hybride qui combine le réseau neuronal artificiel (Artificial Neural Network) (ANN) et l'algorithme méta-heuristique Particle Swarm Optimization (Particle Swarm Optimization(PSO)). En effet, afin d'étayer l'exactitude et d'améliorer l'algorithme ANN-PSO proposée, une méthode de vérification formelle est utilisée. Cette dernière est basée sur un système de transition étiqueté pour vérifier certaines formules critiques de la logique temporelle linéaire (Linear Temporal Logics) (LTL).

Pour la composition de services cloud, les auteurs de [Naseri and Jafari Navimipour, 2019] présentent une approche hybride basée sur une méthode à base d'agents et PSO. La méthode basée sur les agents est utilisée pour composer les services en cloud en identifiant les paramètres de la QoS, et l'algorithme PSO est employé pour sélectionner les meilleurs services cloud combinés sur la base d'une fonction d'aptitude.

Dans [Fekih et al., 2016], les auteurs utilisent la méthode skyline pour réduire le nombre de services candidats. Ensuite, ils proposent l'approche Valued Constraint Satisfaction Optimization(VCSOP) basée sur la méta-heuristique PSO, pour modéliser le problème de composition de services web tout en tenant compte des besoins de l'utilisateur et des changements de contexte.

Dans [Dahan, 2021], les auteurs introduisent une méta-heuristique améliorée, appelée Enhanced Flying Ant ColonyOptimization (EFACO). Il est proposé de restreindre le processus de vol et un nouveau processus de sélection des voisins pour surmonter le problème du temps d'exécution. Cependant, la qualité de la sélection s'en trouve réduite. Par conséquent, ils ont utilisé une méthode multi-phéromone pour améliorer l'exploration en attribuant une valeur de phéromone à chaque attribut de la QoS. Les travaux de [Barkat et al., 2021] portent sur la composition de services web dans un environnement multi-cloud. Par conséquent, les auteurs ont donc utilisé un algorithme de gouttes d'eau intelligentes -Intelligent Water Drops (IWD)- basé sur la QoS.

4. **Méthodes basées sur la dominance de Pareto :**

   Ces méthodes sont utilisées pour gérer le nombre croissant de services sur le web. En outre, elles tiennent compte de l'implication de l'utilisateur dans l'introduction de ses préférences. Plus précisément, ces approches sont proposées pour résoudre des problèmes multi-objectifs en recherchant un ensemble de solutions optimales.

   Dans [Alrifai et al., 2010], les auteurs proposent une approche pour la sélection et la composition de services basée sur la relation de dominance. Tout d'abord, l'éspace de recherche est réduit en éliminant les services qui ne sont pas dans la ligne de mire de chaque classe abstraite. Ensuite, l'algorithme K-means est utilisé pour le regroupement hiérarchique afin de réduire le nombre de skyline. Enfin, l'éspace de composition est exploré en utilisant la combinaison des clusters (Clusters heads combination).

   Le travail de [Halfaoui et al., 2015], propose une heuristique basée sur un nouveau concept de sélection locale, appelée Averaged-Fuzzy-Dominated-score

(noté AFDetS. Cette approche est adoptée pour classer les services en fonction de la QoS. Dans [Yu and Bouguettaya, 2013], les auteurs proposent trois algorithmes pour la recherche de compositions de skyline (notées C-SKY) : One Pass Algorithm (OPA), Dual Progressive Algorithm (DPN) et Bottom-Up Algorithm (BUA). Selon l'algorithme OPA, toutes les compositions possibles sont générées en combinant les services skyline (C-Skies). L'algorithme DPN trie les services composites de la skyline par le biais d'une fonction d'aptitude où les critères de QoS sont agrégés en fonction de la somme. Toutefois, cette approche suppose que la QoS des différents services est indépendante les uns des autres. Dans [Benouaret et al., 2011], les auteurs proposent une approche basée sur la relation de dominance floue à travers un nouveau concept, appelé dominance. Ce conceptest utilisé pour élaguer les services Pareto-dominés et sélectionner les services Pareto-dominants. En plus de permettre aux utilisateurs de contrôler la taille de skyline en modifiant le degré, les services web sélectionnés présentent un bon compromis entre les attributs de QoS. Toutefois, cette approche ne tient pas compte de la composition des services. Dans [Wang et al., 2016], le KD-tree (K-Dimensional tree) est adopté pour trouver les services web skyline à partir de clusters. En effet, chaque cluster regroupe les services ayant les mêmes fonctionnalités.

Étant donné que le nombre de services dans les clusters est important, la recherche des services skyline est gérée en employant l'algorithme KDS. En effet, l'ordonnancement par loterie est pour planifier et sélectionner les services. L'idée principale est de distribuer des billets de loterie aux services ; les services de meilleure qualité obtiendront plus de billets et auront plus de chances d'être exécutés.

Dans [Serrai et al., 2016], les auteurs proposent une approche hybride de sélection des services web. Tout d'abord, la méthode skyline présentée dans [Alrifai et al., 2010, Alrifai et al., 2012] est employée pour éliminer les services dominés et permet de réduire l'éspace de recherche. Deuxièmement, la méthode Best-Worst-Method(BWM) est adoptée pour attribuer un poids aux critères de la QoS, car la BWM est moins complexe, plus cohérente et plus fiable que l'AHP. Enfin, la méthode VIKOR (Multi-criteria Optimization and Compromise Solution) est utilisée pour classer et trier les services skyline en fonction de leur pertinence par rapport aux pondérations de la QoS.

Dans [Permadi and Santoso, 2018], les techniques de skyline sont employées pour traiter la composition des services. Cette méthode consiste à calculer les skyline dynamiques à l'aide de l'algorithme Sort And Limit Skyline Algorithm(SALSA) pour trouver la solution optimale qui répond aux préférences des utilisateurs tout en respectant les contraintes budgétaires. Cette approche est proposée pour surmonter les problèmes de l'éspace de recherche. Néanmoins, le temps de calcul continue d'augmenter lorsque le nombre de compositions est important.

Dans [Liang et al., 2019], les auteurs proposent un algorithme skyline amélioré utilisant la méthode Bitmap pour traiter la sélection de services web basée sur la QoS (QoS-based web service sélection). Cet algorithme peut réduire les vérifications de dominance dans les régions et l'ensemble de services candidats, tout en économisant de l'éspace mémoire.

Le travail cité dans [Khanouche et al., 2016] propose une approche nommée EQSA (Energy-centered and QoS-aware services selection). Cette méthode de

sélection de services est centrée sur l'énergie et les autres facteurs de QoS. Cette approche peut gérer la scalabilité et permet aussi de réduire l'énergie consommée. Les auteurs emploient l'optimisation lexicographique pour retenir les services satisfaisant un niveau prédéfini de QoS. Par la suite la mesure de dominance relative au sens de Pareto permet de retenir les meilleurs services candidats de la composition.

3.5 Synthèse

La table 3.2 récapitule les approches présentées dans ce chapitre en les classant selon les attributs suivants : la catégorie d'approche, la technique de résolution utilisée, les attributs QoS traités (Temps de réponse(Tr), Coût (c), Réputation (R), Disponibilité (D), Fiabilité(F), Sécurité (S), Throughput ou débit (Th)), la stratégie de recherche (optimisation locale ou globale), les contraintes QoS, le passage à l'échelle (ou Scalabilité) et la technique d'évaluation (simulation, collection de données choisie, etc).

Approche	**Catégorie de la méthode**	**Technique de résolution**	**Stratégie de recherche**	**Critère de QoS**	**Contraintes de QoS**	**Scalabilité**	**Technique d'évaluation**
[Zeng et al., 2004]	Méthodes exactes	Programmation entière (LIP)	Locale+Globale	C, Tr ,R ,D	Oui	Non	Prototype expérimental
[Ardagna and Pernici, 2007]		Mixed Integer Programming (MIP)	Globale	Tr, Th, D, C	Oui	Non	Simulation utilisant des données synthétiques
[Hadjila et al., 2020]		Programmation par contraintes(CP)	Locale+Globale	Tr, F, D, C,R	Oui	Oui	Utilisation de données synthétiques
[Zeyneb Yasmina et al., 2022]		Programmation par contraintes(CP)	Locale+Globale	Tr, F	Oui	Oui	Utilisation de données synthétiques, QWS Dataset [Al-Masri and Mahmoud, 2008]
[Alrifai et al., 2012]		Décomposition des contraintes de QoS	Globale	Tr, Th, D, C	Oui	Oui	QWS Dataset [Al-Masri and Mahmoud, 2008] et données synthétiques.
[Sun and Zhao, 2012]		Décomposition des contraintes de QoS	Globale	Tr, D, C	Oui	Oui	Simulation utilisant des données synthétiques
[Rosenberg et al., 2009]		Constraints Optimization Problem (COP)	Globale	Tr, Th, D, C	Oui	Non traité	Non spécifié
[Gabrel et al., 2014]		Graphe de dépendance+ LIP 0-1	Globale		Oui	Non traité	Non spécifié
[Yu et al., 2007]	Méthodes heuristiques	Algorithme WS-UHE	Globale	C, Tr, D	Oui	Non	Simulation utilisant des données synthétiques

Approche	**Catégorie de la méthode**	**technique de résolution**	**Stratégie de recherche**	**Attributs de QoS**	**Contraintes de QoS**	**Scalabilité**	**Technique d'évaluation**
[Llinás and Nagi, 2015]		Algorithme MCSP-FP	Globale	C, Tr	Oui	Oui	Simulation utilisant des données synthétiques
[Akbar et al., 2006]		Algorithme C-UHE	Globale	C, Tr	Oui	Non	Non spécifié
[Xia et al., 2011]	Méthodes heuristiques	Algorithme *optics*	Globale	C, Tr, F, Taux de succès	Oui	Non	Prototype éxpérimental
[Luo et al., 2011]		Algorithme HCE	Globale	C, Tr, D	Oui	Non	Non spécifié
[Klein et al., 2011]		Algorithme *Hill climbing*	Globale	C, Tr, D, F	Non	Non	Non spécifié
[Feng et al., 2013]		Apprentissage par renforcement	Globale	C, Tr, D, F	Oui	Non	Non spécifié
[Canfora et al., 2005]	Méthodes méta-heuristiques	Algorithme génétique	Globale	C, Tr,D, F	Oui	Non	Simulation utilisant des données synthétiques
[Xiangbing et al., 2012]		Algorithme génétique	Globale	C, Tr, D, F	Oui	Non	Non spécifié
[Seghir et al., 2019]		Algorithme de colonies d'abeilles	Globale	C, Tr, D, F, Th	Oui	Oui	WS-Dream de [Zheng et al., 2010]
[Elsayed et al., 2017]		Algorithme génétique+ *Q-learning*	Globale		Non	Non	Non spécifié
[Wang et al., 2010]		Algorithme ACO	Globale		Oui	Non	Non spécifié

Approche	**Catégorie de la méthode**	**technique de résolution**	**Stratégie de recherche**	**Attributs de QoS**	**Contraintes de QoS**	**Scalabilité**	**Technique d'évaluation**
[Wu and Zhu, 2013]		Algorithme ACO	Globale		Oui	Non	QWS Dataset [Al-Masri and Mahmoud, 2008]
[Xia et al., 2008]		Algorithme ACO	Globale		Oui	Non traité	Simulation utilisant des données synthétiques
[Yasmina and Fethallah, 2022]		Algorithme des loups gris (GWO)	Locale+Globale	Tr, F, D, C,R	Oui	Oui	Simulation utilisation de données synthétiques
[Yang et al., 2010]		Algorithme hybride (génétique + ACO)	Globale		Oui	Non traité	Simulation utilisant des données synthétiques
[Hadjila, 2014]	Méthodes méta-heuristiques	Sélection clonale	Globale	C, Tr, F, D, R	Oui	Non	Simulation utilisant des données synthétiques
[Hwang et al., 2014]		Apprentissage automatique+ Algorithme recuit simulé	Globale		Oui	Non	Données synthétiques basée sur le Dataset de [Al-Masri and Mahmoud, 2008]
[Chen and Wang, 2007]		Algorithme PSO	Globale		Oui	Oui	Non spécifié
[Liao et al., 2011]		Algorithme Niche PSO	Globale	C, Tr	Oui	Non traité	Simulation utilisant des données synthétiques
[Liu et al., 2011]		Algorithme hqpso+ théorie quantique	Globale		Oui	Non	Simulation utilisant des données synthétiques
[Zhao et al., 2012]		Algorithme de sélection clonale et PSO	Globale	C, Tr,D, F	Oui	Non traité	Prototype expérimental

Approche	**Catégorie de la méthode**	**technique de résolution**	**Stratégie de recherche**	**Attributs de QoS**	**Contraintes de QoS**	**Scalabilité**	**Technique d'évaluation**
[Esfahani et al., 2012]		Algorithme de recherche d'harmonie	Globale	C, Tr	Oui	Non	Simulation utilisant des données synthétiques
[Mohammed et al., 2014]		Algorithme de recherche d'harmonie	Globale	C, Tr,D, F, R	Oui	Non	Simulation utilisant des données synthétiques
[Kousalya et al., 2011]		Algorithme ABC	Globale	D, F, Tr, C	Oui	Non	Simulation utilisant des données synthétiques
[Amine, 2017]		Algorithme SOMA	Globale	C, Tr,D, F, R	Oui	Non	Prototype expérimental
[Khanouche et al., 2016]	Approches basées sur la dominance au sens de Pareto	Algorithme EQSA	Globale	Qualité énergétique	Non	Oui	Simulation utilisant des données synthétiques
[Chen et al., 2015]		Algorithme DPSA	Globale	Non mentionné	Oui	Non	Dataset [Zhang et al., 2011] et données synthétiques
[Trummer et al., 2014]		Algorithme A-FPTAS	Globale	Non mentionné	Oui	Non	QWS Dataset [Al-Masri and Mahmoud, 2008]
[Jin et al., 2014]		Une fonction d'utilité+ un modèle de service physique	Globale		Oui	Non	Données synthétiques basées sur le Dataset de [Alrifai et al., 2012].
[Li et al., 2010]		Algorithme multi-objectif chaos ant coloni (MOCACO)	Globale	Tr,D, R, C	Oui	Non	Simulation utilisant des données synthétiques

Approche	**Catégorie de la méthode**	**technique de résolution**	**Stratégie de recherche**	**Attributs de QoS**	**Contraintes de QoS**	**Scalabilité**	**Technique d'évaluation**
[Zhao et al., 2017]		Algorithme multi-objectif chaos génétique	Globale	Non mentionné	Oui	Non	Simulation utilisant des données synthétiques
[Al-Helal and Gamble, 2014]		Une fonction d'utilité+ un mécanisme de remplaçabilité de services	Globale	Non mentionné	Oui	Non traité	QWS Dataset [Al-Masri and Mahmoud, 2008]

Tableau 3.2 – *Catégories d'approches de composition à base de QoS [BEKKOUCHE, 2018]*

D'après le tableau comparatif 3.2, nous observons que la majorité des approches utilisent les méta-heuristiques pour résoudre le problème de composition à base de QoS. Plus particulièrement, nous remarquons un emploi massif (majoritaire) des meta-heristiques de type : algorithmes génétiques, recherche harmonique, optimisation par colonies de fourmis, optimisation par essaims particulaires (PSO) et optimisation par colonies d'abeilles.

D'autre part, nous remarquons que certaines approches combinent les heuristiques avec les meta-heuristiques pour améliorer davantage la qualité des compositions trouvées et réduire le temps de recherche [Yasmina and Fethallah, 2022].

Aussi, nous pouvons constater que la majorité des travaux adoptent des paramètres de QoS quantitatifs, et en l'occurrence nous citons : le temps de réponse, le coût, la réputation, la fiabilité et la disponibilité.

3.6 Conclusion

Dans ce chapitre, nous avons présenté quelques concepts de base sur la composition de services web et nous avons montré des travaux connexes qui appartiennent à quatre catégories (exacte, heuristique, meta-heuristique, orientée optimum de pareto). Par la suite, nous avons montré une synthèse qui évalue l'ensemble des méthodes de composition avec un ensemble de critères de comparaison couvrant la stratégie d'optimisation, l'algorithme optimisation employé, la prise en compte des contraintes globales, et le passage à l'echelle (ou scalabilité). Dans le chapitre suivant, nous raffinons cet état de l'art en présentant des approches plus spécifiques, en particulier, nous exposons les méthodes qui considérent que la QoS est incertaine ou imprécise.

Chapitre 4

Sélection des services web avec QoS incertaine

Sommaire

4.1 Introduction

La sélection des services composites est l'une des problématiques les plus fondamentales de l'architecture orientée service (SOA). Elle a fait l'objet de plusieurs études et investigations scientifiques.

Étant donné une requête sous la forme d'un workflow composé d'un ensemble de tâches abstraites, la sélection de services composés traite la question suivante : comment sélectionner efficacement un service convenable pour chaque activité, sachant que les données de QoS de chaque services fluctuent en fonction du temps et du contexte ?

En effet, chaque tâche peut avoir plusieurs instances candidats similaires de point de vue fonctionnalité et qui ont différentes caractéristiques non fonctionnelles. L'objectif de la phase de sélection est de concrétiser un workflow en un processus opérationnel qui satisfait les attribus non-fonctionnels. La figure 4.1 fournit une illustration du framework de sélection à base de QoS. Dans cette figure, chaque cercle vide modélise un service abstrait (une entité fonctionnelle), qui doit être instanciée par l'un des services concrets (qui sont présents dans des cercles pleins) [Canfora et al., 2005], [Fki, 2015].

En général, un workflow est composé d'un ensemble d'activités abstraites, et chaque activité ou tâche peut être implémentée par l'un des services concrets (disponibles dans la tâche), l'identification des services adéquats de chaque activité est faite par la l'étape de découverte de service.
L'étape de sélection trouve la meilleure instance de service de chaque activité (puisqu'il sont distincts d'un point de vue non fonctionel) et fournit un service composé exécutable.

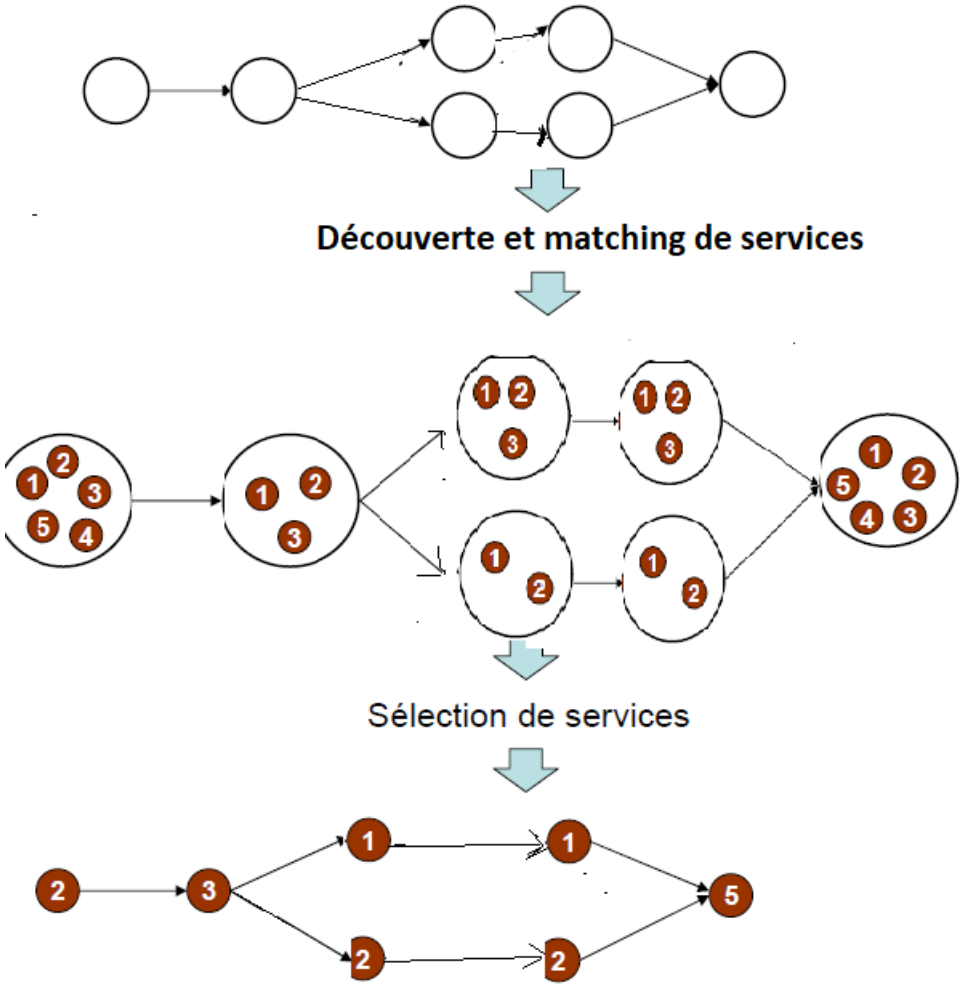

Figure 4.1 – *Sélection des services composites à base de QoS*

En présence de la multiplicité de services qui sont fonctionnellement équivalents (mais ayant des valeurs de QoS différentes et intrinsèquement incertaines ou imprécises), le problème clé à résoudre dans le contexte de la composition (avec QoS incertaine/imprécise), est de choisir optimalement les services qui satisfont le mieux les exigences de l'utilisateur (la preservation des constraintes de QoS requises).
De manière générale, nous pouvons traiter l'incertitude et l' ambiguïté/imprécision soit avec la théorie des ensembles flous (logique floue) ou la théorie des probabilités. Ce chapitre passera en revue les méthodes qui tiennent compte de l'imprécision dans la sélection (les ensembles flous) dans la section 4.2, et les méthodes qui tiennent compte de l'incertitude dans la sélection (avec les théorie des probabilités) dans la section 4.3.

