République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLAB Blida

Faculté des Sciences

Département d'Informatique

Mémoire de fin d'études en vue de l'obtention du Diplôme

Master en Informatique

Option: Système Informatique & Réseau

Thème :

Proposition d'un nouvel algorithme de sélection des services Cloud manufacturing sous différentes contraintes QOS

Présentée par :

GUETTAL Roumaissa **&** HAMMOUM Sarah

Présidente : Mme OUKID Lamia

Promotrice : Mme BEY- Fella

Examinatrice : Mme MANCER Yasmin

Année universitaire : 2022/2023.

Remerciements

*Avant tout, nous souhaitons rendre grâce à notre **Dieu**, qui nous a accordé la force, la persévérance et la clarté d'esprit nécessaires pour mener à bien ce travail. Sa guidance divine a été notre lumière dans les moments de doute et d'incertitude.*

*Nous tenons également à remercier notre promotrice, **Mme BEY-Fella**, pour sa précieuse expertise, son soutien constant et sa patience tout au long de cette entreprise. Ses conseils éclairés et ses encouragements ont été essentiels à notre progression.*

Nous exprimons notre gratitude envers l'ensemble du corps professoral du département d'Informatique de l'université SAAD DAHLAB BLIDA 1, qui nous a offert un enseignement de qualité et des ressources précieuses pour enrichir nos connaissances.

Nous tenons à exprimer notre reconnaissance envers nos camarades de promotion SIR 2023, avec qui nous avons partagé des moments de collaboration, d'échange et d'entraide. Leurs idées, leur esprit d'équipe et leur engagement ont contribué à notre réussite collective.

Enfin, nous voudrions adresser nos remerciements à toutes les personnes qui, de près ou de loin, ont contribué à notre projet en partageant leurs connaissances, leurs commentaires constructifs et leurs encouragements. Votre contribution a été inestimable et a contribué à l'amélioration de notre travail.

Nous sommes profondément reconnaissants envers chacune de ces personnes pour leur apport précieux dans la réalisation de notre Projet de Fin d'Études.

Merci a tous ...

Dédicace

A la personne la plus chère à mon cœur, ma mère qui m'a encouragé à poursuivre mes rêves et soutenu dans chaque étape de ma vie, mon roc, ma source d'inspiration et ma plus grande admiratrice. Que Dieu le tout puissant veille sur elle et la protège .

A mon superbe père qui m'indique toujours la bonne voie, qui a fait de moi ce que je suis aujourd'hui, et qui fait tout pour nous rendre heureux , celui qui m'a fait une femme, ma source de vie, d'amour et d'affectation, mon épaule solide.

A l'homme de ma vie, à mon frère Yasser. Tu es mon compagnon de vie, mon confident. Tu as toujours été là pour moi, prêt à me soutenir dans les moments difficiles et à partager mes joies dans les moments de bonheur.

A ma princesse Serine qui sait toujours comment procurer la joie et le bonheur pour toute la famille.

A ma cousine, et ma sœur Racha qui n'a pas cessé de me conseiller, encourager et soutenir tout au long de mes études. Aucun mot ne pourra exprimer mon attachement envers pour vous.

A mes meilleures amis Imene et Ikram, vous avez toujours été là pour moi, me soutenant et m'encourageant dans tous les aspects de ma vie. Votre amitié précieuse et votre soutien inconditionnel ont été des sources de bonheur et de force pour moi.

A tous les membres de ma grande famille, mes grand parents, mes oncles, tantes, et mes cousines.

*A mon **chère binôme Sarah** pour son soutien moral, sa patience et compréhension tout au longue de ce projet*

Sans oublier mon chat mimi, pour sa compagne à mes nuits blanches, son amour et affection.

Roumaissa

Dédicace

À l'homme de ma vie, a celui qui s'est toujours sacrifié pour me voir réussir, ma source de vie, d'amoure et d'affectation a mon support qui été toujours a mes cotés pour me soutenir et m'encourager

A Mon cher papa.

A mon paradis, a la prunelle de mes yeux ,a la source de ma joie et mon bonheur ,a ma première amie , la lune et le fil d'espoir qui allumer mon chemin ,ma moitié.

Maman ma chérie qui j'adore.

***A mes chère frères**, Oussama et Ismail votre présence a été un réconfort indéfectible tout au long de ce parcours, vos encouragements, vos conseils et votre soutien ont été précieux. Nous avons partagé des moments de rire, de soutien mutuel et de complicité.*

***A mon ange mon chère frère,** Riyad qui nous a quittés trop tôt ta présence me manque chaque jour ta mémoire reste gravée dans mon cœur et dans mes pensées toutes au longue de ma vie.*

***A ma meilleur amie**, Imane, ma moitié, ma sœur, ma source de soutien et ma complice tout au long de cette aventure. Je suis reconnaissante d'avoir une amie aussi merveilleuse et je te remercie d'avoir partagé ce chemin avec moi.*

*Sans oublier mon **chère binôme Romaissa** pour son soutien moral, sa patience et compréhension tout au longue de ce projet*

***A toutes mes amies et mes collègues** et tous c qui ont participé à ma réussite et a tous qui m'aiment ainsi qu'à tous mes enseignants que j'ai l'honneur de rencontrer tout au long de mes études au département des Mathématiques et Informatique.*

Sarah

Résumé

L'un des défis majeurs auxquels sont confrontés les systèmes de fabrication en nuage (Cloud Manufacturing) est la présence croissante de services offrant des fonctionnalités similaires mais présentant des performances différentes. Afin de garantir que la fabrication en nuage réponde aux exigences des tâches complexes, il est essentiel de résoudre le problème dela sélection optimale du service CMfg (OSCM). Notre étude se concentre sur la phase de sélection des services CMfg et propose un nouvel algorithme de sélection amélioré avec des contraintes QoS.

Dans un premier temps, nous proposons une approche novatrice basée sur la décomposition des contraintes globales de qualité de service (QoS) en contraintes locales de QoS. Ensuite, nous présentons un algorithme de sélection basé sur le classement des services Cloud, qui évalue et classe tous les services candidats, identifiant ainsi la combinaison la plus proche de l'optimalité sans nécessiter une recherche exhaustive de toutes les possibilités.

Enfin, plusieurs expériences ont été réalisées afin d'évaluer l'efficacité de notre approche à l'aide d'un ensemble de données. Les résultats expérimentaux démontrent que notre solution proposée est capable de résoudre des problèmes à grande échelle dans les systèmes CMfg.

Mots clés : Fabrication en nuage, sélection optimale du service CMfg, contraintes QoS, contraintes locales, contraintes globales.

Abstract

One of the major challenges faced by Cloud Manufacturing systems is the increasing presence of services that offer similar functionalities but differ in terms of performance. To ensure that cloud manufacturing meets the requirements of complex tasks, solving the problem of Optimal Service CMfg Selection (OSCM) is crucial. Our study focuses on the service selection phase of CMfg and proposes a novel selection algorithm enhanced with Quality of Service (QoS) constraints.

Firstly, we propose an innovative approach based on decomposing the global QoS constraints into local QoS constraints. Next, we present a ranking-based service selection algorithm for Cloud services, whiche valuates and ranks all candidate services, identifying the combination that comes closest to optimality while avoiding an exhaustive search through all execution plans.

Finally, we describe several performance experiments conducted to evaluate the effectiveness of our approach using a dataset. The experimental results demonstrate that our proposed solution is superior and effective in solving large-scale problems in CMfg systems.