La recherche d'une composition de services qui respecte les exigences de la QoS des utilisateurs est une tâche difficile. En effet, la nature dynamique de l'environnement des services web a un impact sur la QoS du service, et par conséquent, sur la QoS de la composition. D'une manière générale, la QoS peut être imprécise (définie sous forme d'intervalles et non ponctuelle) et peut varier dans le temps, et ceci représente un autre défi à adresser dans le processus de composition.

4.2 Composition dynamique de services web en tenant compte de la QoS imprécise (floue)

Dans cette section, nous nous concentrons sur les travaux de sélection et/ou de composition de services qui considérent que la QoS est :

- soit imprécise (sous forme de sous ensembles flous ou d'intervalles)
- ou elle est définie avec des termes vagues (par exemple à l'aide de mots tels que :**QoS peu coûteuse**, **trés fiable**, **environ de**, etc).

La littérature a reconnu plusieurs travaux qui adressent la QoS floue et imprecise ; dans ce cadre, nous citons l'approche de [Benouaret et al., 2012b] qui est l'une des méthodes précurseuses dans la prise en charge de l'imprécision dans les requêtes de composition de services. Les auteurs supposent que la requête est formée par plusieurs partenaires qui expriment des besoins variés sur une collection de critères de QoS (à l'aide d'intervalles ou ensembles). Pour sélectionner les services approriés, la proposition calcule les scores de matching de chaque critère appartenant à un utilisateur donné, et aprés cela, elle sélectionne les services pertinents au sens de la relation de dominance majoritaire. Cette relation favorise les services pour lesquels plus de 50% des scores de matching des utilsateurs ne sont pas dominés (au sens de la dominance floue) par le reste des candiadats.

Dans le travail de [Chouiref et al., 2016], les auteurs modélisent les attributs de QoS, les attributs contextuels, les attributs de profil d'utilisateur avec la théories des sous ensembles flous, et aprés cela, ils utilisent un algorithme de matching basé sur trois étapes : dans la première étape, ils mettent en oeuvre un filtre pour les attributs explicitement présents dans la requête de l'utilisateur. Par la suite, ce résultat est raffiné par le deuxième filtre (basé sur des règles d'inference floue) qui utilise des attributs implicites (ou déduits des régles floues), et la troisième étape agrége les deux scores précédents en un score final grace à des régles empiriques.

L'approche citée dans [Halfaoui et al., 2018] met en oeuvre la méta-heuristique de migration autonome (SOMA) et la dominance floue pour gérer et composer les workflows séquentiels. La principale faiblesse de cette méthode réside dans le choix des paramètres idéaux de SOMA.

L'approche citée dans [Benouaret et al., 2014] met en oeuvre deux régles pour comparer les critères de QoS imprécis (modélisés avec les sous-ensembles flous) : l'indice de Jaccard et le degré d'appartenance (graduel) à une implication logique. Une fois le score de comparaison est calculé, la méthode de sélection calcule d'abord un score de dominance floue de chaque service d'une tâche donnée (ceci est calculé en fonction de tous les élements de la même tâche), et par

la suite un score de pertinence de la composition est calculé en agrégeant les scores des services constituants.

4.3 Composition dynamique de services web en tenant compte de la QoS incertaine

4.3.1 Typologie des méthodes existantes

Dans le contexte de la composition de services à base de QoS incertaine, on peut distinguer trois catégories d'approches :

(a) **Les approches basées sur l'optimisation locale :** Cette classe de sélection permettent de déterminer le service candidat optimal (selon une certaine heuristique) pour chaque service abstrait de la composition, et de manière indépendante par rapport aux autres services [Hwang et al., 2014, Kim et al., 2016]. En effet, les services candidats sont classés en termes de QoS en utilisant des heuritiques variées (dominance floue agréege de [Benouaret et al., 2012b], technique SAW (Simple Additive Weighting) qui agrège les attributs de QoS en une seule valeur d'utilité [Zeleny, 1976]). La sélection de services est ensuite transformée en un problème d'optimisation mono-objectif pour déterminer le service candidat ayant la meilleure valeur d'utilité. Bien que ces approches soient très efficaces en raison de leur complexité en temps de calcul qui est polynomial [Zeng et al., 2004], elles ne garantissent en aucun cas l'aboutissement à un service composé satisfaisant les contraintes globales de QoS imposées par l'utilisateur.

(b) **Les approches basées sur l'optimisation globale :** Cette catégorie considère les contraintes de QoS au niveau du service compplexe (et non pas les composants). La recherche globale évalue et les services composés pour déterminer le meilleur workflow en termes d'une fonction objectif (qui dépend des critères de QoS et contraintes globales). L'optimisation globale est un problème NP-difficile, ce qui crée une difficulté pour aboutir à une solution optimale en un temps raisonnable [Garey and Johnson, 1979]. Beaucoup d'approches proposées préférent davantage la recherche des solutions quasi-optimales au lieu des solutions optimales.

(c) **Les approches basées sur l'optimisation hybride :** Cette catégorie considère deux étapes pour trouver les compositions pertinentes [Hadjila et al., 2020,Zeyneb Yasmina et al., 2022,Benouaret et al., 2014]. la première étape implémente un pré-filtrage des services grace à des heuristiques de nature différentes (probabilisites, floues, etc). La deuxième recherche les Top-K compositions optimales au sens d'une fonction objectif (qui peut être aussi probabiliste ou floue, etc). Cette recherche implémente une méthode exacte, une méthode heuristique, ou une meta-heuristique.

4.3.2 Etude des approches de composition à base de QoS incertaine

Comme cité précédemment, les approches de sélection peuvent être classées selon la stratégie de résolution utilisée [Jatoth et al., 2017], [Strunk, 2010],

[Khanouche et al., 2016]. Nous montrons par la suite les travaux connexes de chaque classe.

4.3.2.1 Sélection globale avec QoS incertaine

Les travaux de [Seghir et al., 2019] proposent l'approche Interval-based Multi-Objective Artificial Bee Colony (IM_ABC) pour gérer l'incertitude de la QoS dans le processus de composition de services. En effet, le problème de composition est considéré comme une optimisation multi-objectif sous contrainte d'intervalle, où une nouvelle relation de dominance sous contrainte incertaine est introduite pour traiter les fonctions objectifs à valeur d'intervalle et classer les services candidats. En plus, le contrôle de la diversité des solutions non dominées est abordé enaméliorant la distance d'encombrement originale de NSGA-II par une nouvelle définition de la distance d'intervalle.

Les travaux proposés dans [Jiang, 2007] étendent la notion traditionnelle de skyline (optimum de pareto) [Tan et al., 2001] aux skyline probabilistes afin de récupérer les services d'objets incertains les plus dominants ayant un degré d'au moins p. Pour calculer le p-skyline, deux algorithmes sont proposés pour cette tâche. Ces deux techniques sont basées sur trois heuristiques : la délimitation, l'élagage et l'affinage (bounding, pruning, and refining). Cependant, les services bruités ont un impact négatif sur la P-skyline [Jiang, 2007]. Pour pallier à ce problème, le concept de p-dominant-skyline est introduit dans [Yu and Bouguettaya, 2010] pour surmonter la limitation mentionnée. Pour calculer la p-dominant-skyline, les auteurs proposent un processus d'élagage double (dual pruning process) à travers un algorithme en deux étapes qui utilise une extension de R-tree [Guttman, 1984] appelée p-R-tree. Cette approche ne prend pas en compte le processus de composition.

La composition de services dans un environnement multi-Cloud est abordée dans [Haytamy and Omara, 2020] par un PSO modifié tenant compte de l'incertitude des attributs de QoS. Le travail avancé dans [Xu et al., 2018], intègre à la fois la théorie des ensembles flous (fuzzy set theory(FST)) et l'algorithme génétique (GA) pour résoudre la composition incertaine des services de QoS. En effet, l'approche proposée est appelée algorithme génétique flou triangulaire (Triangular Fuzzy Genetic (TGA)) où elle est utilisée dans la spécification des informations incertaines des propriétés de QoS avec des nombres flous triangulaires (QoS properties with triangular fuzzy numbers). En effet, il présente une méthode réalisable pour la normalisation de la QoS et propose une méthode pratique de défragmentation de la QoS. La fonction d'aptitude basée sur la dominance de Pareto est employée dans la TGA pour évaluer le critère et sélectionner la solution qui répond le mieux aux préférences de l'utilisateur.

Dans [Mostafa and Zhang, 2015], la modélisation des compositions de services multi-objectifs dans un environnement dynamique est traitée à l'aide du processus Partially Observable Markov Decision Process (POMDP). Pour résoudre ce problème, les méthodes d'apprentissage par renforcement multi-objectif (Multi-Objective Reinforcement Learning, MORL) sont employées par le biais de l'algorithme appelé Baseline Multi-Objective Reinforcement Learning for Service Composition. En plus, deux approches ont été proposées pour traiter la composition de services multi-objectifs à objectifs multiples. La première approche est

basée sur les préférences de l'utilisateur, et la seconde approche est basée sur un opérateur de coque convexe pour trouver les solutions optimales de Pareto. Les résultats expérimentaux ont montré une bonne efficacité pour résoudre la composition de services dans un environnement dynamique. Cependant, elle souffre d'un coût et d'une complexité élevés. Les auteurs de [Tripathy et al., 2015], proposent une représentation des systèmes basés sur les services Web (Web-Service Based System (SBS)) par le biais du graphe des services cluster. En plus, une sélection dynamique de services tenant compte du temps est efficacement en employant l'algorithme du plus court chemin de Bellman ford. En effet, la gestion des exigences dynamiques en matière de QoS est gérée en employant un schéma d'adaptation en cours d'exécution basé sur une approche de réélection de service.

Pour résoudre la composition de service sensible à la QoS, le travail dans [Elsayed et al., 2018] propose une approche hybride qui combine l'algorithme genetique et le Q-learning. L'algorithme Q-Learning est utilisé pour générer la population initiale au lieu de la génération aléatoire. En effet, les opérateurs génétiques sont exploités pour la population afin de trouver la composition optimale en termes de QoS.

4.3.2.2 Sélection locale avec QoS incertaine

[Benouaret et al., 2012a] est une autre approche de recherche qui introduit une approche améliorée basée sur la skyline. Elle aborde la QoS incertaine par le biais d'une représentation de distribution de probabilité pour sélectionner les services candidats les plus pertinents. Dans cette approche, les extensions de la relation de dominance **pos-dominance** et **nec-dominance** ont été présentées pour remplacer la p-skyline traditionnel. En outre,l'optimisation du calcul de dominance est exploitée pour surmonter le coût de calcul élevé.

Un autre travail est proposé dans [Hwang et al., 2014], pour gérer la sélection de services avec une QoS incertaine. Ce travail aborde la modélisation de la QoS fluctuante pour les services atomiques et composites. Pour identifier les meilleurs services, l'approche commence par une décomposition des contraintes globales en contraintes locales où les seuils locaux calculés sont la fraction de la contrainte globale par rapport à la moyenne des attributs de QoS de chaque classe. Ensuite, la composition est effectuée et évaluée en fonction d'une mesure objective locale. La solution obtenue est améliorée sur la base de la méthode du recuit simulé (simulated annealing method) en remplaçant certains services sélectionnés par d'autres lorsque les contraintes globales ne sont pas respectées.

Dans [Kim et al., 2016], les auteurs abordent un problème important dans la sélection de services où ils affirment que les valeurs aberrantes ou bruitées affectent la QoS des services. Dans ce contexte, les auteurs traitent des valeurs aberrantes et proposent une décomposition des contraintes globales en contraintes locales. Les contraintes locales sont ensuite utilisées pour éliminer les services candidats qui ne respectent pas ces contraintes. En plus, la sélection des services s'effectue au moyen d'un score de classement qui maximise la probabilité de satisfaire une contrainte globale de QoS d'un service composé.

4.3.2.3 Sélection hybride avec QoS incertaine

Les travaux présentés dans [Wen et al., 2014] traitent l'incertitude de la QoS en utilisant la dominance probabiliste. Dans ce contexte, le concept de capacité de dominance est introduit pour calculer les K meilleures compositions dominantes. L'approche proposée s'appuie sur deux algorithmes appelés Bounding Batch Computing (BBC) et Top-k Services Retrieving (TSR) pour sélectionner les k premiers services dominants. Les auteurs proposent aussi une approche de sélection indépendante des services composants (notée HIS) pour améliorer la précision des top-k services composés obtenus auparavant. Il est important de noter que ce travail ne tient pas en compte les contraintes globales.

L'architecture proposée dans [Yasmina et al., 2018] adopte l'heuristque de dominance probabiliste pour retenir les Top-k meilleurs services de chaque classe abstraite, par la suite une recherche globale basée sur la la recherche avec back-tracking est mise en oeuvere sur les resultats de la première phase pour retenir les Top-K compositions optimales au sens de la conformité globale de QoS.

[Hadjila et al., 2020] est un récent travail qui traite de l'incertitude de la QoS dans la composition des services Web. L'approche proposée adopte plusieurs heuristiques probabilistes/floues et une optimisation globale. L'optimisation s'appuie sur un ensemble de règles pour élaguer l'espace de recherche, telles que le classement à base d'heuristiques probabilistes, le classement basé sur la dominance floue, le classement probabiliste à base de skylines. En plus des heuristiques précédentes, les auteurs mettent en oeuvre la programmation par contraintes pour fournir les K meilleures compositions.

L'architecture proposée dans [Zeyneb Yasmina et al., 2022] adopte l'heuristque du grade majoritaire pour retenir les Top-k services pertinents de chaque classe abstraite, par la suite une recherche globale basée sur la programmation par contraintes est appliquée sur les resultats de la première phase pour retenir les Top-K compositions pertinentes au sens de la conformité globale de QoS. Dans un travail similaire à celui [Zeyneb Yasmina et al., 2022], les auteurs dans [Remaci et al., 2021], combinent l'heuristique du vote par score et la recherche avec back-tracking pour retenir les Top-K compositions quasi-optimales au sens de la conformité globale de QoS.

Dans la même ligne d'idées, [Yasmina and Fethallah, 2022] adopte l'heuristique des sous ensembles flous hesitants [Torra, 2010] pour retenir les Top-k services pertinents de chaque classe abstraite, par la suite, une recherche globale basée sur la meta-heuristique bio-inspirée des loups gris [Mirjalili et al., 2014] est mise en oeuvre pour retenir les Top-K compositions pertinentes au sens de la conformité globale de QoS.

4.4 Synthèse

La table 4.1 résume les principales approches de la sélection et/ou de la composition des services. Nous précisons si les chercheurs mettent l'accent sur la prise en charge de la QoS incertaine ou imprécise (notée QoSI), le degré de staisfaction de l'optimalité (notée OPT), la prise en compte des contraintes globales (notée GC), et la stratégie de recherche appliquée qui est soit globale (notée RG), ou locale (notée RL) ou hybride (RL et RG en même temps), la prise

en charge de la sémantique (notée S), l'utilisation des méta-heuristiques (notée MH).

Tableau 4.1 – *Comparaison des approches de sélection de services avec QoS incertaine*

méthode	QoSI	RL	RG	CG	OPT	S	MH
[Zeyneb Yasmina et al., 2022]	✓	✓	✓	✓	✓	x	x
[Yasmina et al., 2018]	✓	✓	✓	✓	✓	x	x
[Hadjila et al., 2020]	✓	✓	✓	✓	✓	x	x
[Kim et al., 2016]	✓	✓	✓	✓	✓	x	x
[Hwang et al., 2014]	✓	✓	x	✓	x	x	x
[Belouaar et al., 2017]	✓	✓	x	✓	x	x	✓
[Halfaoui et al., 2018]	x	✓	x	x	✓	x	x
[Benouaret et al., 2014]	✓	✓	✓	x	x	✓	x
[Seghir et al., 2019]	✓	x	✓	✓	x	x	✓
[Chen and Ha, 2018]	x	✓	✓	✓	✓	x	x
[Bekkouche et al., 2017]	x	x	✓	x	x	✓	✓
[Mohammed et al., 2014]	x	x	✓	x	x	x	✓
[Yasmina and Fethallah, 2022]	✓	✓	✓	✓	x	x	✓
[Etchiali et al., 2023]	✓	✓	✓	✓	✓	✓	✓

4.5 Conclusion

Nous avons montré dans le présent chapitre les differntes classes de méthodes de composition de services avec QoS incertaine. Nous avons montré les méthodes de recherche globale, locale et hybride. Dans le prochain chapitre, nous allons présenté nos méthodes qui se basent sur la classe hybride, et plus particulierement, nous montrons nos meta-heuristiques et méthodes exactes qui assurent la recherche globale.

Chapitre 5

Approches proposées pour la composition et sélection de services

Sommaire

5.1 Introduction

Dans ce chapitre, nous survolons les approches proposées pour résoudre les problématiques de sélection de services web et services IoT. Dans la première section nous présentons le problème général de composition ainsi que le cas spécifique de sélection de services web composés (ou agrégation de services). Ensuite nous introduisons deux approches permettant la sélection de services composés avec prise en compte de la QoS incertaine. Dans la troisième section, nous présentons un algorithme pour la sélection des services IoT. En conclusion, nous présentons un résumé pour nos trois contributions.

5.1.1 Motivations

La composition de services (web IoT, ou cloud) est l'une des étapes les plus cruciales de l'architecture orientée services SOA. Selon [Rao and Su, 2004], La composition de services web est «le processus qui consiste à combiner plusieurs services pour satisfaire un besoin complexe de l'‘utilisateur ». Dans cette thèse nous nous intéressons à un cas particulier de compositions qui se nomme aussi sélection de services composés. Ce dernier consiste à instancier un workflow de tâches abstraites avec des services concrets (pour chacune des tâches), tout en vérifiant les contraintes globales, et traitant des attributs de QoS incertains.

Pour résoudre efficacement cette tâche de sélection (ou instanciation/composition), nous devons adresser le grand nombre de compositions possibles. Par exemple, si nous avons n tâches abstraites (voir la figure 5.1), m services par tâche, nous aurons m^n compositions candidates. Pour gerer cette difficulté, nous proposons deux phases de sélection : une pré-sélection (ou recherche locale) dans laquelle, nous allons retenir les Top-K services (pertinents en termes de QoS) de chaque tâche. Ceci permet de reduire le nombre de services pertinents de chaque classe de m à k. La deuxième phase est appelée selection globale (ou recherche globale), elle permet de selectionner les Top-K compositions qui satisfont les contraintes globales ainsi que la conformité globale en QoS.

Dans la pratique, nous constatons que la QoS des applications SaaS (software as a service) est toujours en fluctuation et en changement permanent ; par exemple, le coût de la réservation d'une chambre d'hôtel ou d'un billet d'avion est incertain et dépend de la période (comme la saison ou le mois), des événements sociaux, et d'autres aspects contextuels. Pour comparer les services de la même classe de fonctionnalité tout en considérant les différentes réalisations des critères QoS, nous pouvons utiliser des mesures statistiques telles que la valeur moyenne de QoS ou la valeur médiane pour déduire les meilleures alternatives. Malheureusement, ces mesures peuvent ne pas être efficaces et donnent des résultats mediocres ou insatisfaisants. Par exemple, le domaine de l'apprentissage ensembliste (plus spécifiquement la méthode Adaboost [Schapire et al., 2013]) nous confirme (mathématiquement) que la moyenne pondérée de différentes prédictions (qui est un cas spécial de la valeur moyenne) peut largement dévier de la vraie prédiction si le bruit est présent dans l'ensemble d'entraînement [Dietterich, 2000], ce qui signifie que la valeur moyenne d'un échantillon peut ne pas être le représentant correct d'une série statistique si le bruit est largement présent dans les données. De plus, le théorème "limite centrale", impose des conditions strictes pour que la moyenne de l'échantillon suivra approximativement la distribution normale et convergera vers l'espérance de distribution.

Dans la même ligne d'idées, nous soulignons que la pertinence des compositions de service par rapport à la requête de l'utilisateur n'est plus un score déterministe, mais il sera spécifié comme une probabilité de satisfaire les contraintes globales QoS ; ce score est appelé la conformité globale de QoS (GQC) [Hwang et al., 2014]. En conséquence, la complexité du problème de sélection dépend du nombre de tâches, la taille de chaque tâche, et la taille du échantillon QoS (i.e., the nombre de realisations par attribut de QoS). Le GQC peut également être considéré comme l'esperance (ou moyenne) d'une variable aléatoire appelée Z, où Z fournit un résultat égal à 1 si la QoS agrégée satisfait la contrainte globale (et 0 sinon).

En outre, il est très souhaitable d'obtenir des compositions de services qui satisfont à un nombre maximal de contraintes globales en termes de QoS médiane (ce qui signifie que 50% des réalisations de la solution - d'un seul attribut de QoS - assurera les bornes de la contrainte globale). Ce critère est désigné comme le pourcentage de contraintes mondiales satisfaites (PSGC). Cette dernière mesure peut être considérée comme une alternative à la "fonction objectif" du CCG, car elle moins complexe en termes du coût de calcul. Par exemple, selon la table 5.1(où nous avons un workflow à 02 tâches et une seule contrainte globale (GC) spécifiée dans la première ligne), nous observons que seules (S1, S10) et (S2, S10) seront conservées comme solutions faisables puisque le PSGC = 100% (l'exemple comprend un seul attribut de QoS) ; nous remarquons que la somme de 20 et 26 est plus grande que 45 pour les deux paires ; cependant, les compositions restantes ne sont pas faisables, et par conséquent, le PSGC = 0 (en effet, la somme des QoS médianes ne dépasse pas la borne de la contrainte globale).