Keywords: Cloud Manufacturing, Optimal Service CMfg Selection, QoS constraints, local constraints, global constraints.

ملخص

أحد التحديات الرئيسية التي تواجهها أنظمة التصنيع السحابي هو الوجود المتزايد للخدمات التي تقدم وظائف مماثلة ولكنها تختلف من حيث الأداء. للتأكد من أن التصنيع السحابي يلبي متطلبات المهام المعقدة، يعد حل مشكلة اختيار CMfgللخدمة المثلى (OSCM) أمرا بالغ الأهمية. تركز دراستنا على مرحلة اختيار الخدمة في CMfgوتقترح خوارزمية اختيار جديدة معززة بقيود جودة الخدمة (QOS)

أولا، نقترح خوارزمية مبتكرة تعتمد على تحليل قيود جودة الخدمة العالمية إلى قيود جودة الخدمة المحلية. بعد ذلك، نقدم خوارزمية اختيار الخدمة القائمة على الترتيب للخدمات السحابية، والتي تقوم بتقييم وترتيب جميع الخدمات المرشحة، وتحديد المجموعة الأقرب إلى الأمثل مع تجنب البحث الشامل من خلال جميع خطط التنفيذ.

أخيرا، نصف العديد من تجارب الأداء التي أجريت لتقييم فعالية المنهجية المقترحة باستخدام مجموعة بيانات، توضح النتائج التجريبية أن الحل المقترح متفوق وفعال في حل المشكلات واسعة النطاق في أنظمة.CMfg

الكلمات المفتاحية: التصنيع السحابي ، اختيارCMfgللخدمة المثلى ، قيود جودة الخدمة ، القيود المحلية ، القيود العالمية.

Table des matières

Liste des abréviations

ABC : Artificial bee colony.

ACO : Ant colony optimization.

AGACQP : Approche globale augmenter de contraint QoS avec des priorités.

ASSCCT : Approche de sélection de services CMfg consciente du temps.

CC : Cloud Computing.

CMfg : Cloud Manufacturing (La fabrication en nuage).

CMSC : Cloud Manufacturing Service Composition.

CS : Candidat service.

GA : Genetic Algorithm.

GWO : Grey Wolf Optimizer.

HABC : Hybrid Artificial bee colony.

MM : Mathematical Model.

NIST : National Institute of Standards and Technology.

NSGA-II-SA : Non-dominated Sorting Genetic Algorithm II Simulated Annealing.

OSCMCS : Optimal selection of the CMfg service.

PSO : Particle swarm optimization.

QOS : Quality of service.

SCOS : Service composition and optimal selection.

ST : Sub tasks.

Liste des Tableaux

Liste des figures

Introduction générale :

À l'ère actuelle, les approches de fabrication ont subi une transformation majeure, passant d'une fabrication axée sur la production à une fabrication centrée sur le client. Un paradigme émergent, connu sous le nom de fabrication en nuage (CMfg), incarne cette transformation en proposant un modèle de fabrication en réseau axé sur les services. Ce modèle permet aux utilisateurs de services de sélectionner, d'utiliser et de configurer des ressources à la demande afin de réaliser des tâches de fabrication personnalisées.

L'objectif du CMfg est de concevoir un système qui permet aux entreprises de répondre rapidement à la demande du marché, de favoriser le développement scientifique et d'accélérer considérablement la recherche et la découverte de services. Dans cette optique, les défis majeurs consistent à classer et à virtualiser les ressources, ainsi qu'à sélectionner et à composer de manière optimale les services.

La sélection optimale des services représente le défi fondamental du CMfg, puisqu'elle consiste à choisir les services appropriés parmi les ressources cloud de fabrication disponibles, afin de mener à bien la tâche de fabrication et de satisfaire les utilisateurs. Dans le cadre de notre mémoire, nous apportons une contribution essentielle à la phase de sélection des services CMfg en proposant un nouvel algorithme enrichi de contraintes QoS (Qualité de Service).

Problématique :

La satisfaction de certaines demandes complexes des clients requiert une exécution de multiples sous-tâches de services, ce qui appelle un modèle d'exécution adapté pour orchestrer ces différentes tâches. Afin de répondre à ce besoin, un algorithme de sélection des meilleurs services candidats est mis en place pour accomplir toutes les sous-tâches de services commandées, tout en respectant les contraintes spécifiées par le client.

Dans le cadre du problème de sélection des services sous différentes contraintes QoS, l'approche proposée repose sur la décomposition des contraintes globales exigées par le client en contraintes locales. Cela permet de prendre des décisions concernant le choix des services candidats pour chaque sous-tâche, afin de réaliser l'ensemble des tâches requises. Cette approche se distingue par sa rapidité d'exécution.

La problématique principale de ce sujet réside dans la recherche d'une solution pour résoudre le problème d'une complexité exponentielle liée à l'approche globale. Il est donc nécessaire de trouver une alternative qui permette de réduire cette complexité tout en garantissant des résultats proches de l'optimale pour l'exécution des différentes sous-tâches de services.

Objectifs

1. Étude comparative de solutions existantes.
2. Proposition d'un nouvel algorithme de sélection des services cloud manufacturing sousdifférentes contraintes QOS.
3. Sélectionner le meilleur service parmi les services candidats.
4. Minimiser la complexité et le temps d'exécution.
5. Validation de la solution proposée à travers une application java.
6. Réalisation de plusieurs expériences pour comparer nos résultats avec d'autre approche de sélection.

Plan de travail :

Le premier chapitre de cette étude, intitulé "Introduction aux Technologies du Cloud", fournit une vue d'ensemble des concepts fondamentaux du Cloud Computing et du Cloud Manufacturing. Il vise également à développer des connaissances sur la sélection et la composition des services dans le cloud, en se concentrant sur les paramètres de qualité de service (QoS).

Le deuxième chapitre propose une revue de la littérature portant sur les travaux réalisés pour résoudre le problème de la composition des services. Il examine les différentes approches et méthodologies qui ont été étudiées dans ce domaine.

Dans le troisième chapitre, nous détaillons notre démarche conceptuelle pour développer une solution à ce problème. Nous commençons par décrire en détail le problème de sélection des services CMfg. Ensuite, nous présentons chaque étape de notre solution et décrivons l'implémentation de notre algorithme. Nous débutons ce chapitre en introduisant les outils et l'environnement de développement utilisés. Par la suite, nous présentons des interfaces qui illustrent les résultats obtenus grâce à la mise en œuvre de notre solution. Enfin, nous concluons le chapitre en discutant des résultats obtenus et de leur signification.

Enfin, ce mémoire se termine par une conclusion générale qui récapitule le travail réalisé et propose des perspectives futures pour des améliorations ou des extensions potentielles de cette recherche.

Chapitre 1 : Généralité sur la

Technologie de Cloud

1 Introduction :

Aujourd'hui, La technologie de cloud computing est devenue omniprésente dans notre vie quotidienne et a transformé la façon dont nous stockons, partageons et accédons à nos données. Elle offre des avantages considérables aux entreprises de toutes tailles, leur permettant de stocker et de gérer leurs données en toute sécurité et de manière économique. Les entreprises manufacturières peuvent également bénéficier de la technologie de cloud computing en utilisant le cloud manufacturing, qui leur permet de stocker et de gérer à distance leurs ressources de fabrication. Dans ce chapitre, nous allons examiner de plus près la technologie de cloud computing et son application dans le domaine du cloud manufacturing, nous concentrant sur la sélection des services de cloud manufacturing pour répondre aux besoins des entreprises manufacturières.

2 Cloud computing :

De nombreuses définitions et explications du cloud computing ont été proposées, La définition proposée National Institute of Standards and Technology (NIST) [1] est devenue la définition la plus largement acceptée du cloud computing parmi les experts.

D'après cette définition [1], le cloud computing est un modèle qui offre un accès facile, rapide et universel à un pool partagé de ressources informatiques configurables. Ces ressources peuvent être provisionnées en quelques instants à partir de n'importe quel endroit et via Internet ou un réseau permettant ainsi aux utilisateurs d'accéder à des réseaux, des serveurs, du stockage ou des applications. Les services cloud peuvent être déployés rapidement et personnalisés selon les besoins des clients, avec peu d'interaction nécessaire avec le fournisseur de cloud et un effort de gestion minimal, pour répondre aux attentes des utilisateurs et fournir des services de qualité, le cloud computing doit présenter certaines caractéristiques.

Selon le NIST [1] les cinq caractéristiques fondamentales peuvent être regroupées comme suite :

Libre-service à la demande : Le service à la demande permet aux clients d'accéder rapidement et facilement aux ressources dont ils ont besoin, tandis que le libre-service assure une une fourniture automatique de ces ressources, sans intervention humaine nécessaire.

- **Accès réseau étendu :** afin d'utiliser les services de cloud computing, il est nécessaire d'avoir accès à Internet ou les utilisateurs peuvent accéder à tous les services via le réseau et en utilisant des protocoles standard, en utilisant des appareils tels que des ordinateurs, des ordinateurs portables ou des téléphones mobiles qui sont connectés à Internet.
- **Regroupement de ressources :** Les ressources qui peuvent être assignées aux utilisateurs peuvent être de traitement, de logiciel, de stockage, de machines virtuelles et de bande passante réseau. ces ressources sont regroupées pour servir les utilisateurs à un seul emplacement physique et/ou à différents emplacements physiques en fonction des conditions d'optimalité par exemple, sécurité, performance, demande des consommateurs. Le cloud donne l'impression d'une indépendance des emplacements de ressources au niveau inférieur (serveur, cœur), mais pas au niveau supérieur (centre de données, ville, pays).
- **Elasticité rapide :** La caractéristique clé de cloud computing est son élasticité. Les ressources sont disponibles pour les utilisateurs en quantité quasi-illimitée et peuvent être facilement ajustées en fonction des besoins de l'utilisateur, sans intervention du fournisseur de services. Cette élasticité est assurée de manière sécurisée pour garantir la prestation de services de haute qualité.
- **Service mesuré** : Les systèmes de cloud computing sont équipés d'un système de mesure qui permet de facturer les utilisateurs en fonction de l'utilisation des ressources. Les utilisateurs peuvent ainsi accéder à différents niveaux de qualité de services moyennant des coûts différents afin d'optimiser les ressources à différents niveaux d'abstraction, en fonction des types de services.

3 Cloud manufactering :

Le Cloud Manufacturing (CMfg) a suscité un intérêt croissant dans la recherche ces dernières années, conduisant à l'émergence de différentes définitions, il existe maintenant une variété de définitions, C'est pourquoi une définition complète de la CMfg est proposée, combinant les concepts clés de toutes ces définitions et mettant en évidence le potentiel de la CMfg pour une fabrication durable.

Le Cloud Manufacturing, ou fabrication en nuage est un modèle de fabrication orienté vers les services qui permet de virtualiser les ressources et les capacités de production en services accessibles à la demande via le cloud de fabrication. Cette approche transforme les chaînes d'approvisionnement en temporaires, offrant ainsi plus de flexibilité et une meilleure évolutivité, ce qui se traduit par une plus grande résilience et une durabilité accrue tout au long du processus de fabrication. [2]

Le modèle de fabrication en nuage est une plateforme intelligente, basée sur les connaissances et multi-locataire, qui peut fournir des solutions durables tout au long du cycle de vie du produit et du processus grâce à une collaboration efficace, l'exploitation de données et une communication fluide entre les développeurs, les fabricants et les consommateurs. [2]

Ce nouveau paradigme de fabrication présente plusieurs caractéristiques uniques qui le distinguent des autres modèles de fabrication traditionnels. Dans ce contexte, nous allons présenter les principales caractéristiques du Cloud Manufacturing :

- **Flexibilité et évolutivité** : le CMfg offre une évolutivité flexible aux utilisateurs de ressources de fabrication, grâce à des lignes de production temporaires qui permettent la production en petites ou grandes séries. Les chaînes d'approvisionnement du CMfg sont hautement reconfigurables, dynamiques et agiles, ce qui permet au système de s'adapter aux changements imprévus de circonstances. Pour être agile lors de la reconversion des ressources de fabrication, le CM nécessite un haut niveau d'automatisation à travers les chaînes d'approvisionnement. [3]
- **Multiutilisateurs** : Le CMfg peut connecter de manière intelligente plusieurs utilisateurs pour un bénéfice mutuel sans intervention humaine, grâce à la multi-tenance. Chaque utilisateur a un certain degré de contrôle sur l'architecture de son cloud de fabrication, sans changer le code d'application. [4]
- **Outil de prise de décision intelligente et à forte intensité de connaissances :**CMfg permet de partager des données générées par les processus de fabrication afin d'extraire des connaissances utiles pour soutenir des algorithmes de prise de décision intelligents. Les vastes ensembles de données générées tout au long du processus de fabrication nécessitent une collecte et une analyse efficaces pour améliorer le processus et le partage des connaissances de fabrication. Le CMfg nécessite également une prise en charge des connaissances pour fonctionner efficacement. [5]

- **Fabrication intelligente à la demande** : La fabrication en réseau et le Cloud Manufacturing sont deux paradigmes de fabrication qui intègrent des ressources de fabrication distribuées dans un processus de fabrication commun. La principale différence entre les deux est la méthode de communication entre les entreprises. Le Cloud Manufacturing utilise une plateforme cloud pour coordonner un réseau de ressources de fabrication qui peuvent fonctionner automatiquement et/ou indépendamment grâce aux décisions prises par la plateforme cloud. Les ressources de fabrication peuvent être partagées entre plusieurs entreprises et coordonnées de manière intelligente pour répondre aux besoins des utilisateurs. La plateforme cloud doit agir de manière intelligente pour éviter une surcharge de travail sur une seule ressource de fabrication[3].
- **La fabrication en tant que service :** Consiste à intégrer des services tout au long du cycle de vie du produit. Le CMfg réalise cela en empaquetant les ressources et les capacités de fabrication en tant que services coordonnés via le cloud sur un schéma de paiement à l'utilisation. L'objectif est de lier les consommateurs de cloud aux fournisseurs de cloud qui peuvent répondre aux exigences des consommateurs tout en respectant les contraintes de coûts, de délais et de qualité. Le manufacturing as a service est réalisé en créant des services de fabrication cloud (MCS) qui sont vendus sur le cloud sur un schéma de paiement à l'utilisation, offrant une plus grande flexibilité aux fabricants par rapport aux contrats traditionnels fixes. [6,7]

4 La différence entre cloud computing et cloud manufactering :

Le cloud computing et le cloud manufacturing sont deux concepts liés mais distincts qui ont des objectifs différents. Le cloud computing est une technologie qui concentre sur la fourniture de services informatiques, tels que le stockage de données, le traitement et la connectivité réseau. Il est accessible à partir de serveurs distants connectés via Internet. D'autre part, le cloud manufacturing est une technologie qui utilise le cloud computing pour optimiser les processus de fabrication, en fournissant des services de fabrication tels que la conception, la simulation, la planification, la production. Bien que ces deux technologies aient des objectifs différents, elles sont complémentaires et ont le potentiel de révolutionner la manière dont les entreprises gèrent leurs opérations. Le cloud computing permet aux entreprises de fournir des

services informatiques à la demande, tandis que le cloud manufacturing permet aux entreprises de fournir des services de fabrication à la demande.
Les deux technologies permettent également aux entreprises de réduire leurs coûts de gestion des ressources informatiques et de fabrication, en utilisant des ressources partagées pour des applications multiples.