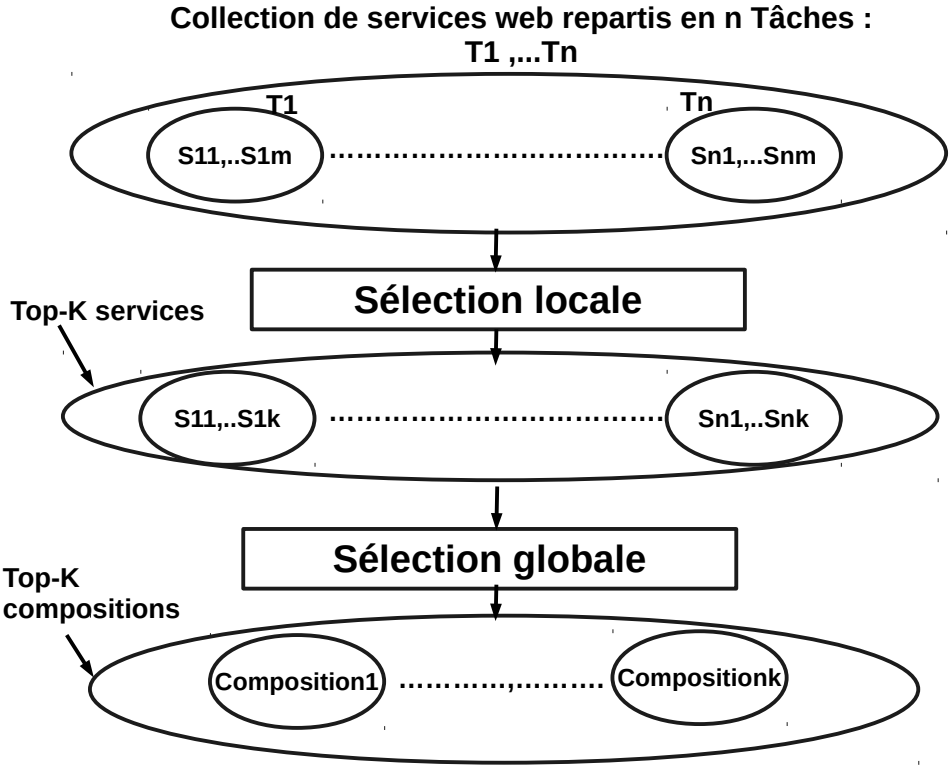

Figure 5.1 – *Processus de sélection des compositions*

Tableau 5.1 – *Services avec QoS fluctuante*

G.C : $AggregatedQoS(S_x, S_y) \geq 45$	
Tâche T1	**Tâche T2**
S_1 : $QoS(S_1) =< 5, 15, 20, 30, 70 >$	S_9 : $QoS(S_9) =< 10, 11, 15, 20, 22 >$
S_2 : $QoS(S_2) =< 6, 18, 20, 40, 90 >$	S_{10} : $QoS(S_{10}) =< 15, 18, 26, 30, 40 >$
S_3 : $QoS(S_3) =< 4, 15, 18, 25, 250 >$	S_{13} : $QoS(S_{13}) =< 3, 8, 10, 20, 30 >$

Pour faire face aux difficultées citées précédemment, nous avons adopté (voir la figure 5.1) un processus de sélection à 02 phases. La première phase emploie des heuristiques efficaces pour le classement des services des tâches du workflow (flux de travail) ; de même, la deuxième phase emploie des méthodes efficaces (peu coûteuses en temps) pour explorer les compositions des services.

5.1.2 Paramètres et notations

Pour résoudre notre problème, nous utilisons la notation indiquée dans la table 5.2 . Nous supposons que le flux de tâches de l'utilisateur est composé de n tâches séquentielles T1, T2,...,Tn, chaque tâche est réalisée par un service Si qui a r attributs QoS. Chaque critère de QoS est matérialisé par un échantillon de r réalisations (voir tableau 5.2).

Dans ce qui suit, nous présentons le formalisation mathèmatique du problème de sélection des compositions de services avec une QoS incertaine.

5.2 Spécification du problème

Dans ce travail, nous ne considérerons que les attributs QoS positifs (cad, des variables à maximiser). Pour les attributs négatifs, nous les multiplierons simplement par -1 et traiterons les nouvelles versions comme positives. Nous notons que notre flux de travail est composé de n tâches séquentielles. La QoS agrégé d'un flux de travail (avec différents modèles tels que la séquence, les boucles, le parallélisme et le choix) est présenté dans [Zheng et al., 2012], [Alrifai et al., 2012].

La mesure de conformité globale de QoS (Global QoS Conformance ou GQC) [Hwang et al., 2014] est utilisée pour classer les compositions TOPK. GQC est la probabilité que la composition des services satisfasse toutes les contraintes globales (voir l'équation 5.1). En particulier, nous disons qu'une composition C1 est meilleure qu'une autre compositions C2 si le GQC de C1 est plus élevé que celui de C2. Si C1 et C2 ont le même score GQC, alors nous les trions selon la fonction d'utilité (U (C)) qui est montrée dans l'équation 5.4, plus le score de U(C) est grand, plus le rang est bon.

Tableau 5.2 – *Variables du problème*

Paramètre	**Sémantique**
n	Le nombre de tâches
m	Le nombre de services par classe (tâche)
r	Le nombre d'attributs de QoS
cl_1, cl_2, ..., cl_n	L'ensemble de tâches ; chaque tâche contient des services SaaS avec la même fonctionalité est des valeurs de QoS différentes.
s_1 (respectivement s_2, ..., s_m)	Représente l'identifiant du service sélectionné relatif à cl_1 (respectivement cl_2, ..., cl_n).
QoS_{piju}	La valeur du peme attribut de QoS relatif à la ueme instance du service $S_i \in cl_j$.
b_1, b_2, ..., b_r	La contrainte globale de l'utilisateur (i.e., les bornes que doit satisfaire les valeurs de QoS de la composition).
w_1, ..., w_r	Les poids des attributs de QoS ; la valeur par défaut de chaque w_p est $\frac{1}{r}$.
k	La taille de la liste de sortie (nombre de compositions).

Notre but est de chercher des compositions $C(s_{w1}, \ldots, s_{wn})$ telles que GQC est maximisé :
$GQC((S_{w_1}, \cdots, S_{w_n}), (b_1, \cdots, b_r)) =$

$$\prod_{p=1}^{r} CC((S_{w_1}, \cdots, S_{w_n}), b_p) \tag{5.1}$$

Puisque nous avons supposé que les critères de QoS sont indépendents, la conformité globale de QoS est définie comme le produit de contraintes de conformité (notées CC).
Le critère CC est défini comme suit :
$CC((S_{w_1}, \cdots, S_{w_n}), b_p) =$

$$\frac{1}{l^n} \sum_{u_1=1}^{l} \ldots \sum_{u_n=1}^{l} step(aggregate(QoS_{pw_1u_1}, \cdots, QoS_{pw_nu_n}), b_p) \tag{5.2}$$

La fonction CC calcule le degré de satisfaction d'une seule contrainte globale. Finalement, la fonction binaire nommée‘Step" est definie comme suit :
$Step(Aggregate(QoS_{pw_1u_1}, \cdots, QoS_{pw_nu_n}), b_p) =$

$$\begin{cases} 1 & \text{if } Aggregate(s_{pw_1u_1}, \cdots, QoS_{pw_nu_n}) \geq b_p \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

$$U(C) = \sum_{p=1}^{r} w_p * \frac{(MedianQ_p^{'}(C) - Qmin^{'}(p))}{(Qmax^{'}(p) - Qmin^{'}(p))} \tag{5.4}$$

$$Qmin^{'}(p) = \sum_{j=1}^{n} Qmin(j,p) \tag{5.5}$$

$Qmin^{'}(p)$ est la valeur minimale de la QoS agrégée du p eme attribut de toutes les compositions possibles.

$$Qmax^{'}(p) = \sum_{j=1}^{n} Qmax(j,p) \tag{5.6}$$

$Qmax^{'}(p)$ est la valeur maximale de la QoS agrégée du p eme attribut de toutes les compositions possibles.
Les équations $Qmin(j,p), Qmax(j,p)$ sont définies comme suit :

$$Qmin(j,p) = Min_{u\in\{1,\ldots,l\},s_i\in cl_j}(QoS_{piju}) \tag{5.7}$$

$Qmin(j,p)$ est la valeur de QoS minimale du p eme attribut de tous les services relatifs à la i eme tâche.

$$Qmax(j,p) = Max_{u\in\{1,\ldots,l\},s_i\in cl_j}(QoS_{piju}) \tag{5.8}$$

$Qmax(j,p)$ est la valeur de QoS maximale du p eme attribut de tous les services relatifs à la i eme tâche.

$$MedianQ_p^{'}(C) = \sum_{j=1}^{n} Median_{u\in\{1,\ldots,l\}} QoS_{ps_jju} \tag{5.9}$$

En assumant que le critère p est positif, la contrainte globale spécifiée au sens de la valeur de QoS médiane est donnée comme suit :

$$MedianQ'_p(C) \geq b_p; \forall p \in \{1,\ldots,l\} \tag{5.10}$$

En assumant que le p eme critère est agrégé avec la somme function, $MedianQ_p^{'}(C)$ représente la QoS agrégée de C au sens des valeurs de QoS médianes de chaque composant de C (du p eme attribut).
L'équation (5.10) serve à determiner si les contraintes globales sont respectées ou non par la composition C. PSGC est le ratio des contraintes (sous la forme d'équation (5.10)) qui sont satisfaites par la composition.
Pour clarifier le calcul des formules précédentes, nous continuons avec l'exemple précédent de la table 5.1 :

- $MedianQ'_p(C =< s_1, s_{10} >) = 20 + 26 = 46 \geq 45$.
- $GQC(C) = 16/25 = 0.64$. Si $Qmin(1,1) = Qmin(2,1) = 0$ et $Qmax(1,1) = Qmax(2,1) = 300$, alors :
- $U(C) = \frac{46-0}{(600-0)} = 0.075$. La composition C est faisable. Mais si, les composants de C' sont $< s_3, s_{13} >$, alors :
- $MedianQ'_p(C' =< s3, s_{13} >) = 18 + 10 = 28 \leq 45$.
- $GQC(C') = 9/25 = 0.36$.
- $U(C') = \frac{28-0}{(600-0)} = 0.046$.
  La composition C' n'est pas faisable parce-qu'elle viole une contrainte globale.

5.3 Premiére contribution : sélection basée sur l'heuristique des intervalles majoritaires et recherche exhaustive

Pour comparer et trier les services de la même classe de fonctionnalité tout en considérant les différentes réalisations des critères QoS (nous supposons que la QoS d'une application SaaS est dynamique et incertaine) nous devons utiliser des mesures statistiques qui comblent les lacunes des mesures statistiques traditionnelles telles que la valeur moyenne de QoS ou la valeur médiane. Malheureusement, des mesures telles que la médiane ou la moyenne d'une echantillon statistique peuvent ne pas être efficaces et donner des résultats trompeurs ou insatisfaisants. Par exemple, dans le domaine de l'apprentissage d'ensembles, et plus spécifiquement la méthode Adaboost [Schapire, 2013], nous pouvons montrer mathèmatiquement que la moyenne pondérée de différentes prédictions (qui est un cas spécial de la valeur moyenne) peut largement dévier de la vraie prédiction si le bruit est présent dans l'ensemble d'entraînement [Cao et al., 2012]. Ceci signifie que la valeur moyenne d'un échantillon peut ne pas être le représentant correct d'une série si le bruit est largement présent dans les données.

En outre, selon le théorème central des limites, la moyenne de l'échantillon suivra approximativement la distribution normale et convergera à l'espérance de distribution (pour une taille infinie d'un échantillon) si un certain nombre de conditions sont déja satisfaites, sinon le moyenne n'est pas le meilleur représentant de la série (ou l'attribut de QoS). Ces conditions impliquent, entre autres, la finitude de la variance de distribution, l'échantillonnage à partir de la même distribution et l'indépendance de chaque échantillon par rapport au reste (ce qui est difficile à atteindre dans la plupart des scénarios), ainsi que l'obtention d'un nombre suffisant de données qui assure la faisabilité de l'échantillon. De même la valeur médiane n'est pas toujours décisive et crée beaucoup d'ambiguité même pour les services de données de grande taille. La table 5.1 montre l'inaptitude de la statistique de moyenne pour comparer objectivement S1 et S3. Par exemple, la moyenne de S3 est plus grande que celle de S1 (donc S3 est meilleur S1), mais il est fortement probable que la dernière réalisation de S3 (250) soit une valeur aberrante ou un bruit ; par conséquent, la comparaison precédente n'est plus fiable puisque la moyenne peut etre erronnée. De même, nous remarquons aussi l'inaptitude de la médiane pour comparer efficacement S1 et S2, puisque leur médiane est la même (20). Une analyse approfondie de S1 et S2 montre que les réalisations de S1 sont plus consistantes et moins étalées par rapport à celles de S2 (et cela veut dire que probablement S1 est meilleur que S2).

Pour adresser ses difficultés, nous proposons une heuristique appelée "intervalle majoritaire". L'idée de base est de calculer pour chaque serie statistique (qui représente un critère de QoS) un intervalle pour lequel plus de 50% de l'échantillon a une performance supèrieure ou égale à sa borne inféreieure. Aprés cela, la comparaison des deux intervalles est réalisée avec la fonction RelU (voir la figure 5.2 qui est largement utilisée en machine learning).

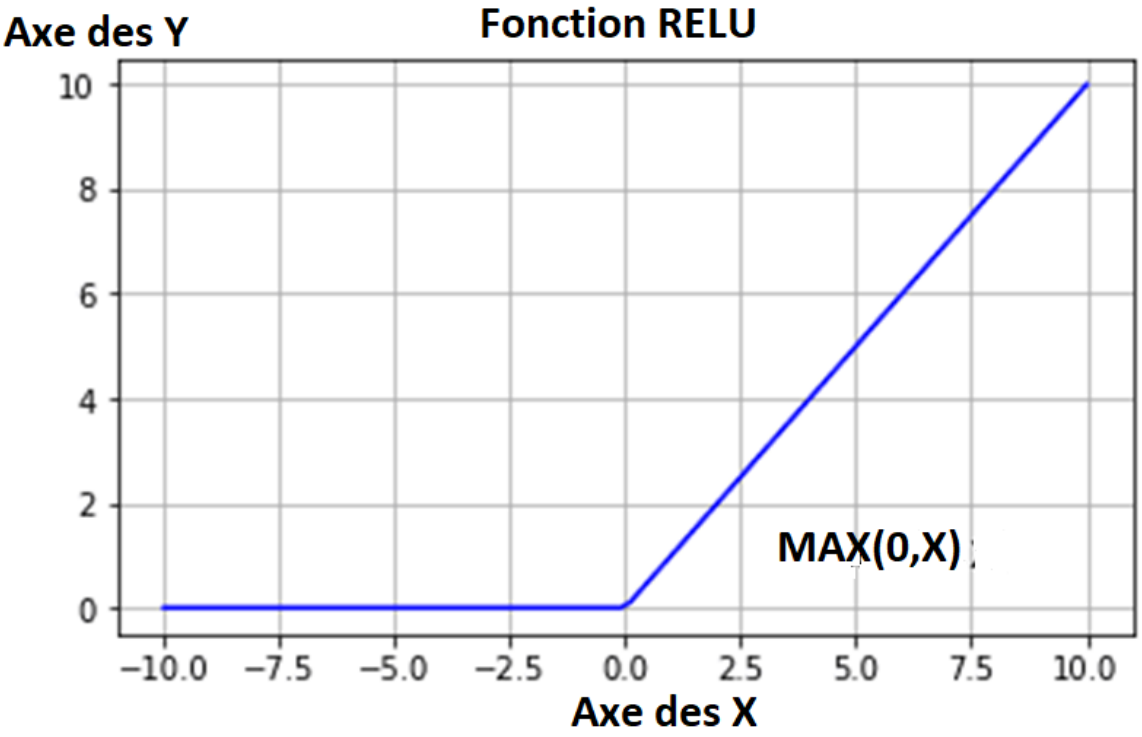

Figure 5.2 – *Fonction Relu*

5.3.1 La dominance à base d'intervalles majoritaires (notée H4)

Notre heuristique denommée "intervalle majoritaire" compare les services en se basant sur l'intervalle médian (IM).IM est calculé pour chaque attribut de QoS (de chaque service), cela veut dire que le d^{eme} attribut de chaque service S_x est représenté par un interval $[lb_{x,d}, ub_{x,d}]$). Aprés cela, nous classons les services en comparant ces intervalles médians (ou majoritaires). pour expliquer cette heuristique (que l'on note H4 dans le chapitre d'expérimentation), nous considérons les services S1 and S2 of Table 5.1. L'intervalle médian de S1 est [15, 30], et celui de S2 est [18, 40]. Pour comparer l'intervalle médian, nous utilisons la fonction presentée dans [Abdelhak et al., 2019] ; cette fonction est définie dans l'equation (5.11), et elle est nommée dominance à base d'intervalle majoritaire (DIM). (On assume que les services comparés S_x and S_y sont des membres de la tâche j et que l'attribut de QoS courant est d ; S_x est représenté par $[a_1, a_2]$, et S_y est représenté par $[b_1, b_2]$.)

$$MDIM([a_1, a_2], [b_1, b_2]) = \frac{ReLU(a_1 - b_1) + ReLU(a_2 - b_2)}{2 \times (Qmax(j, d) - Qmin(j, d))} \tag{5.11}$$

Avec ReLU [Brownlee, 2019] est la fonction d'activation utilisée en "deep learning".

Par exemple, si nous considerons $Qmax(j, d) = 300, Qmin(j, d) = 0$, alors $MID(S_1, S_2) = MID([15, 30], [18, 40]) = 0$ and $MID(S_2, S_1) = MID([18, 40], [5, 30]) = \frac{13+10}{2\times300} = 0.038$.

La dominance à base d'intervalle majoritaire est illustrée dans l'équation (5.12).

$$AMID(u,v) = \prod_{d=1}^{r} MID(u_d, v_d) \tag{5.12}$$

u_d, v_d représentent les intervalles médians des attributs de QoS comparés (ayant le rang d). La fonction "competition" est définie dans l'équation (5.13) :

$$AMIDC(S_w, S_q) = \begin{cases} 1 & \text{si } AMID(S_w, S_q) \geq AMID(S_q, S_w) \\ 0 & \text{sinon} \end{cases} \tag{5.13}$$

Dans les expérimentations, nous trions les services de chaque tâche selon l'ordre décroissant de l'équation 5.14 et nous prenons les premiers k éléments. La formule est implementée dans l'algorithme **SelectionLocale**, en remplacant la fonction nommée **heuristique** par l'équation 5.12.

$$MID\_SCORE(S_w) = \frac{1}{m-1} \sum_{w \neq q} AMIDC(S_w, S_q) \tag{5.14}$$

Algorithme 1 : Selection Locale (SL).

Input :
$< T_1, \ldots, T_n >$: les tâches abstraites.
k : La taille des listes de sortie.
Output :
$< TopKList_1, \ldots, TopKList_n >$: les listes (des TopK élements) ordonnés par ordre de merite.

```
for i ← 1 to n do
    RankedTi =<> ;
end
for j ← 1 to n do
    for y ← 1 to m do
        Score(y) =<> ;
    end
    for i ← 1 to m do
        for i' ← 1 to m do
            if (i ≠ i') then
                if (Heuristique(i) ≥ heuristique(i')) then
                    score(j) ← score(j) + 1
                end
            end
        end
    end
    RankedTi = classement − decroissant(T_i);
    TopKList_i = ToK(RankedTi)
end
return (< TopKList_1, ..., TopKList_n >)
```

La complexité de Selection Locale est $O(n \times (m^2 \times r + mlogm))$.

5.3.2 Recherche exhaustive

Algorithme 2 : RechercheExhaustive (RE)

Input :
$< TopKList_1, \ldots, TopKList_n >$: les listes d'entrées données par les heuristiques d'optimisation locale.
GC : les bornes des contraintes globale.
k : la taille de la liste des résultats.
Output :
Top-K Compositions : Les Top-K compositions qui vérifient toutes les contraintes globales et maximisent la fonction objectif GQC.(initialement, elle est vide.

```
1 TopKCompositions=<> // Le resultat est initialement vide
  for i ← 1 to k^n do
2 |  c ← NextElement(TopKList_1, ..., TopKList_n) score ← GQC(c) if
  |   (PreservationQoSGlobale(c) then
3 |  |  if (meilleurGQC(c, TopKCompositions) then
4 |  |  |  MAJ(C, TopKCompositions)
5 |  |  end
6 |  end
7 end
8 return TopKCompositions)
```

La complexité de Selection Globale est $O(k^n \times r \times klogk)$.

5.4 Deuxième contribution

Dans cette contribution, nous proposons toujours une solution à deux phases selection locale et globale. Dans la phase de recherche locale, nous proposons trois heuristques (H_1, H_2, et H_3,) basées sur les relations de dominance floue et stochastique. Dans la deuxième phase, nous proposons une meta-heuristique basée sur le comportement des chauves-souris, cette approche est nommée algorithme des chauves-souris discret.

5.4.1 Selection locale

Dans ce qui suit, nous introduisons 03 heuristiques (H_1, H_2, H_3,) qui retiennent un sous ensemble (de taille k) de chaque tâche. Ces services sont les plus performants en termes de chaque H_i. Dans ce travail, nous supposons que les valeurs les plus hautes des critères de QoS signifient que le service est bon.