5 La sélection des services dans le contexte de cloud manufactering :

La sélection d'un service CMfg fait référence au processus de sélection d'un fournisseur de services de fabrication en nuage pour externaliser les opérations de fabrication, Le processus de sélection implique l'évaluation de différents fournisseurs de services de fabrication en nuage sur la base de la qualité de service.
Les objectifs de la sélection des services de fabrication en nuage peuvent être divisés en deux catégories : la sélection d'une composition optimale de services et : le choix du meilleur service.

- **Le choix du meilleur service :** Comme le consommateur peut spécifier la qualité de service de chaque élément, le résultat de ce type de sélection de services en nuage est généralement le meilleur service possible. [8]
- **La sélection d'une composition optimale de services :** Pour satisfaire ses exigences en matière de fonctionnalités complexes, le consommateur doit généralement choisir un ensemble de services composés. [8]

5.1 La composition des services CMfg

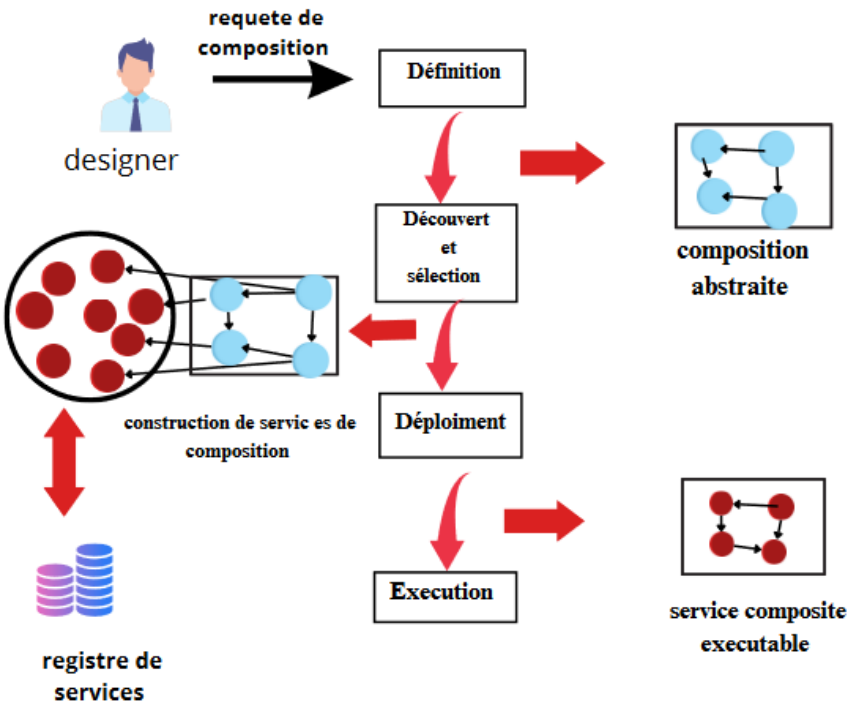


Figure 1:Processus de la sélection de services CMfg .

La composition des services de fabrication en nuage se divise en trois étapes, qui sont définies comme suit :

- **Décomposition des tâches :** Chaque tâche T comprend une série de sous-tâches ST, $T=\{ST_1, ST_2, ..., ST_n\}$.
- **Sélection de services :** Toute sous-tâche ST a un ensemble de services candidats correspondant, qui a plusieurs services candidats.
- **Composition de services :** la sélection d'un service candidats pour chaque sous tache en respectent un ensemble des contraintes QOS.

La composition du service CMfg a quatre structure de base, qui sont des structures parallèles, séquentielles, en boucle et sélectives.

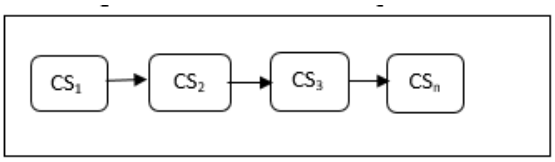
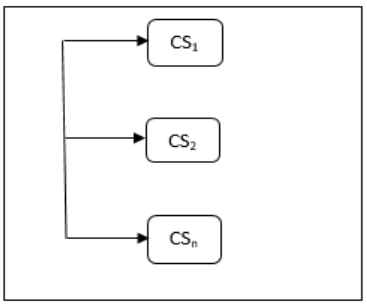
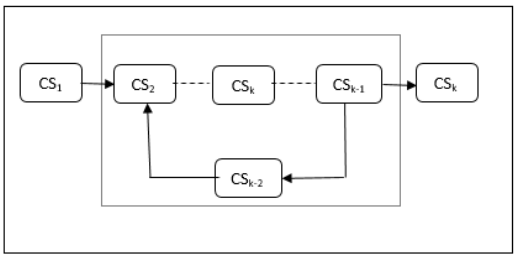
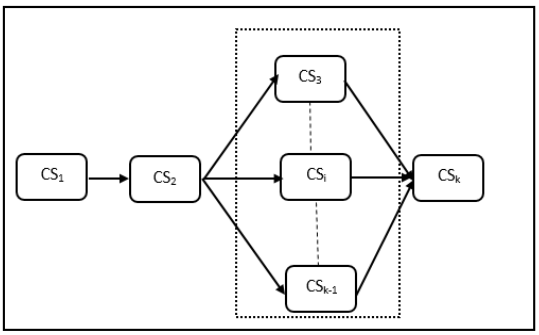
Les structures	Démonstration
Séquentielles : exécution séquentielles des services**.**	
Parallèles : exécution des services au même temps	
Boucle : exécution répétitif n fois d'un ou plusieurs services.	
Sélectives : sectionnement d'un service parmi plusieurs services.	

Tableau 1: La structure de la composition du service CMfg

5.2 Les paramètres QOS incluent dans les services de cloud manufactering :

La qualité des services Web (QoS) est un aspect non fonctionnel (par exemple coût, fiabilité, temps de réponse etc.) utilisé pour exprimer le degré de satisfaction des

consommateurs à l'égard des services Web. Les valeurs de ses paramètres sont mesurées Pour faciliter la description.

L'ensemble des paramètres de qualité de service est divisé en deux sous-ensembles : les attributs positifs et les attributs négatifs. Les valeurs des attributs positifs doivent être maximisées, tandis que les valeurs des attributs négatifs doivent être minimisées. [11,12]

Il existe Quatorze QOS paramètres parmi eux on trouve :

- **Temps de réponse :** Le temps nécessaire pour envoyer une demande et recevoir une réponse est mesurée en millisecondes (ms).
- **Coût :** Représente le coût nécessaire pour une demande.
- **Disponibilité :** c'est le rapport entre le temps total pendant lequel un service peut être utilisé au cours d'une période donnée ce rapport est exprimé en pourcentage (%).
- **Débit :** Nombre total d'invocations pour une période donnée, qu'est mesurée en invocation / seconde.
- **Succès :** Nombre de messages de réponse / nombre de messages de demande exprimé en pourcentage (%).
- **Fiabilité** : rapport entre le nombre des messages d'erreur par rapport les messages totale (%).
- **Documentation :** mesure de la documentation (la clarté, la complétude, le mise à jour, accessibilité, multilinguisme, support) (%).
- **Latence :** Temps nécessaire au serveur pour traiter une requête donnée mesuré en milliseconde (ms).

6 Conclusion :

Dans ce chapitre, nous avons fourni une vue d'ensemble des services CMfg en mettant l'accent sur l'importance de la sélection et de la composition de ces services. Nous avons examiné de près les différents aspects liés à la composition des services, y compris les problèmes de recherche, les méthodes de sélection des services composites, ainsi que les structures et les cycles de vie associés.

En outre, une attention particulière a été accordée aux paramètres QoS, qui sont essentiels pour évaluer et comparer les services CMfg. Ces paramètres QoS constituent des critères clés permettant de mesurer la performance, la fiabilité et d'autres aspects importants des services CMfg.

Dans le prochain chapitre, nous présentons une revue approfondie des travaux existants portant sur le problème de sélection des services CMfg. Cette revue de l'état de l'art nous permettra de mieux appréhender les approches et les algorithmes existants, les méthodologies et techniques employées, ainsi que les avantages et les limitations de chaque approche.

Chapitre 2 Etat de l’art des approches de sélection des services Cloud Manufacturing (étude comparative)

1 Introduction :

Le Cloud Manufacturing est une solution innovante qui permet aux entreprises de délocaliser leurs processus de production et de stockage de données en utilisant des services Cloud. Cependant, la sélection des services Cloud appropriés peut être un défi pour les entreprises, en particulier pour les petites et moyennes entreprises qui peuvent ne pas avoir les ressources nécessaires pour évaluer les différents fournisseurs de services Cloud. Afin de répondre à ce défi, de nombreuses approches de sélection des services Cloud Manufacturing ont été développées. Dans ce chapitre, nous présentons un état de l'art de ces approches, en nous concentrant sur les différentes techniques et méthodes utilisées pour sélectionner les services Cloud appropriés pour les besoins de l'entreprise. L'objectif de ce chapitre est de présenter une étude comparative des différentes approches actuelles de sélection des services Cloud Manufacturing, en présentant les avantages et les inconvénients de chaque approche utilisant des critères tels que la qualité de service (QoS).

2 Etude comparative de solutions existantes :

Dans cette section, nous allons présenter l'ensemble des approches existantes du problème de la sélection des services CMfg sous différents contraintes QOS :

2.1 L'algorithme de MCDM Hybride (Multi-Criteria decision-making):

Le MCDM Hybride (Multiple Criteria Decision Making Hybrid) est une approche décisionnelle qui combine différentes méthodes, techniques ou modèles pour résoudre des problèmes complexes impliquant plusieurs critères et alternatives. L'idée principale du MCDM hybride est d'exploiter les avantages de différentes méthode sa fin d'améliorer la précision, la robustesse et la pertinence des résultats de décision. En intégrant plusieurs approches, Hybrid MCDM vise à fournir une solution plus complète et équilibrée, prenant en compte tous les aspects et perspectives du terminal. [13]

Deux articles ont été étudiés pour approfondir notre compréhension de cet algorithme

a) Une méthode de sélection de services cloud basée sur la qualité de service (QoS) a été proposée par **M. Liu et al.** [14]. En utilisant la méthode de poids d'entropie et l'algorithme GRA-ELECTRE III, afin d'évaluer l'importance des critères de QoS pour

classer les services en fonction de leur adéquation aux exigences de QoS de l'utilisateur. Les tests réalisés sur des données de services cloud publics ont révélé que cette méthode était plus efficace que les méthodes existantes en termes de précision de choix de service et de temps de réponse. L'approche se compose de trois étapes principales : le traitement des données hétérogènes des attributs de QoS, le calcul objectif du poids des attributs de QoS en utilisant une méthode de mesure d'entropie étendue, et l'évaluation des services cloud à l'aide de la méthode intégrée GRA-ELECTRE III. Cette étude démontre que cette méthode hybride de prise de décision multicritère peut aider les utilisateurs à sélectionner le meilleur service cloud en fonction de leurs exigences de QoS. De plus, elle peut être utilisée dans divers domaines tels que la fabrication en nuage, la santé et l'éducation. [14]

b) Afin d'améliorer le processus de prise de décision MCDM un nouveau algorithme est présenter [15] en gérant l'incertitude et en résolvant le RRP(Rank Reversal Problem), pour rendre l'algorithme applicable à divers problèmes de prise de décision multicritère, un modèle de sélection de fournisseur de services en nuage est présenté, ce modèle utilise TOPSIS(Technique for Order Preference by Similarity to Ideal Solution)pour classer les alternatives, l'entropie pour calculer les poids des critères et SVN(Single Valued Neutrosophic) pour traiter l'incertitude. L'étude démontre que le modèle résolve efficacement le problème de renversement de classement (RRP) dans des conditions d'incertitude. Le processus de normalisation, le calcul des valeurs idéales positives (PIS) et négatives (NIS) et la gestion de l'information insuffisante ont été améliorés en intégrant SVN. Des analyses de sensibilité et des mesures statistiques ont validé le modèle et démontrant sa précision et sa robustesse contre le RRP. Mais les méthodes de pondération subjectives n'ont pas été prises en considération. [15]

2.2 L'algorithme de colonie des abeilles (ABC) :

L'algorithme de colonie d'abeilles utilise des sources de nourriture pour représenter des solutions potentielles à un problème d'optimisation, basées sur leur niveau de nectar. La population d'abeilles est divisée en trois groupes qui explorent l'espace de recherche pour trouver des sources de nourriture. Ces groupes ne sont pas les solutions elles-mêmes, mais des itérations qui se concentrent sur les sources de nourriture de meilleure qualité. Lorsqu'une source de nourriture est suffisamment explorée, elle est abandonnée et les

abeilles exploratrices partent à la recherche d'une nouvelle source. En observant le comportement des abeilles à la recherche de nourriture, on remarque qu'elles résolvent des problèmes similaires à celui de la recherche, ce qui permet de simuler leur comportement à l'aide d'algorithmes. [16]

Afin d'approfondir notre compréhension de cet algorithme, nous avons analysé deux articles.

a) **L'algorithme VS-ABC (Venom Simulated Annealing-basedArtificial Bee Colony**):

CMfg Service Composition (CMSC) joue un rôle important dans la résolution de l'interconnexion et l'interopérabilité des ressources et des services CMfg.[17] **D. Liang et al,** proposent un modèle pour la composition de services de fabrication en nuage qui utilise l'algorithme VS-ABC. Les tâches de fabrication sont classées en sous-tâches qui nécessitent des services de fabrication. Les services disponibles sont présentés sous forme de composition de service avec des caractéristiques de qualité, l'algorithme VS-ABC combine les caractéristiques de l'algorithme de colonie d'abeilles artificielles (ABC) pour la recherche locale et l'algorithme VS d'optimisation de recuit simulé pour la recherche globale. L'algorithme VS-ABC améliore la stabilité, la vitesse de convergence et la précision de la solution optimale en évitant les solutions optimales locales. Ce modèle montre une évolution des solutions de composition de services basée sur la qualité de service globale, surpassant les méthodes existantes. Ce modèle peut être étendu à d'autres domaines tels que la fabrication additive, l'Internet des objets industriels. [17]

b) L'algorithme gABC-GWO (guiding artificial bee colony – grey wolf optimization)