5.4.1.1 Dominance floue (H1)

plusieurs alternatives permettent l'implémentation de la version floue de la Pareto dominance. [Benouaret et al., 2011, Hadjila et al., 2020, Wang and Jiang, 2007, Köppen et al., 2005]. Pour comparer 02 vecteurs u_d and v_d, (à r dimensions), nous adoptons l'implementation spécifiée dans [Köppen et al., 2005] puisqu'elle est légèrement efficace que les autres alternatives. La definition est

donnée dans (5.16). La dominance floue élementaire (EFD) compare 02 valeurs de QoS avec l'équation (5.15).

$$EFD(u_d(j), v_d(j)) = \begin{cases} 1 & \text{si } u_d(j) \geq v_d(j) \\ \frac{MIN(u_d(j), v_d(j))}{v_d(j)} & \text{sinon} \end{cases} \tag{5.15}$$

$$FD(u_d, v_d) = \prod_{i=1}^{l} EFD(u_d(i), v_d(i)) \tag{5.16}$$

Nous assumons que u_d and v_d représentent les valeurs du d eme attribut de QoS de services S and S$'$ (respectivement). Pour comparer S and S$'$ par rapport à tous les attributs de QoS, nous utilisons l'équation (5.17) (la dominance floue agrégée (AFD)).

$$AFD(u, v) = \prod_{d=1}^{r} EFD(u_d, v_d) \tag{5.17}$$

La fonction de compétition floue (FC) montrée dans l'équation (5.18) vérifie le pouvoir de dominance floue d'un service w par rapport à un autre service q.

$$FC(S_w, S_q) = \begin{cases} 1 & \text{si } AFD(S_w, S_q) \geq AFD(S_q, S_w) \\ 0 & \text{sinon} \end{cases} \tag{5.18}$$

L'équation (5.19) calcule le score de classement d'un service S_w en appliquant une comparaison avec le reste des services candidats de la tâche courante (plus le score est grand, plus le classement est bon).

$$FD\_SCORE(S_w) = \frac{1}{m-1} \sum_{w \neq q} FC(S_w, S_q) \tag{5.19}$$

Dans l'étude expérimentale, nous classons les services de chaque tâche selon l'ordre decroissant des scores donnés par l'équation (5.19) et nous retenons les k premiers élements.

Nous illustrons le principe de H1 en comparant les services S_1 et S_2 de la Table 5.1 :

En appliquant l'équation (5.17), nous obtenons

$AFD(QoS(S_1), QoS(S_2)) = FD(QoS(S_1), QoS(S_2)) = \frac{5}{6} \times \frac{15}{18} \times \frac{30}{40} \times \frac{70}{90} = 0.40.$

d'autre part :

-0cm $AFD(QoS(S_2), QoS(S_1)) = 1$ Par consequent, $FC(S_1, S_2) = 0$, $FC(S_2, S_1) = 1$, $FC(S_2, S_3) = 1$, $FD\_SCORE(S_2) = 2$.

5.4.1.2 Dominance Stochastique d'ordre zero(H2)

Cette heuristique compare les données de QoS brutes des services [Bruni et al., 2017] (voir l'équation (5.20)).

$$ZSD(u_d, v_d) = \frac{1}{l} \sum_{i=1}^{l} Step(u_d(i), v_d(i)) \tag{5.20}$$

$$Step(u_d(i), v_d(i)) = \begin{cases} 1 & \text{si } u_d(i) \geq v_d(i) \\ 0 & \text{sinon} \end{cases} \tag{5.21}$$

Pour comparer 02 services S et S′ en tenant compte de tous les attributs de QoS, nous utilisons l'équation (5.22) qui représente la dominance stochastique d'ordre 0 (aggregated zero-order stochastic dominance (AZSD)).

$$AZSD(u,v) = \prod_{d=1}^{r} ZSD(u_d, v_d) \tag{5.22}$$

Pour faire le vote majoritaire (dans une tâche donnée), nous devons comparer chaque paire de services. Pour ce faire, nous utilisons la fonction compétition montrée dans l'équation (5.23) ; abrégée aussi "AZSDC" (aggregated zero-order stochastic dominance contest) AZSDC retourne 1 si S_w domine S_q (au sens de AZSD) ; sinon elle retourne 0.

$$AZSDC(S_w, S_q) = \begin{cases} 1 & \text{si } AZSD(S_w, S_q) \geq AZSD(S_q, S_w) \\ 0 & \text{sinon} \end{cases} \tag{5.23}$$

L'équation (5.24) calcule le score de classement du service S_w en accomplissant la comparaison avec le reste des candidats d'une certaine tâche (plus le score est grand, plus le classement est bon).

$$ZSD\_SCORE(S_w) = \frac{1}{m-1} \sum_{w \neq q} AZSDC(S_w, S_q) \tag{5.24}$$

Dans les expérimentations, nous classons les services de chaque tâche selon l'ordre decroissant des scores de l'équation (5.24), nous retenons uniquement les Top-K élements.

5.4.1.3 Dominance stochastique d'ordre 1 (H3)

La dominance stochastique d'ordre 1 (H3) [Bruni et al., 2017] applique le même algorithme de l'heuristique H2 (la dominance stochastique d'ordre 0) avec une différence de remplacer les données de QoS brutes (les l realizations du critère de QoS par la distribution cumulée (CumulDistr) de l'echantillon. Cette heuristique est spécifiée dans l'équation (5.25). Si nous considérons que u_d est l'echantillon de QoS du $d$eme attribut d'un service S, alors la distribution cumulée de u_d est approximée comme suit :
$u'_d(i) = CumulDistr_i(u_d) = \sum_{t=1}^{i} \frac{1}{l}$.
En plus, nous augmentons la résolution (taille) de u'_d et nous l'initialisons à $2 \times l$; les entrées ajoutées (i′) auront un score égal à $\frac{u'_d(i-1)+u'_d(i)}{2}, \wedge i' \in [i-1, i]$.

$$\begin{aligned} FSD(u'_d, v'_d) =& ZSD(CumulDistr(u_d), CumulDistr(v_d)) \\ =& \frac{1}{2 \times l} \sum_{i=1}^{2 \times l} Step(u'_d(i), v'_d(i)) \end{aligned} \tag{5.25}$$

Nous avons les mêmes expressions utilisées dans H2 dans le reste des équations.

$$AFSD(u', v') = \prod_{d=1}^{r} FSD(u'_d, v'_d) \tag{5.26}$$

$$AFSDC(S_w, S_q) = \begin{cases} 1 & \text{si } AFSD(S_w, S_q) \geq AFSD(S_q, S_w) \\ 0 & \text{sinon} \end{cases} \tag{5.27}$$

$$FSD\_SCORE(S_w) = \frac{1}{m-1} \sum_{w \neq q} AFSDC(S_w, S_q) \tag{5.28}$$

Dans les expérimentations, nous classons les services de chaque tâche selon l'ordre decroissant des scores de l'équation (5.28), nous retenons uniquement les Top-K élements.

5.4.2 Sélection globale

5.4.2.1 Métaphore du comportement des chauves-souris

Les chauves-souris sont connues pour leurs capacités sophistiquées d'écholocation, qu'elles utilisent pour localiser et suivre leurs proies dans l'obscurité totale. Leur stratégie de chasse consiste à émettre des impulsions ultrasoniques et à écouter les échos, en ajustant leur trajectoire de vol en fonction des informations reçues. Cette poursuite par écholocation est un processus dynamique et adaptatif, les chauves-souris affinent continuellement leur approche pour attraper des proies agiles. Dans le domaine de l'intelligence artificielle, l'algorithme de la chauve-souris artificielle émule les principes du comportement des chauves-souris dans sa stratégie d'optimisation. Il utilise une population de "chauves-souris" virtuelles représentant des solutions potentielles d'un problème. Ces chauves-souris artificielles naviguent dans l'espace des solutions, et ajustent leur position en fonction des informations locales et de la meilleure solution globale trouvée jusqu'à présent. L'algorithme intègre les mouvements aleatoirs, qui représentent l'incertitude inhérente à la nature, et un mécanisme de réglage des paramètres techniques, qui s'apparente au comportement d'écholocation des vraies chauves-souris, afin d'équilibrer de manière dynamique l'exploration et l'exploitation. Il en résulte un algorithme d'optimisation qui fait preuve d'adaptabilité et d'efficacité, inspiré par les stratégies de poursuite sophistiquées observées dans le comportement naturel des chauves-souris.

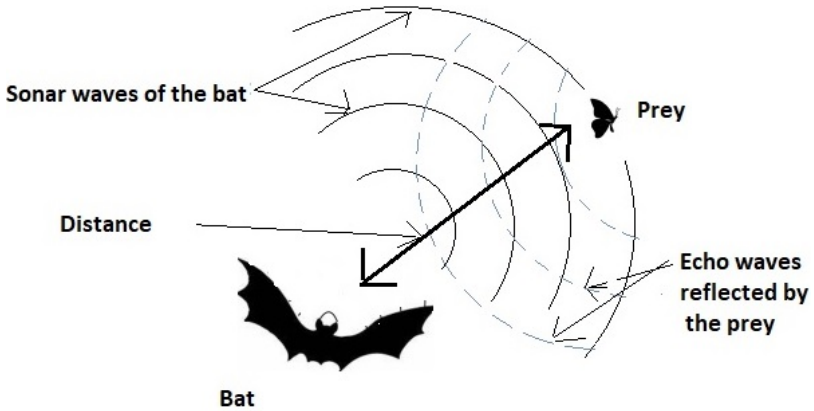

Figure 5.3 – *Comportement de chasse des chauves-souris*

5.4.2.2 Algorithme des chauves-souris discrét

Une fois que les n listes sont données par la première étape de la méthode, il est maintenant temps d'effectuer une recherche globale en composant et en évaluant les compositions des services. Pour ce faire, nous utilisons une méta-heuristique basée sur l'intelligence collèctive qui adapte l'algorithme des chauves-sourist à notre contexte discrèt. Cet algorithme d'optimisation discrète est choisi en raison de sa capacité à combiner la recherche locale et la recherche globale d'une manière harmonieuse. L'algorithme des chauves-souris (Bat algorithm) [Yang, 2010] est une méta-heuristique prometteuse pour l'optimisation continue. Sa métaphore est basée sur le comportement d'écho-loalisation des micro chauves-souris, qui peuvent varier les fréquences, l'amplitude des signaux et les taux de pulsation pour capturer la proie (voir la figure 5.3).

Algorithme 3 : Algorithm des chauve-souris discrét (ACSD).

Input :
$< TopKList_1, \ldots, TopKList_n >$: les listes d'entrées données par les heuristiques de l'optimisation locale.
GC : les bornes des contraintes globales.
k : la taille de la liste des resultats
Output :
TopKCompositions : les Top-K compositions qui satisfonts les contraintes globales en termes de GQC (initialement vide).

$A \leftarrow ones(PopSize)$
$R \leftarrow random(PopSize)$
$Alpha \leftarrow 0.8$
$Gamma \leftarrow 0.8$
for $i \leftarrow 1$ *to* $PopSize$ **do**
  $Bat_i \leftarrow RandomPosition(TopKList_1, \ldots, TopKList_n)$
  $Freq_i \leftarrow random()$
end
$Bat^* \leftarrow ArgMax_{i \in \{1,\ldots,PopSize\}}(GQC(Bat_i))$
while $(it \leq MaxIt)$ **do**
  for $i \leftarrow 1$ *to* $PopSize$ **do**
    $Bat_i \leftarrow GlobalMovement(Bat_i, Freq_i, Bat^*)$
    $Freq_i \leftarrow random()$
    if $(random() \geq R_i)$ **then**
      $neighborhoodSize \leftarrow round(k * mean_{i \in \{1,\ldots,PopSize\}}(A_i)$
      $NewPosition \leftarrow neighbor(Bat^*, neighborhoodSize)$
    end
    if $(random() \leq A_i$ *and* $GQC(NewPosition) \geq GQC(Bat_i))$ **then**
      $Bat_i \leftarrow NewPosition$
      $A_i \leftarrow Alpha \times A_i$ /*reduire l'amplitude de l'onde ultrasonique (loudness rate)*/ ;
      $R_i \leftarrow 0.1 \times (1 - exp(-Gamma \times it)$ /*augmenter la taux de pulsation (the pulse emission rate)*/ ;
    end
    if $GQC(Bat_i) \geq GQC(Bat^*)$ **then**
      $Bat^* \leftarrow Bat_i$;
    end
  end
  $it \leftarrow it + 1$
end
$TopKCompositions \leftarrow update(TopKCompositions, \{Bat^*, Bat_1, Bat_2, \ldots, Bat_n\})$
return *(TopKCompositions)*

Nous pouvons résumer l'algorithme avec 03 régles réactives : la régle du mouvement global, la régle du mouvement local, la régle de mise à jour des paramètres techniques d'amplitude et de pulsation. l'explication de ACSD est donnée comme suit :

- Pop : C'est une matrice de taille PopSize*n ; elle représente toutes les

chauves-souris virtuelles (particules). $Pop = \{Bat_1, \ldots, Bat_{PopSize}\}$.

- Bat^* : La position de la meilleure particule.
- A : elle représente l'amplitude de l'onde sonore de la particule ; c'est un vecteur de taille Popsize, et dont les élements sont compris entre 0 et 1. Il controle la taille du voisinage de la recherche locale. Il est réduit graduellement en fonction des itérations.
- Freq : C'est une matrice de taille PopSize*n ; les élement de la matrice appelés aussi frequences controlent la taille du mouvement global de l'algoithme et en particulier le pas du déplacement. Ces élements sont compris entre 0 et 1.
- R : c'est un vecteur de taille Popsize, et dont les élements sont compris entre 0 et 1. Il représente le taux de pulsation des particules. Techniquement, R controle la probabilité d'execution de la recherche locale.
- Alpha : Le facteur de réduction de A.
- Gamma : C'est un facteur qui controle l'augmentation du taux de pulsation R.
- MaxIt : Le nombre maximal d'itérations de ACSD.

Le pseudo-code de ACSD est detaillé comme suit :

- Ligne 1 : Pour chaque particule, nous initialisons l'amplitude et la pulsation, en plus les facteurs de mise à jour Alpha et Gamma sont initialisés.
- Ligne 2 : Pour chaque particule, nous initialisons sa position et sa frequence qui sera utilisée comme pas de deplacement dans le mouvement global (Lignes 6-7). $Freq_i \in [0, 1]$.
- Ligne 4 : Nous calculons la meilleure position de la population en termes de GQC.
- Lignes 5–18 : C'est la boucle principale de la meta-heuristique ; elle est constituée de MaxIt itérations.
- Lignes 5–16 : cette boucle explore toutes les particules.
- Line 7 : cette fonction crée une nouvelle composition en se déplaçant vers la meilleure solution avec un pas aléatoire. Plus particulierement, chaque composante j d'une particule i est remplacée par la valeur correspondante dans bat^* avec une probabilité égale à $Freq_i(j)$ ($Bat_i(j) = Bat^*(j)$, avec une probabilité = $Freq_i(j)$). la frequence (pas aléatoire) est changée par la suite.
- Lignes 8-9 : avec une probabilité $1 - R_i$, nous creons un voisinage centré sur la meilleure particule Bat^*.
- La taille de ce voisinage est égale à k fois la moyenne de toutes les amplitudes A_i (cette largeur est appelée "etendu") ; aprés, nous créons une nouvelle composition

  NewPosition = $(component_1, \ldots, component_n)$ comme suit :

  Pour chaque $j \in \{1, \ldots, n\}$ $component_j = successor_{Task_j}(Bat^*(j))$ avec une probabilité = $Gaussian_{mean,\sigma}(|Rank(Bat^*(j)) - Rank(successor_{Task_j}(Bat^*(j)))|)$, sachant que $Mean = Rank(Bat^*(j))$ et $\sigma = etendu/2$.

- []Par exemple, si une tâche j est constituée de cette liste ordonnée $< S9, S15, S4, S20, S2 >$, et nous assumons que $Bat^*(j) = S4, mean = Rank(Bat^*(j)) = 3$ (il est classé troisième dans la liste), et $\sigma = etendu/2 = 1$, alors le voisinage de S4, selon la ligne 7, est égal à $\{S15, S4, S20\}$. La probabilité d'avoir chacun d'eux comme une valeur pour $component_j$ est $25\%, 50\%, 25\%$, respectivement (puisque nous approximons la fonction gaussienne pour ces 03 observations).
- Dans la Ligne 10, nous acceptons la solution NewPosition (i.e., nous mettons à jour Bat_i), avec une probabilité A_i. En plus, NewPosition doit avoir un score (GQC) meilleur que celui de Bat_i.
- Dans la Ligne 11, nous réduisons l'amplitude A_i.
- Dans la Ligne 12, nous augmentons le taux de pulsation R_i pour reduire les chance d'exécuter la recherche locale dans le futur (i.e., Ligne 8).
- Dans les Lignes 14–15, nous mettons à jour la meilleure solution, si la particule courante est meilleure $Bat*$ en termes de GQC.

En fin, nous notons que ACSD a une complexité temporelle égale à $O(PopSize \times (1 + n + r \times l^n + Maxit \times (n + n \times k + r \times l^n))$. La complexité de la fonction GQC est $O(r \times l^n)$.

La figure 5.4 montre un exemple d'application d'une itération de ASCD, plus précisement, la particule notée BAT_7 est mise à jour grace aux étapes 1, 2 et 3 (initialement $BAT_7 = (S_4, S_9)$) . Dans l'étape 01, nous appliquons la régle du mouvement basé sur la mémoire (ou global) et la nouvelle valeur de $BAT_7 = (S_7, S_9)$. Dans l'étape 02, nous exécutons le mouvement local, et la nouvelle position $=(S_6, S_5)$. Dans l'étape 03, nous acceptons la position du mouvement local $(BAT_7 = (S_6, S_5))$ et nous mettons à jour A7, R7 et BAT*.

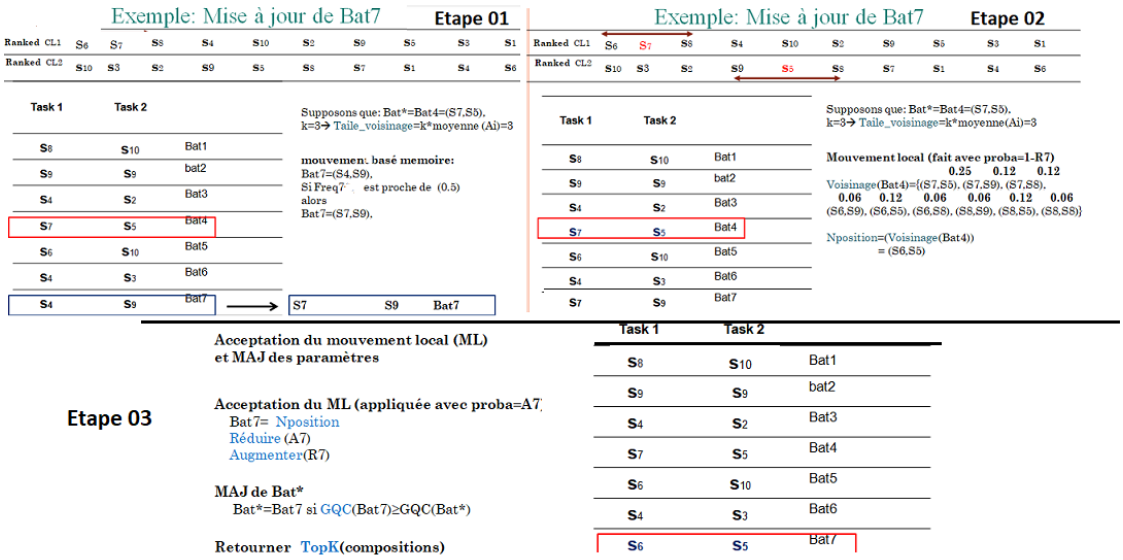

Figure 5.4 – *Exemple de déroulement de ASCD*

5.5 Troisième contribution

Dans cette section, nous supposons que l'utilisateur cherche un service IoT (par exemple un service fournissant des données météologiques) avec une specification précise pour la localisation géographique et la période de temps requise ; donc la requête (i) est constituée des attributs contextuels suivants :

- periode de temps= [debut, fin]
- localisation= coordonnées de longitude et latitude

l'utilisateur cherche aussi les meilleurs services en termes de QoS qui satisfont sa requête mentionée avant (contrainte d'espace-temps). L'algorithme proposé appelé aussi sélection de services IoT(SSIoT) calcule une similarité temporelle et spatiale entre les besoins de la requête et les caracteristiques contextuelles du service (voir les formules 5.29, 5.30). Ces deux scores de similarité sont agrégés dans une similarité contextuelle notée $SimilaritéContextuelle_{i,j}$. De même, SSIoT calcule la performance en termes de QoS de chaque service IoT notée $ScoreQoS_j$ en utilisant l'heuristique de dominance stochastique H2 qui est illustrée dans l'équation 5.24. Les deux scores $SimilaritéContextuelle_{i,j}$ et $ScoreQoS_j$ sont utilisés conjointement pour classer les services candidats et retourner les Top-K élements. Les services IoT S_j sont classés d'abord avec les scores contextuels $SimilaritéContextuelle_{i,j}$, ensuite, les élements ex aequo sont departagés avec le score $ScoreQoS_j$. Le pseudo-code de SSIoT est donné comme suit :

Algorithme 4 : SelectionServiceIoT (SSIoT).