Une nouvelle approche pour améliorer la qualité de service des tâches de fabrication en nuage en tenant compte des incertitudes est présentée par **YangBo.al** [18]. Dans le contexte de la fabrication en nuage, l'approche combine plusieurs méthodes et algorithmes pour la composition de services et la sélection optimale. En cas d'incertitude, un algorithme basé sur une structure CMS robuste (rCMS) est développé pour prévoir des services alternatifs. En renforçant la robustesse du CMS pendant la planification, un modèle de composition de services robuste et de sélection optimale rSCOS est créé. L'algorithme gABC-GWO est utilisé pour résoudre les problèmes de rSCOS en

combinant l'optimisation de colonie d'abeilles artificielles (gABC) et l'optimisation de colonie d'abeilles grises (GWO). l'algorithmeg ABC-GWO garantit une sélection optimale des services de fabrication, améliorant ainsi la qualité de service globale dans un environnement incertain et imprévu. [18]

2.3 L'algorithme génétique de tri non dominant II (NSGA-II) :

Le NSGA-II (Non-dominated Sorting Genetic Algorithm II) est une version améliorée et plus efficace de son prédécesseur, le NSGA. Il se propage rapidement et efficacement lorsqu'une région non dominée est identifiée. Son avantage réside dans sa stratégie de préservation de la diversité, sans nécessiter de paramètres fixes. Dans le NSGA-II, une population d'enfants est créée à partir de la population parente, puis triée selon le principe de domination. Les solutions non dominées sont attribuées une valeur de fitness et retirées de la population. Ce processus se répète jusqu'à épuisement des solutions à évaluer. La sélection des sous-ensembles se fait en utilisant une mesure de densité appelée distance de regroupement. [19]

Dans le cadre de notre analyse de cet algorithme, nous avons consulté deux articles afin de renforcer notre compréhension.

a) La composition de services dans la fabrication en nuage pose un défi majeur en raison de la complexité des interactions et des dépendances entre les différents services impliqués. Pour résoudre ce problème, l'article [20] propose une approche basée sur le NSGA-II. Ce dernier souligne des problèmes tels que la prise en compte insuffisante des besoins des participants et les changements dynamiques des indicateurs d'évaluation des services. L'approche utilise l'algorithme NSGA-II avec une stratégie de sélection d'élite afin de traiter les indicateurs d'évaluation des services avec des attributs historiques. Une méthode de décision basée sur les cibles grises est utilisée pour trouver la meilleure solution de Pareto. Cela évite la subjectivité lors de la prise de décision sur la composition des services. L'analyse de cas montrent que le NSGA-II réduit considérablement la distance de la solution optimale par rapport à l'algorithme génétique. Cela démontre la faisabilité du modèle d'optimisation et l'efficacité de l'algorithme proposé. [20]

b) **L’algorithme NSGA-II-SA (Non-dominated Sorting Genetic Algorithm II Simulated Annealing):** Une approche de composition de services pour la fabrication en nuage proposée par **Chen Chen et al.** [21] vise à maximiser les critères de qualité de service (QoS) tout en réduisant l'empreinte carbone. Pour obtenir des solutions Pareto optimales, ils utilisent l'algorithme évolutif hybride multi-objectif NSGA-II-SA.Les résultats montrent que cet algorithme offre de meilleures performances globales en optimisant la durée de vie, le coût, les émissions de carbone et la fiabilité du service. De plus, les auteurs proposent une stratégie d'optimisation afin d’obtenir une solution relativement optimale adaptée aux besoins spécifiques des utilisateurs qui est basée sur TFNAHP(Technique for Order of Preference by Similarity to Ideal Solution) qui est une méthode d'analyse multicritère utilisée pour prendre des décisions basées sur des préférences et des similitudes par rapport à une solution idéale et EWM(Entropy Weight Method)qui est une technique d'évaluation des poids utilisée dans l'analyse multicritère pour déterminer l'importance relative des différents critères. En récapitulation, la méthode proposée est efficace pour la composition de services pour la fabrication en nuage en utilisant l'algorithme NSGA-II-SA pour l'optimisation multi-objectifs et en prenant en compte les critères deQoS et de l'empreinte carbone. [21]

2.4 La colonie de fourmis (ACO) :

En étudiant le comportement d'une colonie de fourmis lorsqu'elles recherchent de la nourriture près de leur nid, on constate qu'elles parviennent à résoudre des problèmes complexes tels que la recherche du chemin le plus court. Les fourmis utilisent des mécanismes simples mais efficaces pour résoudre ces problèmes, ce qui permet de les modéliser facilement à l'aide d'algorithmes. Ainsi, il est possible de simuler leur comportement en utilisant des approches algorithmiques. [22]

Deux articles ont été examinés en détail dans le contexte de cet algorithme pour élargir notre compréhension

a) L’algorithme IACO (Improved Ant Colony Algorithm) :

Une méthode d'optimisation de la composition des services de fabrication en nuage basée sur l'analyse de correspondance-synergie a été proposée par Yin et al. [23]. Cette méthode propose un système d'évaluation de la qualité de la composition des services cloud en utilisant des indicateurs tels que le degré d'appariement des services et le degré de synergie de la composition des services, ainsi que des indices traditionnels tels que le temps et le coût. En tenant compte des intérêts des demandeurs de services et des fournisseurs de ressources, un modèle de préférence à double contrainte est construit et résolu à l'aide de l'algorithme de colonie de fourmis amélioré (IACO). L'efficacité de la méthode proposée est vérifiée à l'aide d'un exemple de service de fabrication en nuage pour la production d'un pare-chocs de voiture. Cette méthode offre une approche pratique pour la sélection optimale des services cloud dans le domaine de la fabrication, en prenant en compte à la fois la correspondance des services et l'interaction de leurs combinaisons [23].

b) L’algorithme ACOS (Ant Colony Optimization with Selection Mechanism):

La fabrication en nuage (CMfg) possède plusieurs caractéristiques et méthodologie de la sélection et de la planification des services, **Cao, Y et al.** [24] examinent l’ensemble de ces caractéristiques en utilisant des facteurs de QOS tels que le temps, la qualité, le coût et service ces derniers sont pris en compte lors du choix et de l'organisation des services CMfg. Contrairement aux recherches antérieures qui utilisaient des approches pondérées linéaires pour réduire directement le problème d'optimisation multi-objectif à un problème à objectif unique, les auteurs utilisent une théorie de prise de décision floue pour convertir les valeurs des facteurs de QOS en scores de supériorité relative. Lors de la création d'un objectif global d'optimisation, ces degrés sont ensuite combinés en utilisant des facteurs de pondération déterminés par le processus analytique hiérarchique (AHP). Le modèle existant de sélection et de planification des services est également modifié pour l'optimisation par colonies de fourmis et peut être facilement ajusté pour inclure d'autres critères, tels que l'énergie, en convertissant leurs valeurs en degrés de supériorité relative et en dérivant les coefficients de pondération correspondants à l'aide de l'AHP. Les résultats des simulations montrent l'utilité du modèle proposé et l'efficacité d'ACOS par rapport à d'autres algorithmes populaires. [24]

2.5 Le Modèle Mathématique (MM) :

Le modèle mathématique est une traduction de la réalité pour pouvoir lui appliquer les outils, les techniques et les théories mathématiques, puis généralement, en sens inverse, la traduction des résultats mathématiques obtenus en prédictions ou opérations dans le monde réel tels que la fabrication industriels. [25]

Pour approfondir notre connaissance de cet algorithme, nous nous sommes référés à deux articles pertinents

a) La planification des services de fabrication en nuage en temps réel est un défi complexe qui nécessite une approche efficace et adaptée. Dans cette optique, une méthode novatrice basée sur la simulation dynamique axée sur les données est proposée par **Zhou. L et al.** [26]. Un modèle Mathématique est créé pour gérer planification dynamique des services de fabrication en nuage. Les règles de planification basées sur le temps de service, le temps logistique et l'état de la file d'attente des sous-tâches des services candidats sont utilisées dans la méthode proposée. La stratégie de simulation intègre des informations sur les tâches et les services en temps réel pour sélectionner les meilleures règles de planification. Une étude de cas d'usinage par commande numérique dans le contexte de la fabrication en nuage est utilisée pour tester les performances de la méthode. Les résultats montrent que la méthode permet de créer des plans de service solides et de choisir les meilleures stratégies de planification dans un environnement de fabrication en nuage instable. [26]

b) La planification des tâches dans les systèmes de fabrication en nuage est une problématique complexe qui nécessite une approche rigoureuse et bien structurée. Dans cette perspective, l'article [27] présent un modèle permet d'optimiser l'affectation des opérations et des services logistiques en fonction de divers objectifs, tels que la minimisation des coûts d'opérations et de transport. De plus, l'article discute également de l'importance de la définition d'objectifs de qualité de service (QoS) et propose une méthode pour diviser les tâches en sous-

tâches et les comparer aux services disponibles sur la plateforme de fabrication en nuage. Cette étude montre que ce modèle peut créer des plans de production solides en tenant compte de divers types de partage, y compris le partage logistique, le partage des opérations et le partage de la production. [27]

2.6 L'algorithme génétique (GA) :

L'algorithme génétique est une approche méta heuristique initialement conçue pour résoudre des problèmes impliquant des variables discrètes. Il s'agit d'un algorithme élitiste qui repose sur une population de solutions. Au fil de plusieurs générations, cette population évolue en sélectionnant les individus les plus performants à chaque étape. Certains individus se reproduisent, tandis que d'autres sont supprimés, ce qui permet de transmettre un héritage génétique de génération en génération et favorise la survie des individus les mieux adaptés. Ce processus est répété jusqu'à ce qu'un certain critère d'arrêt soit atteint. L'algorithme génétique offre ainsi une approche puissante pour l'optimisation de problèmes complexes en explorant de manière itérative l'espace des solutions et en favorisant l'émergence de solutions de haute qualité. [16]

Dans le cadre de notre analyse de cet algorithme, nous avons consulté deux articles afin de renforcer notre compréhension.

a) Afin de se concentrer sur le défi de la planification en temps réel des tâches dans le domaine de la fabrication en nuage, l'étude présentée [28] propose une approche novatrice basée sur un algorithme génétique multi objectif (GA). Pour relever ce défi, **Ahn.G et al**[2],proposent l'utilisation d'un algorithme génétique multi objectif (GA). Présent une étude aborde le défi de la planification en temps réel des tâches dans la fabrication en nuage en utilisant un algorithme génétique multi objectif (GA). L'objectif est d'optimiser la satisfaction des clients en tenant compte de facteurs tels que le coût, la qualité et la ponctualité. Cependant, la planification en temps réel et multi objectif constitue un défi dans le CMfg. Les auteurs présentent l'algorithme utilisé sous forme de programmation binaire en nombres entiers. Les résultats expérimentaux montrent que leur approche basée sur cet algorithme produit des plannings presque optimaux pour les tâches mineures et des plannings efficients en temps réel pour les scénarios complexes. L'intégration de cette méthode dans les plateformes existantes de CMfg est

soulignée, avec des plans pour améliorer un ordonnanceur basé sur un modèle d'apprentissage profond en utilisant l'algorithme génétique multi objectif recommandé. [28]

b) Afin de relever le défi de l'intégration de la logistique avant et de la logistique inverse dans le domaine de la fabrication en nuage. **Moghaddam, S et al.** [29] proposent une approche novatrice qui met l'accent sur le transport des produits retournés. Cette étude propose une architecture basée sur des agents qui remplissent différentes tâches telles que la détermination du prix de rachat, la détermination des coûts des flux inverses, la détermination du prix de revente, la prévision du taux de retour, la redistribution des produits retournés, la planification du transport, la planification de l'approvisionnement et la planification des processus de fabrication. Un algorithme génétique est utilisé pour optimiser les performances de l'agent planificateur de transport démontrant ainsi l'efficacité et l'applicabilité de cette approche. Cette étude souligne l'importance de la prise en compte la logistique inverse et le transport des produits retournés dans les systèmes CMfg. [29]

2.7 ASSCCT Approche de sélection de services CMfg consciente du temps:

Une nouvelle approche pour la sélection de services de fabrication en nuage est présentée [30] en tenant compte des préférences incertaines des utilisateurs et des prédictions QoS inconnues. L'approche se concentre sur la prédiction de la valeur QoS temporelle et l'évaluation globale de la QoS sensible au temps. Elle utilise des séries temporelles multidimensionnelles et des techniques d'apprentissage automatique pour prédire la qualité future des services de fabrication en nuage. Un algorithme de recherche de similarité est utilisé pour améliorer les préférences incertaines des utilisateurs. Les résultats démontrent que la méthode proposée améliore significativement la précision de la prédiction de la QoS tout en maintenant un temps de traitement acceptable pour les utilisateurs. Cette méthode résout également le problème de la préférence unilatérale de l'utilisateur en utilisant une fonction de contention QoS temporelle. Les résultats expérimentaux montrent que la méthode proposée surpasse les méthodes classiques de prévision des séries chronologiques et sélectionne les meilleurs services en réduisant les erreurs dans l'évaluation des préférences des utilisateurs [30].

2.8 l’algorithme Improved-TC: (Teaching-Learning-Based Optimization (TLBO) Crisscross Optimization (CSO)):

Une approche novatrice est proposée par **J. Zeng et al.** [31] pour la composition de services de fabrication en nuage, accompagnée d'une méthode de sélection optimale basée sur l'algorithme TC amélioré. Cet algorithme, résultant de la combinaison dès Les algorithmes TLBO (Teaching-Learning-Based Optimization) et CSO (Cuckoo Search Optimization) sont spécialement conçus pour résoudre des problèmes à grande échelle. Ils utilisent la recherche unidimensionnelle pour permettre à certaines personnes qui sont piégées dans un optimum local de s'échapper de l'itération. De plus, ils utilisent des requêtes "Skyline" pour sélectionner les services optimaux parmi les services candidats, améliorant ainsi la qualité de la solution et la convergence de l'algorithme pendant la phase d'initialisation. Selon cet article, la méthode proposée, Improved-TC, présente une vitesse de convergence plus rapide et une stabilité plus élevée. Les auteurs recommandent donc l'utilisation de leur méthode pour améliorer l'efficacité de la composition de services dans les environnements à grande échelle. Cependant, il convient de noter que cette méthode n'est pas adaptée aux problèmes de composition de services et de sélection optimale dynamiques (SCOS) [31].

3 Comparaison entre les approches étudiées :

Dans le tableau 2, nous représentons une comparaison entre les différentes solutions étudiées qui abordent le problème de la sélection des services CMfg.