Input :
$Collection =< S_1, \ldots, S_m >$: les services IoT de la collection de données, chaque Si est caracterisé par r critères de QoS et chaque critère de QoS est caractrisé par un echantillon de l instances (valeurs) mesurées dans le passé ($S_j.QoS'_j =< instance_1, ..., instance_l >$), de même chaque serice S_j est caractérisé par sa periode d'activité et sa zone d'activité :
$Temps(j) = [debut, fin], Loc(j) =< logitude, latitude >$
Temps(i), Loc(i) : la periode de temps et la localisation exigée par la requête i.
k : La taille des listes de sortie.
Output :
$< TopKList_i >$: la liste (des TopK élements) ordonnée selon la similarité contextuelle et la performance des données de QoS.

```
for j ← 1 to m do
    SimTemps_{i,j} ← SimilariteTemporelle(Temps(i), Temps(j))
     SimLocc_{i,j} ← SimilariteSpatiale(Loc(i), Loc(j))
     SimilaritéContextuelle_{i,j} = SimTemps_{i,j} * SimLoc_{i,j}

    ScoreQoS_j = ZSD_SCORE(S_j)
end
RankedCollection_i =
  classement − decroissant(Collection, SimilaritéContextuelle, ScoreQoS);
  TopKList_i = TopK(RankedCollection_i)
  return (TopKList_i)
```

$$SimilariteTemporelle(Temps(i), Temps(j)) =$$

$$\begin{cases} 1 & \text{si la période de la requête est incluse dans celle du service} \\ 0 & \text{si l'intersection entre les deux intervalles de temps est vide} \\ \frac{range(Temps(i) \cap Temps(j))}{range(Temps(i))} & \text{sinon} \end{cases} \tag{5.29}$$

$$SimilariteSpatiale(Loc(i), Loc(j)) =$$

$$\begin{cases} e^{-SquareDistance(Loc(i),Loc(j))} & \text{si } Distance(loc(i), loc(j)) \leq LimiteSpatiale \\ 0 & \text{sinon} \end{cases} \tag{5.30}$$

avec LimiteSpatiale est une distance fixée par le fournisseur du service IoT et elle représente la couverture physique des capteurs. Notons que la complexité temporelle de de SSIOT est $O(m^2 \times r \times l + mlog(m))$.

5.6 Conclusion

Nous avons presenté dans ce chapitre la specification formelle du problème de sélection de services composés avec QoS incertaines, en l'occurence, nous avons montré deux fonctions objectifs qui permettent l'évaluation des compositions recherchées : la GQC et le PSGC. Nous avons aussi presenté deux contributions pour la sélection des services SaaS et une troisième pour la sélection des services IoT. Les 02 premières contributions proposent quatre heuristiques pour classer localement les services de chaque tâche et retenir les Top-K élements ; par la suite, une recherche globale est appliquée pour retenir les Top-K compositions. La troisième contribution recherche les services IoT en utilisant des attributs (filtres) contextuels et des attributs de QoS.

Dans le chapitre suivant (expérimentation), nous présenterons les diffrentes évaluations de nos trois contributions en montrant les cas des succés et échec de chaque heuristique.

Chapitre 6

Implémentation et expérimentation

Sommaire

6.1 Introduction

Dans ce chapitre, nous décrivons l'architecture de notre système de sélection de services avec QoS incertaines. En premier lieu, nous montrons les différents modules qui composent le prototype proposé, que nous baptisons aussi Environnement de sélection de services composés (ESSC). Nous présentons par la suite les details de la collection de services utilisée pour évaluer l'environnement, et enfin, nous montrons les differentes évaluations de nos trois approches, ainsi que la discusions de leurs performances.

6.2 Architecture du système de sélection de services composés

La figure 6.1 montre les différents modules et intervenants dans notre système baptisé ESSC. Dans la suite, nous détaillons la fonctionalité de chaque élement de ce système.

- Le module de gestion des tâches du workflow (flux de travail) : Son objectif est d'assigner les nouveaux services à leurs tâches correspondantes (une tâche est une fonctionnalité qui est caracterisée par des entrées et des sorties connues (par exemple, la réservation d'hôtel, ou de billets d'avion). Ce composant met également à jour les tâches en modifiant/supprimant les services

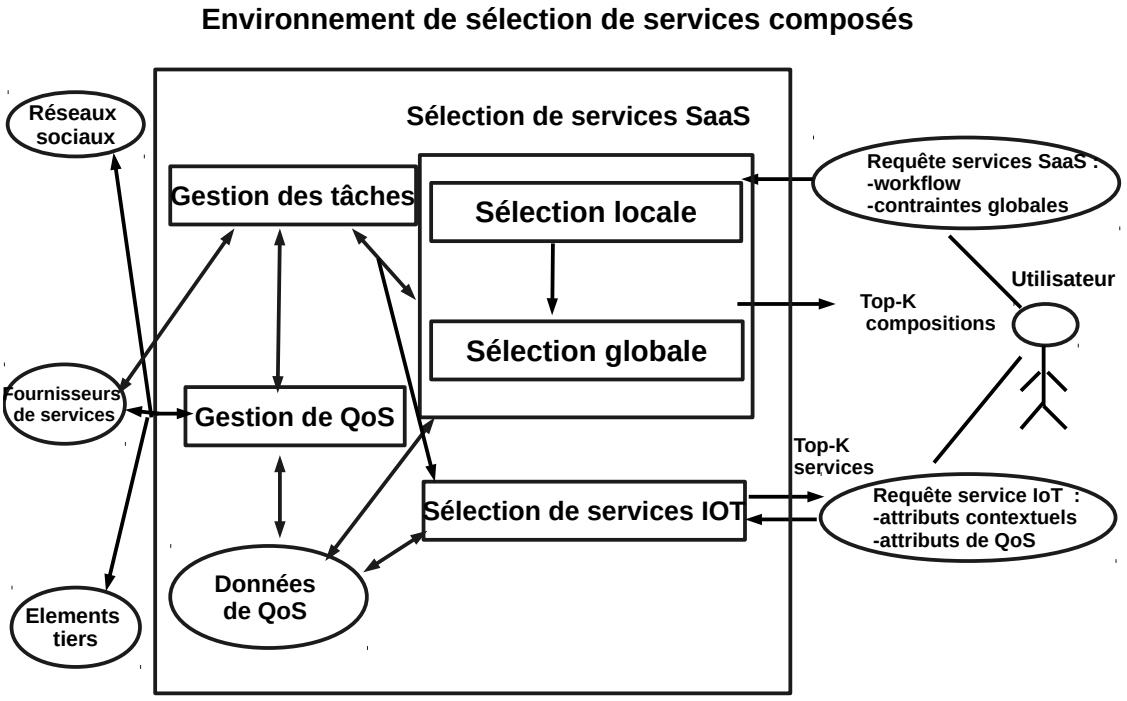

Figure 6.1 – *Architecture de l'environnement de sélection de services*

- Le module de sélection de services (sensisble à la QoS incertaine) : En partant d'un flux de travail d'un utilisateur et de l'ensemble des contraintes globales, le module de sélection permet de rechercher les Top-K compositions de services pertinentes au sens de la fonction objectif GQC. Comme mentionné dans le chapitre précédent, ce module accomplit deux étapes : une optimisation locale (ou tri) et une optimisation globale. La première phase (optimisation locale) utilise un ensemble d'heuristiques (notées H1,H2,H3, et H4) pour classer les services de chaque tâche. Le principal objectif est de réduire la taille de l'éspace de recherche en ne conservant que les premiers k services dans les phases suivantes. La deuxième phase applique l'un des deux algorithmes suivants :
  - Une recherche exhaustive (voir la première contribution du chapitre précédent)
  - Une recherche à base d'algorithme de chauves-souris (voir la deuxième contribution du chapitre précédent).
- Le module de sélection de services IOT : En partant d'un besoin exprimé sous forme de requête spatio-temporelle et un ensemble de critères de QoS à optimiser, ce module permet de rechercher les Top-K services IoT pertinenet au sens des critères contextuels, ainsi que les données de QoS les plus dominantes .
- Le module de gestion (mise à jour) de QoS : Il stocke toutes les réalisations (QoS) de tous les services dans un base de données ; les données de QoS peuvent provenir de différentes sources telles que les réseaux sociaux (par

exemple, les avis, les commentaires, la fidélité), Les sytèmes tiers (qui mesurent par exemple le débit ou la latence) et des fournisseurs de services. (par exemple le coût).

6.2.1 Outils, langages et materiels utilisés

L'environnement ESSC est developé en Java et particulier l'environnement de développement intégré *Netbeans 12.0*. Les données des Critères de QoS sont manipulés graces aux librairies Xerces at JDOM. Nous avons aussi utilisé une machine ayant un processur Core i3 avec une vitesse de 2 GHZ et 32 G.O de RAM et sous Window 10.

6.2.2 Description de la collection des services web et la distribution des critères de QoS

En s'inspirant de [Hwang et al., 2014, Kim et al., 2016], nous avons générer synthétiquement les données de QoS en utilisant une distribution de probabilité gaussienne. en particulier, nous avons utilisé la configuration suivante : moyenne = 0 and ecart type = 1. Nous avons aussi omis les valeurs negatives. Le domaine de chaque paramètre est présenté dans la table 6.1.

Tableau 6.1 – *Paramètres et notation*

Paramètre	**Sémantique**	**Domaine**	**Valeur par defaut**
n	Le nombre de tâches	{2, 5, 8}	2
m	Le nombre de services par classe (tâche)	{500, ..., 1200}	500
r	Le nombre d'attributs de QoS	{4, ..., 11}	4
l	Le nombre de realisations d'un attribut de QoS (i.e., le nombre d'instances)	{15, ..., 350}	21
k	La taille de la liste retournée	{2, 5, 10}	5
b_i	The $i$eme borne de la contrainte globale	reel positif	Pour les attributs agrégé par : la fonction somme : n $\times$ 0.6. la fonction de multiplication : 0.6^n. fonction MAX/MIN : 0.6.
w_i	Le poids du $i$eme attributs de QoS	[0, 1]	1/r

6.3 Expérimentations

6.3.1 Sélection des services (composés) SaaS

Dans chacune des expérimentations suivantes, nous avons varié un seul paramètre et nous avons maintenu le reste à leurs valeurs par défaut (voir la table 6.1). En ce qui concerne l'implementation de la dominance floue de [Hadjila et al., 2020], nous avons conservé la même configuration choisie par les auteurs pour le paramètre ε (qui est égal à 0.1). Nous avons opté pour présenter uniquement les Top-2 compositions pertinentes (en termes du GQC) dans toutes les expérimentations restantes.

Dans la figure 6.2, nous avons observé que les temps d'exécution de H1, H3, et l'heuristique de [Hadjila et al., 2020] étaitent dans les mêmes grandeurs. En outre, nous avons observé une légère augmentation du temps pour H3 puisque la pente de la courbe est proportionnelle à $2 \times l$ au lieu de l. En fin, nous notons que H2 et H4 étaient les heuristiques les plus efficaces (temporellement) puisque la pente de leurs courbes était inférieure à celle des premières alternatives.

La figure 6.3 montre que les temps de CPU de H1, H3, et l'heuristique de [Hadjila et al., 2020] sont comparables, mais les pentes de leurs courbes respectives sont différentes. En outre, nous avons observé une légère augmentation du temps CPU pour l'heuristique de [Hadjila et al., 2020] puisque sa fonction de dominance elementaire comporte plus d'operations que celle de H1. Nous notons que H4 était le plus efficace puisque la comparaison des intervalles médians ne dépend pas de l (nous supposons que le classement des vecteurs de QoS etait accompli en différé).

De manière similaire aux évvaluations précédentes, la figure 6.4 montre que les implémentations de dominance floue de [Hadjila et al., 2020], H1 et H3 avaient des temps de CPU comparables. D'autre part, les heuristiques de grade majoritaire [Zeyneb Yasmina et al., 2022] et H4 avaient un temps d'éxecution assez faible, puisque leur pente théorique n'est pas dépendante de l. Nous notons également que la courbe de H2 avait une pente presque plate, et cela est principalement dû à la faible complexité de l'équation (5.20). Il convient de noter que le principe du grade majoritaire a été initialement présenté par [Balinski and Laraki, 2007] pour le classement des candidats d'une élection. Après cela, il a été adapté par [Zeyneb Yasmina et al., 2022] pour la sélection de services Web (la méthode combine la programmation par contraintes (PC) avec les grades majoritaires).

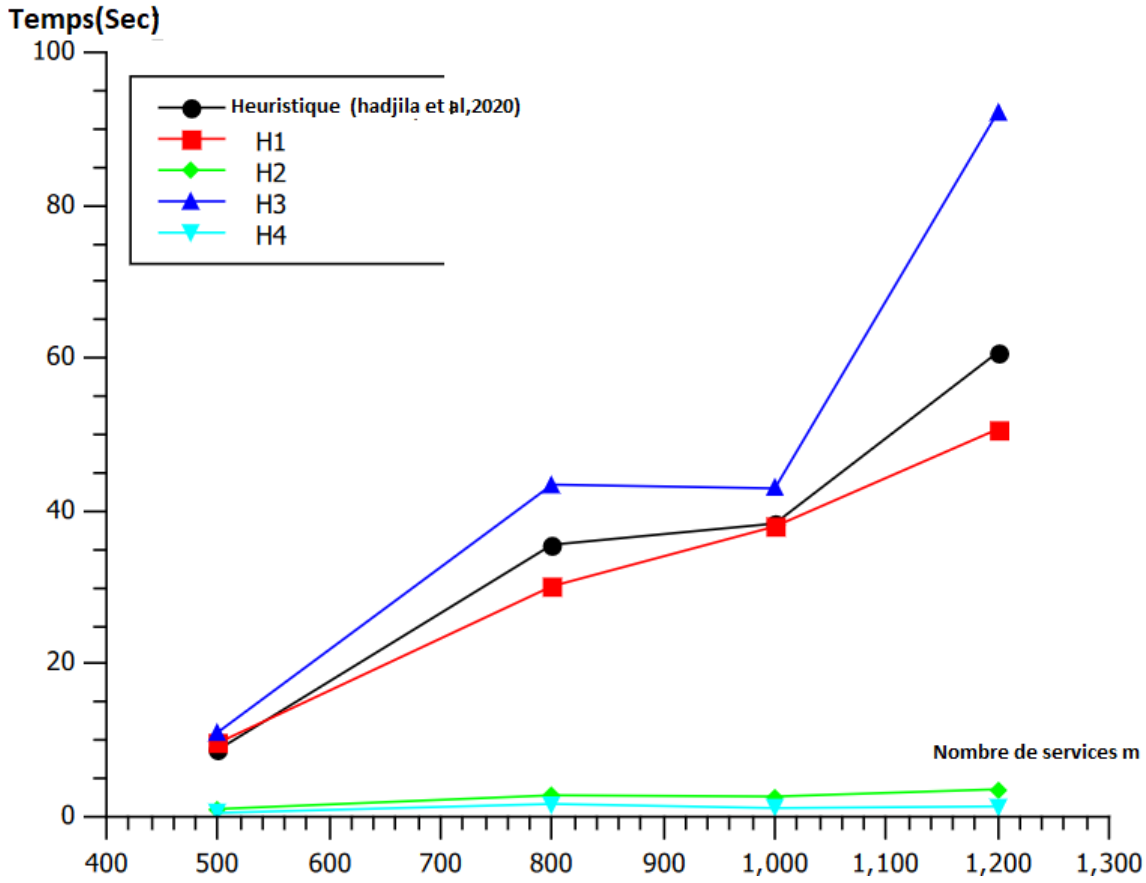

Figure 6.2 – *Temps CPU vs. m.*

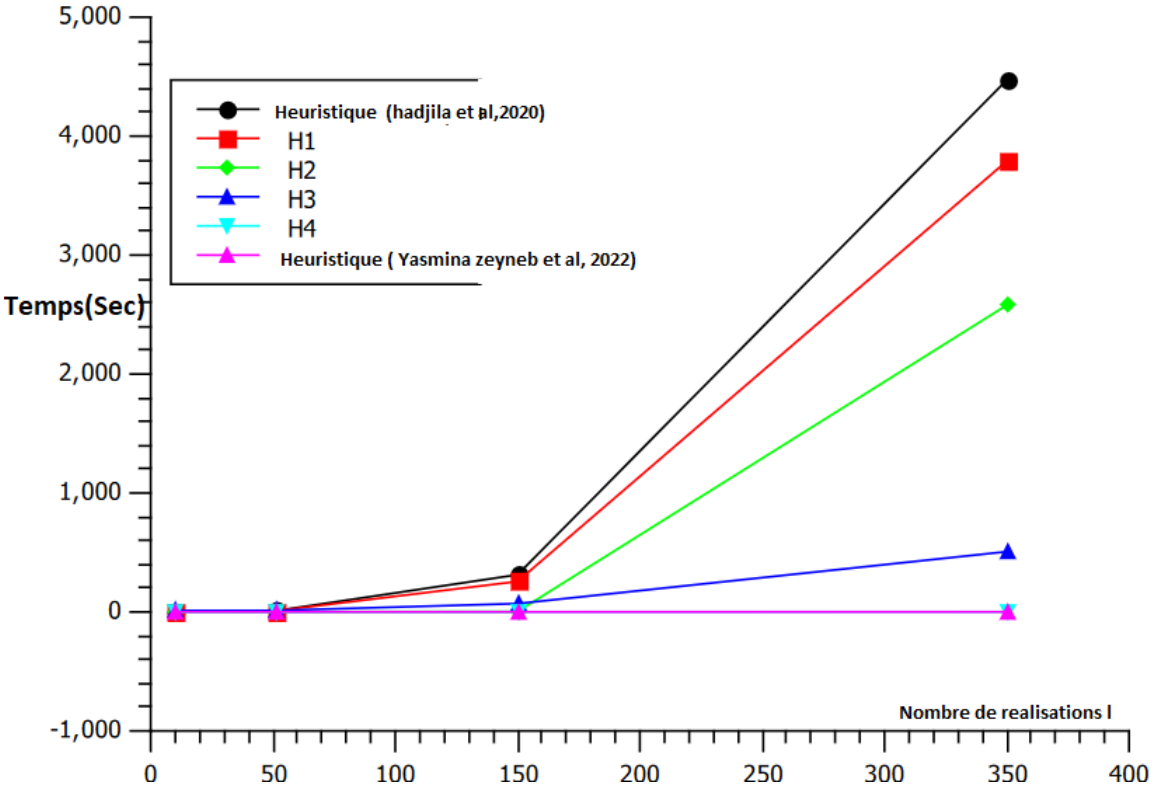

Figure 6.3 – *Temps CPU vs. l.*

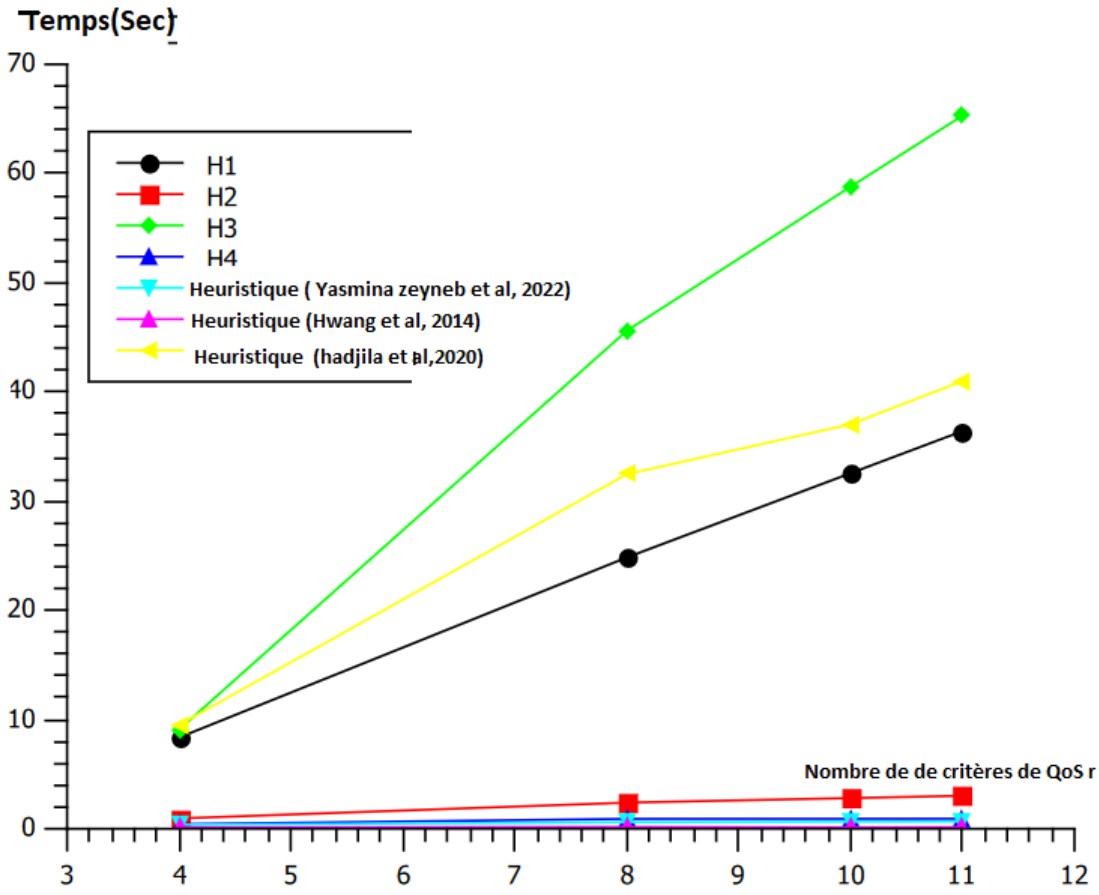

Figure 6.4 – *Temps CPU vs. r.*

Selon la figure 6.5, nous avons observé que toutes les méthodes avaient presque le même temps d'exécution jusqu'à n = 5. Au-delà de ce seuil, le temps a augmenté à différentes échelles (selon chaque alternative). Nous remarquons que la recherche exhaustive était la plus prohibitive puisqu'il y avait un nombre exponentiel de solutions candidates ; cependant, l'approche ACSD (appelée aussi en anglais Discrete Bat Algorithm ou DBA) , avec toutes les configurations, n'explore qu'un nombre polynomial de compositions candidates (mais la complexité de la GQC est toujous exponentielle). En conséquence, le rythme d'augmentation du temps était moins sévére pour les deux configurations de ASCD. En résumé, nous pouvons affirmer qu'un problème de sélection avec moins de huit tâches peut être traité efficacement par ASCD en utilisant moins de 100 particules (chauves-souris). Il convient de noter que la majorité des flux de travail (workflows) du monde réel ont moins de 10 tâches abstraites, et de ce fait, ASCD sera plus pratique pour ce type de problèmes.

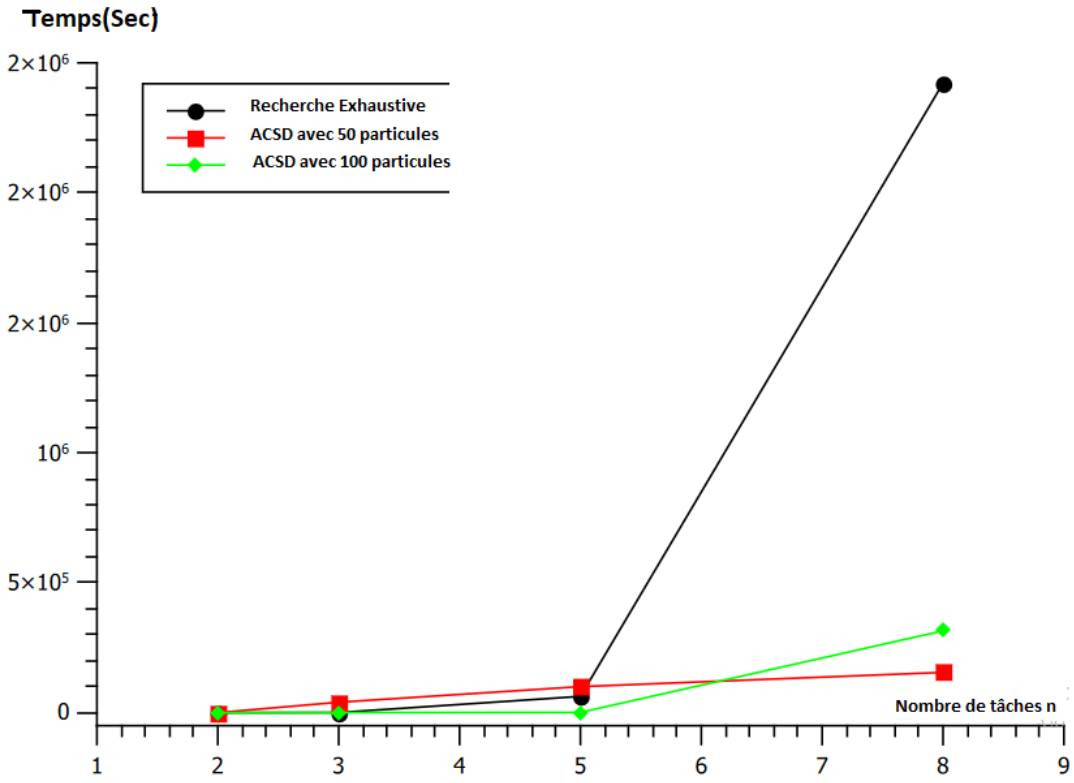

Figure 6.5 – *Temps CPU de l'agorithme des chauves-souris(ACSD) et de la recherche exhaustive.*

Tableau 6.2 – *GQC et satisfaisabilité des contraintes globales vs. r.*

Valeur	**r = 4**		**r = 8**		**r = 10**	
Modèle	**GQC**	**PSGC**	**GQC**	**PSGC**	**GQC**	**PSGC**
H_1	**0.673**	**75%**	**0.484**	**50%**	**0.500**	**40%**
	0.646	**75%**	**0.480**	**62.5%**	**0.480**	**20%**
H_2	**0.721**	**100%**	**0.515**	**37.5%**	**0.570**	**30%**
	0.664	**100%**	**0.515**	**37.5%**	**0.463**	**30%**
H_3	0.302	0%	0.393	0%	0.388	0%
	0.253	0%	0.343	0%	0.356	0%
H_4	0.562	50%	0.6628	12.5%	0.408	10%
	0.486	50%	0.524	25%	0.388	20%
Dominance floue de [Hadjila et al., 2020]	0.673	75%	0.5155	37.5%	0.500	50%
	0.633	50%	0.515	50%	0.441	10%

Pour des raisons de clarté et compacité, nous montrons uniquement la performance (avec GQC et PSGC) des Top-2 compositions dans les tables 6.2, 6.3,6.4,6.5, et 6.6. La table 6.2 démontre le comportement des heuristiques par rapport au nombre d'attributs QoS r. Nous remarquons une détérioration générale de la GQC et du PSGC (pour toutes les heuristiques) lorsque r augmente.

Cette détérioration peut être attendue puisque GQC est un produit de r probabilités (qui sont liées aux r attributs). En outre, nous observons que H2 est l'heuristique la plus performante pour toutes les valeurs de r et pour toutes les méthodes.

La table 6.3 montre le comportement des heuristiques par rapport à l'échantillon des données de QoS de taille l. De manière générale, nous remarquons que la GQC et le PSGC se dégradaient au fur et à mesure de leur croissance. Cette dégradation est logique puisque la satisfaction des contraintes globales devient de plus en plus rare au fur et à mesure de l'augmentation. Nous observons aussi que H1 et H2 étaient plus efficaces que les autres heuristiques ; plus précisément, H2 a dominé H1 pour les valeurs faibles de l (nous pouvons même obtenir un PSGC de 100%) ; cependant, H1 a dominé les autres pour les valeurs moyennes ainsi que les grandes valeurs de l. Contrairement aux heuristiques H2, H3 et H4, nous observons que H1 avait une performance stable et cohérente pour toutes les valeurs de l.

Tableau 6.3 – *GQC et satisfaisabilité des contraintes globales vs. l.*

Valeur	**l = 15**		**l = 21**		**l = 100**	
Modèle	**GQC**	**PSGC**	**GQC**	**PSGC**	**GQC**	**PSGC**
H_1	**0.673**	**75%**	**0.655**	**100%**	**0.420**	**0%**
	0.646	**75%**	**0.544**	**50%**	**0.414**	**0%**
H_2	**0.721**	**100%**	**0.704**	**50%**	**0.408**	**0%**
	0.664	**100%**	**0.655**	**100%**	**0.402**	**0%**
H_3	0.302	0%	0.343	0%	0.346	0%
	0.253	0%	0.311	0%	0.324	0%
H_4	0.562	50%	0.538	25%	0.392	0%
	0.486	50%	0.467	25%	0.390	0%
Dominance floue de [Hadjila et al., 2020]	0.673	75%	0.665	75%	0.415	0%
	0.633	50%	0.588	75%	0.411	0%

La table 6.4 présente la performance des heuristiques par rapport à m (le cardinal de la tâche). Nous observons une légère dégradation pour la GQC et le PSGC lorsque le nombre de services m augmente (pour toutes les heuristiques). Cette observation peut être due au fait que le nouvel ensemble de données (services) étendu avait des niveaux de QoS moins satisfaisants. Nous avons également observé que les heuristiques H1 et H2 étaient plus efficaces que le reste des alternatives (pour toutes les valeurs de m). .

Tableau 6.4 – *GQC et satisfaisabilité des contraintes globales vs. m.*

Valeur	**m = 500**		**m = 800**		**m = 1000**	
Modèle	**GQC**	**PSGC**	**GQC**	**PSGC**	**GQC**	**PSGC**
H_1	**0.673**	**75%**	**0.645**	**75%**	**0.549**	**50%**
	0.646	**75%**	**0.626**	**75%**	**0.491**	**25%**
H_2	**0.721**	**100%**	**0.604**	**50%**	**0.552**	**50%**
	0.664	**100%**	**0.583**	**75%**	**0.486**	**75%**
H_3	0.302	0%	0.358	0%	0.379	0%
	0.253	0%	0.311	0%	0.299	0%
H_4	0.562	50%	0.638	50%	0.506	25%
	0.486	50%	0.620	25%	0.474	25%
Dominance floue de [Hadjila et al., 2020]	0.673	75%	0.590	75%	0.551	50%
	0.633	50%	0.583	75%	0.551	50%

La table 6.5 présente la performance de l'heuristique par rapport au nombre de tâches n. Il est clairement montré que les scores donnés par toutes les heuristiques se dégradent avec l'augmentation de n, car il est plus difficile de satisfaire une contrainte composée d'une plus grande somme de variables aléatoires (selon le théorème de limite centrale, cette somme suit -selon certaines conditions- une distribution de probabilité Gaussienne avec un écart type plus restreint).

Dans la même ligne des résultats obtenus précédemment, nous remarquons que H1 a dominé le reste des heuristiques pour toutes les valeurs de n. En outre, nous notons que H2 avait un meilleur GQC et PSGC pour les valeurs faibles de n, mais ces scores se dégradaient considérablement avec l'augmentation de n.

Tableau 6.5 – *GQC et satisfaisabilité des contraintes globales vs. n.*

Valeur	**n = 2**		**n = 5**		**n = 8**	
Modèle	**GQC**	**PSGC**	**GQC**	**PSGC**	**GQC**	**PSGC**
H_1	**0.673**	**75%**	**0.665**	**50%**	**0.609**	**50%**
	0.646	**75%**	**0.656**	**50%**	**0.592**	**75%**
H_2	**0.721**	**100%**	**0.647**	**75%**	**0.224**	**0%**
	0.664	**100%**	**0.640**	**50%**	**0.219**	**0%**
H_3	0.302	0%	0.163	0%	0.132	0%
	0.253	0%	0.161	0%	0.126	0%
H_4	0.562	50%	0.557	25%	0.512	25%
	0.486	50%	0.541	25%	0.511	25%
Dominance floue de [Hadjila et al., 2020]	0.673	75%	0.563	50%	0.590	50%
	0.633	50%	0.557	50%	0.580	50%

La table 6.6 présente une comparaison entre nos contributions (la ACSD avec H1 et H2) et certaines approches de l'état de art. Il est clairement démontré que la GQC et le PSGC de H1 et H2 étaient plus efficaces que les methodes de l'état de art. Nous avons également observé que le travail de [Hwang et al., 2014] a donné les valeurs les plus faibles pour la GQC, et ceci signifie que les méthodes basées sur la sélection des seuils locaux ont de faibles performances sur les collections de données de moyenne et grande taille. Nous avons également observé que l'implementation de la dominance de Pareto floue à l'aide de l'approche citée dans [Köppen et al., 2005] était meilleure que celle de l'implementation citée dans [Hadjila et al., 2020], puisque les évaluations montrées dans les tables 6.2–6.5 ont confirmé une légère supériorité de la premiere formule.

Tableau 6.6 – *score d'utilité, satisfaisabilité des contraintes globales et GQC de toutes les méthodes (configuration par defaut).*

Heuristique	**GQC**	**US**	**PSGC**
H_1	**0.655**	**0.531**	**75%**
	0.544	**0.482**	**100%**
H_2	**0.704**	**0.511**	**50%**
	0.655	**0.531**	**100%**
H_3	0.342	0.387	0%
	0.311	0.374	0%
H_4	0.538	0.451	25%
	0.467	0.427	25%
Grade majoritaire avec PC [Zeyneb Yasmina et al., 2022]	0.703	0.519	75%
	0.631	0.527	75%
Dominance floue de [Hadjila et al., 2020]	0.665	0.516	75%
	0.588	0.514	75%
Première affectation de [Hwang et al., 2014]	0.302	0.398	0%

6.3.2 Sélection des services IoT

Dans la figure 6.6, nous décrivons le comportement de l'algorithme de classement par rapport au nombres de réalisations de QoS l. Nous observons que la croissance du temps est presque linéaire par rapport à l (surtout pour l 50). Pour évaluer la qualité des Top-k résultats, nous présentons le score de classement et le vote pondéré dans la table 6.7. On observe que les Top3 services sont équivalents en termes de Q1 (besoin temporel) et Q2 (besoin spatial) ; par conséquent, nous départageons les services de même rang grâce à l'utilisation de la dominance stochastique. Nous notons que l'augmentation de m entraînera de nombreux résultats de même score contextuel [Abdelhak et al., 2021]. De plus, nous observons que la dominance stochastique est plus discriminante et assure des classements plus satisfaisants (même si m est grand). Enfin, nous notons que la complexité de l'heuristique de dominance stochastique (voir le heuristique H2 du chapitre précédent) et le vote pondéré sont polynomiaux (en temps) et cela les rend plus adaptés aux grandes collections de services IoT.

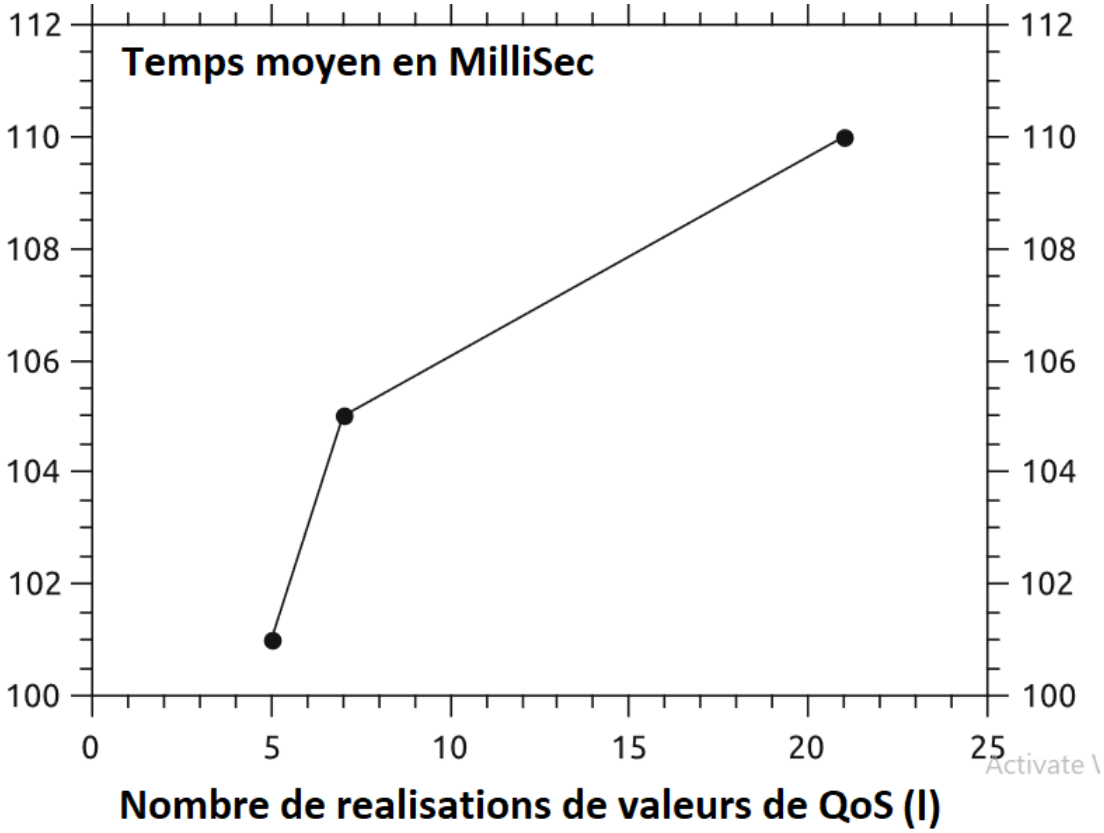

Figure 6.6 – *Temps d'exécution Vs. l*

Tableau 6.7 – *Les Top-K services IoT (pour l = 21, m = 300, k = 5)*

Position	**Score de classement basé sur la Similarité contextuelle**	**Vote agrégé (Score-QoS)**
1	0.65	0.0289
2	0.65	0.0232
3	0.65	0.0193
4	0.64	0.0189
5	0.64	0.0167

6.4 Conclusion

Dans cette étude d'expérimentale, nous avons présenté la mise en œuvre et l'évaluation de nos trois contributions. Nous avons présenté l'environnement

de composition des services qui est constitué de quatre principaux modules : le module gestion des tâches du workflow, le module de gestion des données de QoS, le module de sélection de services (composés) SaaS, et le module de sélection de services IoT. Nous avons également présenté les différents paramètres de notre collection de services. Les résultats des évaluations montrent que les heuristiques de dominance floue (H1) et stochastique d'ordre 0 (H2) sont les plus éfficaces en termes de GQC et PSGC. L'heuristique H2 a montré une légère supériorité par rapport à H1 dans les paramétres m (nombre de services par tâche), r (nombre de critères de QoS), et l (nombre d'instances d'un critère de QoS), par contre, H1 a montré une performance acceptable par rapport à toutes les heuristiques dans le paramètre n (nombre de tâches), et ceci est fortement visible pour les valeurs supérieures ou égales à 5. D'autre part, la meta-heuristique de l'algorithme des chauves-souries a montré des performances acceptables (en termes de temps, GQC, et PSGC) par rapport à la recherche exhaustive et ceci est dû à la coordination synergique entre la régle du mouvement global (basé mémoire) et la régle du mouvement local.

Chapitre 7

Conclusion et perspectives

Le développement rapide des réseaux informatiques et des middlewares a entraîné la naissance de paradigmes de développement de logiciels à base de composants autonomes appelés services. Ces services peuvent etre implémentés avec plusieurs technologies et en l'occurence nous citons les services web. Les services Web constituent une technologie clé pour bâtir des applications distribuées (et complexes) sur des réseaux à grande échelle. La composition de ces modules logiciels interopérables permettent de créer des fonctionnalités avec une valeur ajoutée. Le processus de composition doit faire face à plusieurs défis, tels que le nombre exponentiel des services Web, les fluctuations de l'environnement, l'incertitude de la QoS. L'objectif principal de cette thèse est de résoudre la composition des services Web en présence de ces défis.

7.1 Synthèse

Dans la première partie de cette thèse, nous avons présenté le paradigme orienté service et son implémentation appelée la technologie des services web. Nous avons également montré les deux concrétisations principales des services web qui sont : les services SOAP et les services REST. Par la suite, nous avons présenté les travaux existants (ou connexes) en matière de composition des services basés sur la QoS certaine et incertaine. Pour adresser le problème de la composition des services avec des fluctuations de QoS, nous le spécifions d'abord comme un problème d'optimisation combinatoire avec contraintes globales, puis nous avons détaillé les approches et les modèles développés pour résoudre cette tâche d'optimisation. Dans la deuxième partie, nous avons proposé trois méthodes pour sélectionner et composer les services avec QoS incertaine. Les deux premières approches adressent la composition de services web, alors que la troisième adresse la sélection des services IoT. La première et la deuxième contribution traitent la volumétrie de l'espace de recherche des solutions en proposant deux phases : la recherche locale et la recherche globale. Dans la première phase un certain nombre d'heuristiques ont été développées pour réduire l'espace de recherche, dans la deuxième phase, nous avons développé respectivement deux méthodes : une recherche exhaustive et une recherche basée sur la méta-heuristique des chauves-souris. Les résultats étaient plus prometteurs pour la combinaison algorithme des chauves-souris et relation de dominance stochastique d'ordre 0. Pour la troisième contribution nous avons combiné deux types de critères pour filtrer et classer les services IoT pertinents par rapport à une requête d'un client. La première étape classe (ou filtre) les

services IoT grâce à leurs fenêtres de couverture temporelle et spatiale, alors que la deuxième étape révise ce classement (pour les services ex-aequos) en introduisant la relation de dominance stochastique d'ordre 0 sur les attributs de QoS. Nos contributions ont été évaluées objectivement à l'aide d'un ensemble de données non fonctionnelles ; ces données sont générées avec des distributions de probabilités dont les paramètres sont spécifiés à l'avance. L'analyse et l'évaluation expérimentale montrent que nos heuristiques de recherche locale (en particulier la dominance floue et la dominance stochastique d'ordre 0) sont efficaces pour la présélection. De même, notre méta-heuristique développée (bat algorithm) est très prometteuse en termes de coût temporel et de conformité globale de QoS.

7.2 Perspectives

Comme perspectives à nos travaux de thèse. Nous pouvons citer les axes suivants :

- Concernant la recherche (l'optimisation) locale, nous pouvons envisager d'autres heuristiques qui s'inspirent de la dominance, telles que la L- dominance ou encore la dominance approximative. Concernant la recherche (l'optimisation) globale, nous pouvons comparer nos résultats avec des Méta euristiques récentes, telles que "spider monkey optimization algorithm" (SMO) ou "whale optimization algorithm" (WO) ou "bacrerial foraging optimisation algorithm" (BFO).
- Concernant les services en nuage (cloud), nous pouvons considérer des services IAAS en plus des services SaaS. Dans ce cas, les attributs spécifiques aux services IaaS, tels que la performance des machines, la vie privée des données ou le coût de la mobilité des données, coût des régions, doivent être spécifiés avec un modèle qui represente la logique métier du fournisseur des services cloud.
- La prolifération des services IoT exige aussi l'élaboration de modèles de composition de services IoT qui tiennent compte des propriétés contextuelles de ce genre d'applications. En particulier, nous citons la préservation de l'énergie et la prise en compte de la protection des données privées. Ce dernier critère a une importance primordiale dans le cas de scenario de E-Heath. Il convient de noter que les données échangées (celles des patients) doivent être sécurisées contre toute tentative de viol de confidentialité ou atteinte à la vie privée. Par conséquent, les applications embarquées (en l'occurrence, les services IoT) doivent tenir compte des règles de vie privée en plus des critères de QoS traditionnels.

Liste des publications

- Journaux internationaux avec comité de lecture

[1] Etchiali A, Hadjila F, Bekkouche A. An intelligent bat algorithm for web service selection with QoS uncertainty. Big Data and Cognitive Computing. 2023 Aug 10 ;7(3) :140.

- Conférences internationales avec comité de lecture

[2] Abdelhak E, Fethallah H, Mohammed M. Selecting IoT Services Under Uncertain QoS. InAdvances in Computing Systems and Applications : Proceedings of the 4th Conference on Computing Systems and Applications 2021 (pp. 220-230). Springer International Publishing.

[3] Abdelhak E, Feth-Allah H, Mohammed M. QoS uncertainty handling for an efficient web service selection. InProceedings of the 9th International Conference on Information Systems and Technologies 2019 Mar 24 (pp. 1-7).

[4] Remaci ZY, Hadjila F, Echialli A, Merzoug M. Dynamic Web Service Selection Based on Score Voting. InAdvances in Computing Systems and Applications : Proceedings of the 4th Conference on Computing Systems and Applications 2021 (pp. 185-195). Springer International Publishing.

Bibliographie

[Abdelhak et al., 2019] Abdelhak, E., Feth-Allah, H., and Mohammed, M. (2019). Qos uncertainty handling for an efficient web service selection. In *Proceedings of the 9th International Conference on Information Systems and Technologies*, pages 1–7.

[Abdelhak et al., 2021] Abdelhak, E., Fethallah, H., and Mohammed, M. (2021). Selecting iot services under uncertain qos. In *Advances in Computing Systems and Applications : Proceedings of the 4th Conference on Computing Systems and Applications*, pages 220–230. Springer.

[Akbar et al., 2006] Akbar, M. M., Rahman, M. S., Kaykobad, M., Manning, E. G., and Shoja, G. C. (2006). Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers & operations research*, 33(5) :1259–1273.

[Akkiraju et al., 2005] Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., and Verma, K. (2005). Web service semantics-wsdl-s.

[Al-Faifi et al., 2019] Al-Faifi, A., Song, B., Hassan, M. M., Alamri, A., and Gumaei, A. (2019). A hybrid multi criteria decision method for cloud service selection from smart data. *Future Generation Computer Systems*, 93 :43–57.

[Al-Helal and Gamble, 2014] Al-Helal, H. and Gamble, R. (2014). Introducing replaceability into web service composition. *IEEE Transactions on Services Computing*, 7(2) :198–209.

[Al-Masri and Mahmoud, 2008] Al-Masri, E. and Mahmoud, Q. H. (2008). Investigating web services on the world wide web. In *Proceedings of the 17th international conference on World Wide Web*, pages 795–804. ACM.

[Alrifai and Risse, 2009] Alrifai, M. and Risse, T. (2009). 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. ACM.

[Alrifai et al., 2012] Alrifai, M., Risse, T., and Nejdl, W. (2012). A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 6(2) :7.

[Alrifai et al., 2010] Alrifai, M., Skoutas, D., and Risse, T. (2010). Selecting skyline services for qos-based web service composition. In *Proceedings of the 19th international conference on World wide web*, pages 11–20.

[Alves et al., 2006] Alves, A., Arkin, A., Askary, A., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C., et al. (2006). Web services business process execution language version 2.0. public review draft, 23rd august, 2006.

[Amine, 2017] Amine, M. C. (2017). *La sélection des services web dans une composition à base de critères non fonctionnels.* PhD thesis, UNIVERSITÉ ABOU-BEKR BELKAID-TLEMCEN.

[Andrews et al., 2003] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., et al. (2003). Business process execution language for web services.

[Andrieux et al., 2007] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2007). Web services agreement specification (ws-agreement). In *Open grid forum*, volume 128, page 216. Citeseer.

[Ardagna and Pernici, 2007] Ardagna, D. and Pernici, B. (2007). Adaptive service composition in flexible processes. *IEEE Transactions on software engineering*, 33(6).

[Balinski and Laraki, 2007] Balinski, M. and Laraki, R. (2007). A theory of measuring, electing, and ranking. *Proceedings of the National Academy of Sciences*, 104(21) :8720–8725.

[Baranwal and Vidyarthi, 2016] Baranwal, G. and Vidyarthi, D. P. (2016). A cloud service selection model using improved ranked voting method. *Concurrency and Computation : Practice and Experience*, 28(13) :3540–3567.

[Barkat et al., 2021] Barkat, A., Kazar, O., and Seddiki, I. (2021). Framework for web service composition based on qos in the multi cloud environment. *International Journal of Information Technology*, 13 :459–467.

[Barros et al., 2005] Barros, A., Dumas, M., and Oaks, P. (2005). Standards for web service choreography and orchestration : Status and perspectives. In *International Conference on Business Process Management*, pages 61–74. Springer.

[BEKKOUCHE, 2018] BEKKOUCHE, A. (2018). *Vers une composition automatique des services web sémantiques.* PhD thesis, Université de Tlemcen-Abou Bekr Belkaid.

[Bekkouche et al., 2017] Bekkouche, A., Benslimane, S. M., Huchard, M., Tibermacine, C., Hadjila, F., and Merzoug, M. (2017). Qos-aware optimal and automated semantic web service composition with user's constraints. *Service Oriented Computing and Applications*, 11(2) :183–201.

[Bellwood et al., 2002] Bellwood, T., Capell, S., Clement, L., Colgrave, J., Dovey, M., Feygin, D., Kochman, A., Macias, P., Novotny, M., Paolucci, M., et al. (2002). Universal description, discovery and integration specification (uddi) 3.0. *Online : http ://uddi. org/pubs/uddi-v3. 00-published-20020719. htm*, 10.

[Belouaar et al., 2017] Belouaar, H., Kazar, O., and Rezeg, K. (2017). Web service selection based on topsis algorithm. In *2017 International Conference on Mathematics and Information Technology (ICMIT)*, pages 177–182. IEEE.

[Benatallah et al., 2005] Benatallah, B., Dijkman, R. M., Dumas, M., and Maamar, Z. (2005). Service composition : Concepts, techniques. *Service-Oriented Software System Engineering : Challenges and Practices*, page 48.

[Benouaret et al., 2011] Benouaret, K., Benslimane, D., and Hadjali, A. (2011). On the use of fuzzy dominance for computing service skyline based on qos. In *2011 IEEE International Conference on Web Services*, pages 540–547. IEEE.

[Benouaret et al., 2012a] Benouaret, K., Benslimane, D., and Hadjali, A. (2012a). Selecting skyline web services from uncertain qos. In *2012 IEEE Ninth International Conference on Services Computing*, pages 523–530. IEEE.

[Benouaret et al., 2014] Benouaret, K., Benslimane, D., Hadjali, A., Barhamgi, M., Maamar, Z., and Sheng, Q. Z. (2014). Web service compositions with fuzzy preferences : A graded dominance relationship-based approach. *ACM Transactions on Internet Technology (TOIT)*, 13(4) :1–33.

[Benouaret et al., 2012b] Benouaret, K., Sacharidis, D., Benslimane, D., and Hadjali, A. (2012b). Majority-rule-based web service selection. In *Web Information Systems Engineering-WISE 2012 : 13th International Conference, Paphos, Cyprus, November 28-30, 2012. Proceedings 13*, pages 689–695. Springer.

[Bhattacharya and Chatterjee, 2014] Bhattacharya, A. and Chatterjee, T. (2014). *Goal Programming Based Multi-Objective Optimization Techniques of Task Allocation in Distributed Environment*. Lulu. com.

[Bieberstein, 2006] Bieberstein, N. (2006). *Service-oriented architecture compass : business value, planning, and enterprise roadmap*. FT Press.

[Branke, 2008] Branke, J. (2008). *Multiobjective optimization : Interactive and evolutionary approaches*, volume 5252. Springer Science & Business Media.

[Brans et al., 1984] Brans, J. P., Mareschal, B., Vincke, P., et al. (1984). Promethee : A new family of outranking methods in multicriteria analysis. *Operational research*, 84 :477–490.

[Brownlee, 2019] Brownlee, J. (2019). A gentle introduction to the rectified linear unit (relu). *Machine learning mastery*, 6.

[Bruni et al., 2017] Bruni, R., Cesarone, F., Scozzari, A., and Tardella, F. (2017). On exact and approximate stochastic dominance strategies for portfolio selection. *European Journal of Operational Research*, 259(1) :322–329.

[Canfora et al., 2005] Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2005). An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1069–1075. ACM.

[Cao et al., 2012] Cao, J., Kwong, S., and Wang, R. (2012). A noise-detection based adaboost algorithm for mislabeled data. *Pattern Recognition*, 45(12) :4451–4465.

[Cardoso et al., 2004] Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K. (2004). Quality of service for workflows and web service processes. *Journal of web semantics*, 1(3) :281–308.

[Casati and Shan, 2002] Casati, F. and Shan, M.-C. (2002). Event-based interaction management for composite e-services in eflow. *Information Systems Frontiers*, 4(1) :19–31.

[Cauvet and Guzelian, 2008] Cauvet, C. and Guzelian, G. (2008). Business process modeling : A service-oriented approach. In *Hawaii international conference on system sciences, proceedings of the 41st annual*, pages 98–98. IEEE.

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

[Chen and Ha, 2018] Chen, L. and Ha, W. (2018). Reliability prediction and qos selection for web service composition. *International Journal of Computational Science and Engineering*, 16(2) :202–211.

[Chen and Wang, 2007] Chen, M. and Wang, Z.-w. (2007). An approach for web services composition based on qos and discrete particle swarm optimization. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, volume 2, pages 37–41. IEEE.

[Chen et al., 2015] Chen, Y., Huang, J., Lin, C., and Hu, J. (2015). A partial selection methodology for efficient qos-aware service composition. *IEEE Transactions on Services Computing*, 8(3) :384–397.

[Chinnici et al., 2007] Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 1 : Core language. *W3C recommendation*, 26 :19.

[Chollet, 2009] Chollet, S. (2009). *Orchestration de services hétérogènes et sécurisés*. PhD thesis, Université Joseph-Fourier-Grenoble I.

[Chouiref et al., 2016] Chouiref, Z., Belkhir, A., Benouaret, K., and Hadjali, A. (2016). A fuzzy framework for efficient user-centric web service selection. *Applied Soft Computing*, 41 :51–65.

[Comes et al., 2010] Comes, D., Baraki, H., Reichle, R., Zapf, M., and Geihs, K. (2010). Heuristic approaches for qos-based service selection. In *International Conference on Service-Oriented Computing*, pages 441–455. Springer.

[Dahan, 2021] Dahan, F. (2021). An effective multi-agent ant colony optimization algorithm for qos-aware cloud service composition. *IEEE Access*, 9 :17196–17207.

[Dahan et al., 2019] Dahan, F., Mathkour, H., and Arafah, M. (2019). Two-step artificial bee colony algorithm enhancement for qos-aware web service selection problem. *IEEE Access*, 7 :21787–21794.

[D'Ambrogio and Bocciarelli, 2007] D'Ambrogio, A. and Bocciarelli, P. (2007). A model-driven approach to describe and predict the performance of composite services. In *Proceedings of the 6th international workshop on Software and performance*, pages 78–89.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2) :182–197.

[Dietterich, 2000] Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees : Bagging, boosting, and randomization. *Machine learning*, 40 :139–157.

[Do Prado et al., 2013] Do Prado, P. F., Nakamura, L. H., Estrella, J., Santana, M. J., and Santana, R. H. (2013). A performance evaluation study for qos-aware web services composition using heuristic algorithms. In *ICDS 2013, The Seventh International Conference on Digital Society*, pages 53–58.

[Dorigo et al., 2006] Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4) :28–39.

[Dustdar and Papazoglou, 2008] Dustdar, S. and Papazoglou, M. P. (2008). Services and service composition–an introduction (services und service komposition–eine einführung). *IT-Information Technology*, 50(2) :86–92.

[Dustdar and Schreiner, 2005] Dustdar, S. and Schreiner, W. (2005). A survey on web services composition. *International journal of web and grid services*, 1(1) :1–30.

[Elsayed et al., 2017] Elsayed, D. H., Nasr, E. S., Alaa El Din, M., and Gheith, M. H. (2017). A new hybrid approach using genetic algorithm and q-learning for qos-aware web service composition. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 537–546. Springer.

[Elsayed et al., 2018] Elsayed, D. H., Nasr, E. S., El Ghazali, A. E. D. M., and Gheith, M. H. (2018). A new hybrid approach using genetic algorithm and q-learning for qos-aware web service composition. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017*, pages 537–546. Springer.

[Erl, 2008] Erl, T. (2008). *Soa : principles of service design*, volume 1. Prentice Hall Upper Saddle River.

[Esfahani et al., 2012] Esfahani, P. M., Habibi, J., and Varaee, T. (2012). Application of social harmony search algorithm on composite web service selection based on quality attributes. In *Genetic and Evolutionary Computing (ICGEC), 2012 Sixth International Conference on*, pages 526–529. IEEE.

[Etchiali et al., 2023] Etchiali, A., Hadjila, F., and Bekkouche, A. (2023). An intelligent bat algorithm for web service selection with qos uncertainty. *Big Data and Cognitive Computing*, 7(3) :140.

[Fan et al., 2018] Fan, S.-L., Yang, Y.-B., and Wang, X.-X. (2018). Efficient web service composition via knapsack-variant algorithm. In *Services Computing–SCC 2018 : 15th International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings 15*, pages 51–66. Springer.

[Fekih et al., 2016] Fekih, H., Mtibaa, S., and Bouamama, S. (2016). User-centric web services composition approach based on swarm intelligence. In *2016 IEEE 18th International Conference on High Performance Computing and Communications ; IEEE 14th International Conference on Smart City ; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1087–1094. IEEE.

[Feng et al., 2013] Feng, L.-B., Obayashi, M., Kuremoto, T., Kobayashi, K., and Watanabe, S. (2013). Qos optimization for web services composition

based on reinforcement learning. *Int. J. Innov. Comput., Inf. Control*, 9(6) :2361–2376.

[FethAllah, 2014] FethAllah, H. (2014). Composition et interopération des services web sémantiques. *These de*.

[Fethallah et al., 2012] Fethallah, H., Chikh, M. A., Mohammed, M., and Zineb, K. (2012). Qos-aware service selection based on swarm particle optimization. In *2012 International Conference on Information Technology and e-Services*, pages 1–6. IEEE.

[Fki, 2015] Fki, E. (2015). *Sélection et composition flexible basée services abstraits pour une meilleure adaptation aux intentions des utilisateurs*. PhD thesis, Univerisité Toulouse 1 Capitole.

[Gabrel et al., 2014] Gabrel, V., Manouvrier, M., and Murat, C. (2014). Optimal and automatic transactional web service composition with dependency graph and 0-1 linear programming. In *International Conference on Service-Oriented Computing*, pages 108–122. Springer.

[Gao et al., 2006] Gao, A., Yang, D., Tang, S., and Zhang, M. (2006). Qos-driven web service composition with inter service conflicts. *Frontiers of WWW Research and Development-APWeb 2006*, pages 121–132.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). A guide to the theory of np-completeness. *WH Freemann, New York*, 70.

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

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

[Glover, 1989] Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, 1(3) :190–206.

[Gudgin et al., 2003] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., Karmarkar, A., and Lafon, Y. (2003). Simple object access protocol (soap) 1.2. *World Wide Web Consortium*.

[Guttman, 1984] Guttman, A. (1984). R-trees : A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57.

[Hadjila, 2014] Hadjila, F. (2014). *Composition et interopération des services web sémantiques.* PhD thesis.

[Hadjila et al., 2019] Hadjila, F., Belabed, A., and Merzoug, M. (2019). Flexible service discovery based on multiple matching algorithms. *International Journal of Web Engineering and Technology*, 14(4) :315–340.

[Hadjila et al., 2020] Hadjila, F., Belabed, A., and Merzoug, M. (2020). Efficient web service selection with uncertain qos. *International Journal of Computational Science and Engineering*, 21(3) :470–482.

[Halfaoui et al., 2015] Halfaoui, A., Hadjila, F., and Didi, F. (2015). Qos-aware web services selection based on fuzzy dominance. In *Computer Science and*

Its Applications : 5th IFIP TC 5 International Conference, CIIA 2015, Saida, Algeria, May 20-21, 2015, Proceedings 5, pages 291–300. Springer.

[Halfaoui et al., 2018] Halfaoui, A., Hadjila, F., and Didi, F. (2018). Qos-aware web service selection based on self-organising migrating algorithm and fuzzy dominance. *International Journal of Computational Science and Engineering*, 17(4) :377–389.

[Han et al., 2009] Han, Y., Wang, J., and Zhang, P. (2009). Business-oriented service modeling : A case study. *Simulation Modelling Practice and Theory*, 17(8) :1413–1429.

[Haytamy and Omara, 2020] Haytamy, S. and Omara, F. (2020). Enhanced qos-based service composition approach in multi-cloud environment. In *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, pages 33–38. IEEE.

[Hosseinzadeh et al., 2020] Hosseinzadeh, M., Tho, Q. T., Ali, S., Rahmani, A. M., Souri, A., Norouzi, M., and Huynh, B. (2020). A hybrid service selection and composition model for cloud-edge computing in the internet of things. *IEEE Access*, 8 :85939–85949.

[Huhns and Singh, 2005] Huhns, M. N. and Singh, M. P. (2005). Service-oriented computing : Key concepts and principles. *IEEE Internet computing*, 9(1) :75–81.

[Hwang et al., 2014] Hwang, S.-Y., Hsu, C.-C., and Lee, C.-H. (2014). Service selection for web services with probabilistic qos. *IEEE transactions on services computing*, 8(3) :467–480.

[Jatoth et al., 2017] Jatoth, C., Gangadharan, G., and Buyya, R. (2017). Computational intelligence based qos-aware web service composition : A systematic literature review. *IEEE Transactions on Services Computing*, 10(3) :475–492.

[Jatoth et al., 2019] Jatoth, C., Gangadharan, G., Fiore, U., and Buyya, R. (2019). Selcloud : a hybrid multi-criteria decision-making model for selection of cloud services. *Soft Computing*, 23 :4701–4715.

[Jiang, 2007] Jiang, B. (2007). *Probabilistic skylines on uncertain data*. PhD thesis, UNSW Sydney.

[Jin et al., 2014] Jin, X., Chun, S., Jung, J., and Lee, K.-H. (2014). Iot service selection based on physical service model and absolute dominance relationship. In *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*, pages 65–72. IEEE.

[Jordan et al., 2007] Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., et al. (2007). Web services business process execution language version 2.0. *OASIS standard*, 11(120) :5.

[Karim et al., 2011] Karim, R., Ding, C., and Chi, C.-H. (2011). An enhanced promethee model for qos-based web service selection. In *2011 IEEE international conference on services computing*, pages 536–543. IEEE.

[Kavantzas et al., 2005] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., and Barreto, C. (2005). Web services choreography description language version 1.0. *W3C candidate recommendation*, 9 :290–313.

[Keller and Ludwig, 2003] Keller, A. and Ludwig, H. (2003). The wsla framework : Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11 :57–81.

[Kellert and Toumani, 2003] Kellert, P. and Toumani, F. (2003). Les web services sémantiques. *Web sémantique, Action spéci_que*, 32.

[Khanouche et al., 2016] Khanouche, M. E., Amirat, Y., Chibani, A., Kerkar, M., and Yachir, A. (2016). Energy-centered and qos-aware services selection for internet of things. *IEEE Transactions on Automation Science and Engineering*, 13(3) :1256–1269.

[Kim et al., 2016] Kim, M., Oh, B., Jung, J., and Lee, K.-H. (2016). Outlier-robust web service selection based on a probabilistic qos model. *International Journal of Web and Grid Services*, 12(2) :162–181.

[Klein et al., 2011] Klein, A., Ishikawa, F., and Honiden, S. (2011). Efficient heuristic approach with improved time complexity for qos-aware service composition. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 436–443. IEEE.

[Köppen et al., 2005] Köppen, M., Vicente-Garcia, R., and Nickolay, B. (2005). Fuzzy-pareto-dominance and its application in evolutionary multi-objective optimization. In *International conference on evolutionary multi-criterion optimization*, pages 399–412. Springer.

[Kousalya et al., 2011] Kousalya, G., Palanikkumar, D., and Piriyankaa, P. (2011). Optimal web service selection and composition using multi-objective bees algorithm. In *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on*, pages 193–196. IEEE.

[Lecue and Mehandjiev, 2009] Lecue, F. and Mehandjiev, N. (2009). Towards scalability of quality driven semantic web service composition. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 469–476. IEEE.

[Lee et al., 2003] Lee, K., Jeon, J., Lee, W., Jeong, S.-H., and Park, S.-W. (2003). Qos for web services : Requirements and possible approaches. *W3C working group note*, 25(3) :119.

[Li et al., 2020] Li, C., Li, J., and Chen, H. (2020). A meta-heuristic-based approach for qos-aware service composition. *IEEE Access*, 8 :69579–69592.

[Li et al., 2014] Li, J., Zheng, X.-L., Chen, S.-T., Song, W.-W., and Chen, D.-r. (2014). An efficient and reliable approach for quality-of-service-aware service composition. *Information Sciences*, 269 :238–254.

[Li et al., 2010] Li, W., Dai, X., and Jiang, H. (2010). web services composition based on weighted planning graph. In *Networking and Distributed Computing (ICNDC), 2010 First International Conference on*, pages 89–93. IEEE.

[Li et al., 2018] Li, Y., Hu, J., Wu, Z., Liu, C., Peng, F., and Zhang, Y. (2018). Research on qos service composition based on coevolutionary genetic algorithm. *Soft Computing*, 22 :7865–7874.

[Liang et al., 2019] Liang, X., Lu, Q., and Li, M. (2019). Research on web service selection based on improved skyline algorithm. In *2019 IEEE Intl Conf*

on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), pages 1323–1328. IEEE.

[Liao et al., 2011] Liao, J., Liu, Y., Zhu, X., Xu, T., and Wang, J. (2011). Niching particle swarm optimization algorithm for service composition. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE.

[Liu et al., 2010] Liu, H., Zhong, F., Ouyang, B., and Wu, J. (2010). An approach for qos-aware web service composition based on improved genetic algorithm. In *Web Information Systems and Mining (WISM), 2010 International Conference on*, volume 1, pages 123–128. IEEE.

[Liu et al., 2012] Liu, M., Wang, M., Shen, W., Luo, N., and Yan, J. (2012). A quality of service (qos)-aware execution plan selection approach for a service composition process. *Future Generation Computer Systems*, 28(7) :1080–1089.

[Liu et al., 2011] Liu, Y., Miao, H., Li, Z., and Gao, H. (2011). Qos-aware web services composition based on hqpso algorithm. In *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, pages 400–405. IEEE.

[Liu et al., 2004] Liu, Y., Ngu, A. H., and Zeng, L. Z. (2004). Qos computation and policing in dynamic web service selection. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73.

[Llinás and Nagi, 2015] Llinás, G. A. G. and Nagi, R. (2015). Network and qos-based selection of complementary services. *IEEE Transactions on Services Computing*, 8(1) :79–91.

[Luo et al., 2010] Luo, J., Li, W., Liu, B., Zheng, X., and Dong, F. (2010). Multi-agent coordination for service composition. *Agent-based service-oriented computing*, pages 47–80.

[Luo et al., 2011] Luo, Y.-s., Qi, Y., Hou, D., Shen, L.-f., Chen, Y., and Zhong, X. (2011). A novel heuristic algorithm for qos-aware end-to-end service composition. *Computer Communications*, 34(9) :1137–1144.

[MacKenzie et al., 2006] MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., and Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, 12 :18.

[Martello and Toth, 1987] Martello, S. and Toth, P. (1987). Algorithms for knapsack problems. *North-Holland Mathematics Studies*, 132 :213–257.

[Martin et al., 2005] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., et al. (2005). Bringing semantics to web services : The owl-s approach. In *Semantic Web Services and Web Process Composition : First International Workshop, SWSWPC 2004, San Diego, CA, USA, July 6, 2004, Revised Selected Papers 1*, pages 26–42. Springer.

[Mirjalili et al., 2014] Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, 69 :46–61.

[Moghaddam and Davis, 2013] Moghaddam, M. and Davis, J. G. (2013). Service selection in web service composition : A comparative review of existing approaches. *Web services foundations*, pages 321–346.

[Mohammed et al., 2014] Mohammed, M., Chikh, M. A., and Fethallah, H. (2014). Qos-aware web service selection based on harmony search. In *ISKO-Maghreb : Concepts and Tools for knowledge Management (ISKO-Maghreb), 2014 4th International Symposium*, pages 1–6. IEEE.

[Mostafa and Zhang, 2015] Mostafa, A. and Zhang, M. (2015). Multi-objective service composition in uncertain environments. *IEEE Transactions on Services Computing*.

[Naseri and Jafari Navimipour, 2019] Naseri, A. and Jafari Navimipour, N. (2019). A new agent-based method for qos-aware cloud service composition using particle swarm optimization algorithm. *Journal of ambient intelligence and humanized computing*, 10 :1851–1864.

[Osman et al., 2005] Osman, T., Thakker, D., and Al-Dabass, D. (2005). Bridging the gap between workflow and semantic-based web services composition. *6789@ ABCDE FGHC6D • I*, page 13.

[O'sullivan et al., 2002] O'sullivan, J., Edmond, D., and Ter Hofstede, A. (2002). What's in a service ? *Distributed and Parallel Databases*, 12(2-3) :117–133.

[Ouadah et al., 2019] Ouadah, A., Hadjali, A., Nader, F., and Benouaret, K. (2019). Sefap : an efficient approach for ranking skyline web services. *Journal of Ambient Intelligence and Humanized Computing*, 10 :709–725.

[Paik et al., 2017] Paik, H.-Y., Lemos, A. L., Barukh, M. C., Benatallah, B., and Natarajan, A. (2017). *Web service implementation and composition techniques*, volume 256. Springer.

[Papazoglou et al., 2008] Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2008). Service-oriented computing : a research roadmap. *International Journal of Cooperative Information Systems*, 17(02) :223–255.

[Peltz, 2003] Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10) :46–52.

[Permadi and Santoso, 2018] Permadi, V. A. and Santoso, B. J. (2018). Efficient skyline-based web service composition with qos-awareness and budget constraint. In *2018 International Conference on Information and Communications Technology (ICOIACT)*, pages 855–860. IEEE.

[Purohit and Kumar, 2018] Purohit, L. and Kumar, S. (2018). A classification based web service selection approach. *IEEE Transactions on Services Computing*, 14(2) :315–328.

[Ranjan and Sahoo, 2020] Ranjan, A. and Sahoo, B. (2020). Web service selection mechanism in service-oriented architecture based on publish–subscribe pattern in fog environment. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 269–281. Springer.

[Rao and Su, 2004] Rao, J. and Su, X. (2004). A survey of automated web service composition methods. In *SWSWPC*, volume 3387, pages 43–54. Springer.

[Remaci et al., 2021] Remaci, Z. Y., Hadjila, F., Echialli, A., and Merzoug, M. (2021). Dynamic web service selection based on score voting. In *Advances in Computing Systems and Applications : Proceedings of the 4th Conference on Computing Systems and Applications*, pages 185–195. Springer.

[Rosenberg et al., 2009] Rosenberg, F., Celikovic, P., Michlmayr, A., Leitner, P., and Dustdar, S. (2009). An end-to-end approach for qos-aware service composition. In *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*, pages 151–160. IEEE.

[Ross-Talbot and Fletcher, 2006] Ross-Talbot, S. and Fletcher, T. (2006). Web services choreography description language : Primer. *World Wide Web Consortium, Working Draft*.

[Sadiq and Racca, 2003] Sadiq, W. and Racca, F. (2003). *Business services orchestration : The hypertier of information technology*. Cambridge University Press.

[Schantz and Schmidt, 2001] Schantz, R. E. and Schmidt, D. C. (2001). Middleware for distributed systems : Evolving the common structure for network-centric applications. *Encyclopedia of Software Engineering*, 1 :1–9.

[Schapire et al., 2013] Schapire, R., Schölkopf, B., Luo, Z., and Vovk, V. (2013). Empirical inference. *Berlin, Heidelberg*, pages 37–52.

[Schapire, 2013] Schapire, R. E. (2013). Explaining adaboost. In *Empirical Inference : Festschrift in Honor of Vladimir N. Vapnik*, pages 37–52. Springer.

[Seghir et al., 2019] Seghir, F., Khababa, A., and Semchedine, F. (2019). An interval-based multi-objective artificial bee colony algorithm for solving the web service composition under uncertain qos. *The Journal of Supercomputing*, 75 :5622–5666.

[Serrai et al., 2016] Serrai, W., Abdelli, A., Mokdad, L., and Hammal, Y. (2016). An efficient approach for web service selection. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 167–172. IEEE.

[Serrai et al., 2019] Serrai, W., Abdelli, A., Mokdad, L., and Serrai, A. (2019). How to deal with qos value constraints in mcdm based web service selection. *Concurrency and Computation : Practice and Experience*, 31(24) :e4512.

[Sheng et al., 2014] Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition : A decade's overview. *Information Sciences*, 280 :218–238.

[Shetty and D'Mello, 2018] Shetty, J. and D'Mello, D. A. (2018). Global and local optimisation-based hybrid approach for cloud service composition. *International Journal of Computational Science and Engineering*, 17(1) :1–14.

[Standard, 2007] Standard, O. (2007). Wsbepl ver. 2.0. *Available on : http ://-docs. oasisopen. org/wsbpel/2.0/os/wsbpel-v2. 0-OS. html*.

[Strunk, 2010] Strunk, A. (2010). Qos-aware service composition : A survey. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 67–74. IEEE.

[Sun et al., 2016] Sun, R., Zhang, B., and Liu, T. (2016). Ranking web service for high quality by applying improved entropy-topsis method. In *2016*

17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pages 249–254. IEEE.

[Sun and Zhao, 2012] Sun, S. X. and Zhao, J. (2012). A decomposition-based approach for service composition with global qos guarantees. *Information Sciences*, 199 :138–153.

[Szyperski et al., 1999] Szyperski, C., Bosch, J., and Weck, W. (1999). Component-oriented programming. In *European Conference on Object-Oriented Programming*, pages 184–192. Springer.

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

[Tan et al., 2001] Tan, K.-L., Eng, P.-K., Ooi, B. C., et al. (2001). Efficient progressive skyline computation. In *VLDB*, volume 1, pages 301–310.

[Tan and Zhou, 2013] Tan, W. and Zhou, M. (2013). *Business and Scientific Workflows : A Web Service-Oriented Approach*. John Wiley & Sons.

[Taylor, 1997] Taylor, D. A. (1997). *Object technology : a manager's guide*. Addison-Wesley Longman Publishing Co., Inc.

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

[Tiwari and Kumar, 2021] Tiwari, R. K. and Kumar, R. (2021). G-topsis : a cloud service selection framework using gaussian topsis for rank reversal problem. *The Journal of Supercomputing*, 77 :523–562.

[Torra, 2010] Torra, V. (2010). Hesitant fuzzy sets. *International journal of intelligent systems*, 25(6) :529–539.

[Tripathy et al., 2015] Tripathy, A. K., Patra, M. R., and Pradhan, S. K. (2015). Dynamic qos requirement aware service composition and adaptation. In *Service-Oriented Computing-ICSOC 2014 Workshops : WESOA ; SeMaPS, RMSOC, KASA, ISC, FOR-MOVES, CCSA and Satellite Events, Paris, France, November 3-6, 2014, Revised Selected Papers*, pages 378–385. Springer.

[Trummer et al., 2014] Trummer, I., Faltings, B., and Binder, W. (2014). Multi-objective quality-driven service selection—a fully polynomial time approximation scheme. *IEEE Transactions on Software Engineering*, 40(2) :167–191.

[Vinoski, 2004] Vinoski, S. (2004). An overview of middleware. In *International Conference on Reliable Software Technologies*, pages 35–51. Springer.

[Wagh and Thool, 2012] Wagh, K. and Thool, R. (2012). A comparative study of soap vs rest web services provisioning techniques for mobile host. *Journal of Information Engineering and Applications*, 2(5) :12–16.

[Wang and Jiang, 2007] Wang, G. and Jiang, H. (2007). Fuzzy-dominance and its application in evolutionary many objective optimization. In *2007 International conference on computational intelligence and security workshops (CISW 2007)*, pages 195–198. IEEE.

[Wang et al., 2012] Wang, L., Shen, J., and Yong, J. (2012). A survey on bio-inspired algorithms for web service composition. In *Proceedings of the 2012 IEEE 16th international conference on computer supported cooperative work in design (CSCWD)*, pages 569–574. IEEE.

[Wang et al., 2010] Wang, R., Ma, L., and Chen, Y. (2010). The application of ant colony algorithm in web service selection. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1–4. IEEE.

[Wang et al., 2017] Wang, Y., He, Q., Ye, D., and Yang, Y. (2017). Service selection based on correlated qos requirements. In *2017 IEEE International Conference on Services Computing (SCC)*, pages 241–248. IEEE.

[Wang et al., 2016] Wang, Y., Song, Y., and Liang, M. (2016). A skyline-based efficient web service selection method supporting frequent requests. In *2016 IEEE 20th international conference on computer supported cooperative work in design (CSCWD)*, pages 328–333. IEEE.

[Wen et al., 2014] Wen, S., Tang, C., Li, Q., Chiu, D. K., Liu, A., and Han, X. (2014). Probabilistic top-k dominating services composition with uncertain qos. *Service Oriented Computing and Applications*, 8 :91–103.

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

[Xia et al., 2011] Xia, Y., Chen, P., Bao, L., Wang, M., and Yang, J. (2011). A qos-aware web service selection algorithm based on clustering. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 428–435. IEEE.

[Xia et al., 2008] Xia, Y.-m., Chen, J.-l., and Meng, X.-w. (2008). On the dynamic ant colony algorithm optimization based on multi-pheromones. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 630–635. IEEE.

[Xiangbing et al., 2012] Xiangbing, Z., Hongjiang, M., and Fang, M. (2012). An optimal approach to the qos-based wsmo web service composition using genetic algorithm. In *International Conference on Service-Oriented Computing*, pages 127–139. Springer.

[Xu et al., 2018] Xu, J., Guo, L., Zhang, R., Hu, H., Wang, F., and Pei, Z. (2018). Qos-aware service composition using fuzzy set theory and genetic algorithm. *Wireless Personal Communications*, 102 :1009–1028.

[Yachir, 2014] Yachir, A. (2014). *Composition dynamique de services sensibles au contexte dans les systèmes intelligents ambiants*. PhD thesis, Université Paris-Est.

[Yang, 2010] Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer.

[Yang et al., 2010] Yang, Z., Shang, C., Liu, Q., and Zhao, C. (2010). A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm. *Journal of Computational Information Systems*, 6(8) :2617–2622.

[Yasmina and Fethallah, 2022] Yasmina, R. Z. and Fethallah, H. (2022). Uncertain service selection using hesitant fuzzy sets and grey wolf optimisation. *International Journal of Web Engineering and Technology*, 17(3) :250–277.

[Yasmina et al., 2018] Yasmina, R. Z., Fethallah, H., and Fedoua, D. (2018). Selecting web service compositions under uncertain qos. In *Computational Intelligence and Its Applications : 6th IFIP TC 5 International Conference, CIIA 2018, Oran, Algeria, May 8-10, 2018, Proceedings 6*, pages 622–634. Springer.

[Youssef, 2020] Youssef, A. E. (2020). An integrated mcdm approach for cloud service selection based on topsis and bwm. *IEEE Access*, 8 :71851–71865.

[Yu and Bouguettaya, 2010] Yu, Q. and Bouguettaya, A. (2010). Computing service skyline from uncertain qows. *IEEE Transactions on Services Computing*, 3(1) :16–29.

[Yu and Bouguettaya, 2013] Yu, Q. and Bouguettaya, A. (2013). Efficient service skyline computation for composite service selection. *IEEE Transactions on Knowledge and Data Engineering*, 25(4) :776–789.

[Yu et al., 2007] Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1) :6.

[Yu et al., 2013] Yu, Y., Ma, H., and Zhang, M. (2013). An adaptive genetic programming approach to qos-aware web services composition. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1740–1747. IEEE.

[Zeleny, 1976] Zeleny, M. (1976). The theory of the displaced ideal. In *Multiple criteria decision making Kyoto 1975*, pages 153–206. Springer.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). Qos-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5) :311–327.

[Zeyneb Yasmina et al., 2022] Zeyneb Yasmina, R., Fethallah, H., and Fadoua, L. (2022). Web service selection and composition based on uncertain quality of service. *Concurrency and Computation : Practice and Experience*, 34(1) :e6531.

[Zhang et al., 2011] Zhang, Y., Zheng, Z., and Lyu, M. R. (2011). Exploring latent features for memory-based qos prediction in cloud computing. In *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on*, pages 1–10. IEEE.

[Zhao et al., 2012] Zhao, X., Song, B., Huang, P., Wen, Z., Weng, J., and Fan, Y. (2012). An improved discrete immune optimization algorithm based on pso for qos-driven web service composition. *Applied Soft Computing*, 12(8) :2208–2216.

[Zhao et al., 2017] Zhao, Y., Tan, W., and Jin, T. (2017). Qos-aware web service composition considering the constraints between services. In *Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, pages 229–232. ACM.

[Zheng et al., 2012] Zheng, H., Zhao, W., Yang, J., and Bouguettaya, A. (2012). Qos analysis for web service compositions with complex structures. *IEEE Transactions on Services Computing*, 6(3) :373–386.

[Zheng et al., 2010] Zheng, Z., Zhang, Y., and Lyu, M. R. (2010). Distributed qos evaluation for real-world web services. In *2010 IEEE International Conference on Web Services*, pages 83–90. IEEE.

[Zhou et al., 2004] Zhou, C., Chia, L.-T., and Lee, B.-S. (2004). Qos-aware and federated enhancement for uddi. *International Journal of Web Services Research (IJWSR)*, 1(2) :58–85.

[Zribi, 2014] Zribi, S. (2014). *La gouvernance SOA pour une approche de conception de Système d'Information de Médiation : réconciliation non-fonctionnelle de services pour mettre en œuvre les processus métier*. PhD thesis, Ecole des Mines d'Albi-Carmaux.

Résumé

La technologie des services web constitue une implémentation idéale du paradigme du calcul orienté services (SOC). Étant donné que l'objectif principal du SOC est d'assurer l'interopérabilité des applications et la création de compositions d'applications (ou de services) avec valeurs ajoutées, il conviendra de concevoir et de mettre en œuvre des modèles permettant de combiner des services web individuels dans des workflows satisfaisants des critères de performance objectifs. Il convient de noter que les services web courants sont caractérisés par différents attributs de QoS qui jouent un rôle majeur dans la spécification des compositions de services désirés. Il est utile de souligner que les attributs de QoS dépendent largement des fluctuations de l'environnement (par exemple, la surcharge des réseaux, ou la fluctuation des coûts en raison des saisons ou des événements socioculturels) et par conséquent, leur incertitude créera des difficultés supplémentaires dans la modélisation mathématique du problème de composition. Dans cette thèse, nous adressons la composition des services avec incertitude de QoS en proposant deux contributions principales, toutes les deux exploitent une recherche locale et globale pour alléger la complexité temporelle du problème. La première contribution exploite l'heuristique des intervalles majoritaires pour effectuer la recherche locale, en outre, la recherche globale est effectuée à l'aide d'une recherche exhaustive qui exploite les contraintes globales. Dans la deuxième contribution, nous adoptons une version discrète de la méta-heuristique de l'algorithme des chauves-souris (bat algorithm) en plus d'un ensemble d'heuristiques (telles que la dominance floue et la dominance stochastique d'ordre zéro) pour effectuer à la fois la recherche locale et globale. Les résultats obtenus confirment l'efficacité de nos contributions, et en particulier, les performances étaient satisfaisantes pour les workflows qui ont une taille variant entre 2 et 10 composants.

Mots clés : calcul orienté services, services web, composition de services web, qualité de service incertaine, algorithmes des chauves souris, méta-heuristique, dominance floue, dominance stochastique, intervalle majoritaire, recherche exhaustive, services IoT.

Abstract

The web service technology constitutes the golden standard of the service-oriented computing paradigm (SOC). Since the ultimate objective of SOC is how to ensure interoperability of applications and how to create pertinent compositions of services, it will be appropriate to design and implement models for combining individual web services and creating sophisticated workflows. It is worth noting that the actual web services are characterized by different QoS attributes that constitute a major role in specifying the desired service compositions. It should be underlined that the QoS attributes are largely dependent on the environment fluctuations (e.g., network overload), and therefore their inherent uncertainty will create additional difficulties in the mathematical modelling of the problem. In this thesis, we address the service composition with QoS uncertainty by proposing two main contributions; both of them leverage a local and global search to alleviate the complexity of the problem. The first contribution leverages the majority interval heuristic to perform the local search, while the global search is performed using exhaustive search. In the second contribution, we adopt a discrete version of the bat algorithm meta-heuristic, in addition to a set of heuristics (such as fuzzy dominance and zero-order stochastic dominance) for performing both the local and global search. The obtained results confirm the effectiveness of the contributions, especially for workflows with a size ranging from 2 up to 10 components.

Keywords: service oriented computing, web services, web service composition, uncertain quality of service, bat algorithm, meta-heuristics, fuzzy dominance, stochastic dominance, majority interval, exhaustive search, IoT services.

الخلاصة

تعد تقنية خدمات الويب تطبيقًا مثاليًا لنموذج الحوسبة الموجهة نحو الخدمة (SOC). نظرًا لأن الهدف الرئيسي لـ SOC هو ضمان قابلية التشغيل البيني للتطبيقات وإنشاء مجموعات من التطبيقات (أو الخدمات) ذات القيم المضافة، سيكون من الضروري تصميم وتنفيذ نماذج تسمح بدمج خدمات الويب الفردية في سير العمل بما يحقق الهدف معايير الأداء. تجدر الإشارة إلى أن خدمات الويب العشرة تتميز بسمات جودة الخدمة المختلفة التي تلعب دورًا رئيسيًا في تحديد تركيبات الخدمة المطلوبة. سيكون من المفيد التأكيد على أن سمات جودة الخدمة تعتمد إلى حد كبير على التقلبات البيئية (مثل الحمل الزائد على الشبكة، أو تقلب التكلفة بسبب المواسم أو الأحداث الاجتماعية والثقافية)، وبالتالي فإن عدم اليقين المتأصل فيها سيخلق صعوبات إضافية في النمذجة الرياضية لمشكلة التكوين. في هذه الأطروحة، نتناول تكوين الخدمة مع عدم اليقين في جودة الخدمة من خلال اقتراح مساهمتين رئيسيتين، كلاهما يستغلان البحث المحلي والعالمي للتخفيف من التعقيد الزمني للمشكلة. تستغل المساهمة الأولى إرشادية الفاصل للأغلبية لإجراء البحث المحلي، علاوة على ذلك، يتم إجراء البحث الشامل باستخدام بحث شامل يستغل القيود الكلية. في المساهمة الثانية، نعتمد نسخة منفصلة من الاستدلال الفوقي لخوارزمية الخفافيش بالإضافة إلى مجموعة من الاستدلالات (مثل الهيمنة الغامضة والهيمنة العشوائية ذات الترتيب الصفري.) لإجراء البحث المحلي والكلي. تؤكد النتائج التي تم الحصول عليها فعالية مساهماتنا، وعلى وجه الخصوص، كان الأداء مرضيا بالنسبة لسير العمل الذي يتراوح حجمه بين 2 و 10 مكونات.

الكلمات الرئيسية : الحوسبة الموجهة نحو الخدمة، خدمات الويب، تركيب خدمات الويب، جودة الخدمة غير المؤكدة، خوارزميات الخفافيش، الفوقية الإرشادية، الهيمنة الغامضة، الهيمنة العشوائية، فاصل الأغلبية، البحث الشامل، خدمات إنترنت الأشياء.