Algorithme	référence	Paramètres QOS	Avantages	Inconvenient	Domains d'utilisation
(MCDM) Hybride	**M. Liu, et al [14]**	✓ le coût ✓ le temps de réponse ✓ Succès ✓ la fiabilité ✓ la réputation	-Un bon traitement de la subjectivité, le flou et l'incertitude des attributs de QoS -refléter les préférences faibles et l'indifférence entre les alternatives.	-L'approche se concentre principalement sur les attributs de qualité de service (QoS), sans prendre en compte d'autres aspects importants tels que la sécurité	-la santé - l'éducation. - les finances. -les entreprises.
	Abdelaziz A. S, et al [15]	✓ le cout ✓ la latence ✓ les performances de cpu. ✓ les performances de lecture/écriture aléatoire sur disque et sur la mémoire	-amélioration de processus de prise de décision sous l'incertitude avec une grande précision et une robustesse contre le RRP.	-cette recherche n'a pas pris en compte les approches basées sur la pondération subjective, qui déterminent les poids des critères en fonction des jugements des décideurs.	-services de cloud
Algorithme	**référence**	**Paramètres QOS**	**Avantages**	**Inconvenient**	**Domains d'utilisation**
L'algorithme ABC	**D. Liang et al. [17]**	✓ Le coût ✓ Le temps d'energie ✓ La fiabilité ✓ la disponibilité	-Garantit la vitesse de convergence -Améliore la tendance du problème à tomber dans la solution optimale locale -Améliore la précision de la solution optimale -Une stabilité et une vitesse est remarquable	-Ne traite pas en profondeur de la collecte et de la mise à jour des données en temps réel des ressources de fabrication. Considérer -Adéquatement pour le traitement des commandes urgentes seulement	-Fabrication additive. -L'Internet des objets industriels. - les villes intelligentes.

	Yang.Bo et al. [18]	✓ Le temps ✓ Le cout ✓ La disponibilité ✓ La réputation	-Améliorant de la qualité de service globale dans un environnement incertain et imprévu.	-La nécessité de développer des méthodesde gestion pour les changements de requis pendant l'exécution des tâches	-La fabrication en nuages -la fabrication de produits électroniques - les industrieschimiques - la fabrication demachines
Algorithme	**référence**	**Paramètres QOS**	**Avantages**	**Inconvenient**	**Domains d'utilisation**
L'algorithme (NSGA-II)	**Guo, K et al. [20]**	✓ Le cout ✓ Le temps	-Considération globale des intérêts des acteurs multiples dans la fabrication en nuages	-La nécessite de mener des recherches plus approfondies pour rendre le modèle plus pertinents pour les besoins réels pour la composition de services	- La fabrication en nuages
	Chen, C et al. [21]	✓ temps ✓ cout ✓ fiabilité ✓ Émissions de carbone liées au service	-Une meilleure performance globale dans la tâche d'optimisation de la composition de services en fonction de différents objectifs tels que la consommation d'énergie et le coût.	L'algorithme est spécifiquement optimisé pour la composition des services dans un environnement de fabrication de cloud à faible émission de carbone, ce qui signifie qu'il peut ne pas être applicable dans d'autres environnements.	-un environnement defabrication en nuage à faible émission de carbone
Algorithme	**référence**	**Paramètres QOS**	**Avantages**	**Inconvenient**	**Domains d'utilisation**

L'algorithme ACO	**Yin, C et al [23]**	✓ Le cout ✓ Le temps ✓ La fiabilité ✓ Le degré de correspondance des services ✓ Le degré de synergie de la composition services	- Modèle d'optimisation à double contrainte qui prend en compte les contraintes des demandeurs de service et des fournisseurs de ressources de service.	-Absence de considération des indicateurs tels que la consommation d'énergie et la crédibilité dans le modèle proposé pour mieux refléter la réalité.	-la fabrication automobile - la fabrication en nuages
	Cao, Y et al [24]	✓ Le temps ✓ Le cout ✓ La qualité	- Le modèle établi prend en compte plusieurs critères simultanément, ce qui permet de trouver des solutions qui optimisent différents aspects de la sélection et de la planification des services CMfg	- La qualité du produit n'est pas définie de manière exhaustive, car le service doit prendre en compte divers aspects tels que l'évaluation globale de la qualité d'usinage, le taux de réussite, la fiabilité, et d'autres critères pertinents.	- Industrie manufacturière
Algorithme	**référence**	**Paramètres QOS**	**Avantages**	**Inconvenient**	**Domains d'utilisation**
Modèle mathématique (MM)	**Zhou, L et al. [26]**	✓ Temps de service ✓ Temps logistique	- La méthode de planification proposée basée sur la simulation permet de sélectionner de meilleures stratégies de planification en utilisant des informations en temps réel sur l'état des services et l'exécution des tâches.	-Certains détails essentiels tels que les coûts de service et la qualité ne sont pas pris en compte dans les objectifs de planification.	- Fabrication industrielle - Fabrication en nuage
	Delaram , J. et al [27]	✓ Le cout logistique ✓ Le cout des opérations	- Le modèle vise à minimiser les coûts d'opération et logistiques, ce qui peut conduire à une utilisation plus efficace des	- Les paramètres QoS tels que le temps et la qualité sont pris en compte de manière limitée.	- Fabrication en nuage - Fabrication industrielle

			ressources et à une réduction des coûts de production		
	Notre solution	✓ Cout ✓ Latence ✓ Disponibilité ✓ Fiabilité ✓ Temps de réponse ✓ Succès ✓ Documentation ✓ Débit	-La structure modulaire du modèle mathématique offre une flexibilité pour prendre en compte différentes contraintes de QoS et les combiner de manière adaptable. -Le modèle mathématique permet une évaluation objective et quantifiable des performances des différents algorithmes de sélection de services CMfg, ce qui permet de démontrer l'efficacité et les améliorations apportées par notre nouvel algorithme par rapport aux approches existantes	- Utilisation d'un ensemble de données de dataset qui est n'est pas issu du cloud manufacturing	- Fabrication en nuage
Algorithme	**référence**	✓ **Paramètres QOS**	**Avantages**	**Inconvenient**	**Domains d'utilisation**
L'algorithme génétique (GA	**Ahn, G et al. [28]**	✓ Le cout ✓ La qualité La fiabilité	-ce modèle maximise la satisfaction de client en partageant les ressources de fabrication entre les entreprises, optimisant ainsi les retards, les coûts, la qualité et la fiabilité, et permettant une planification en temps réel pour une meilleure réactivité et efficacité	-la méthode utilisée peut générer des solutions infaisables ou inférieures dans certaines situations.	- Fabrication en nuage

L'algorithme génétique (GA **Algorithme**	**Moghad-dam, S. N et al. [29]**	✓ Le temps ✓ Le coût	- N'intègre pas seulement la logistique inverse et directe mais aussi les flux de transports et ajuste les services de logistique inverse en prenant des décisions basées sur les politiques, les attributs et les besoins de cloud.	-L'algorithme génétique utilisé dans cette étude conduit à une solution quasi optimale	- Fabrication en nuage
	référence	✓ **Paramètres QOS**	**Avantages**	**Inconvenient**	**Domains d'utilisation**
QoSTS-TA	**Ying Yu et al. [30]**	✓ La capacité de stockage. ✓ La disponibilité. ✓ La sécurité. ✓ La fiabilité.	-Évaluer la qualité de service temporelle et tenir compte des prévisions de QoS imprécises .	-L'utilisation des préférences de l'utilisateur pour évaluer les services peut être considérée comme unilatérale, ce qui peut affecter la fiabilité de la sélection de services.	-La fabrication en nuage (CMfg) -l'automatisation industrielle
l'algorithme Improved-TC	**J. Zeng et al. [31]**	✓ Réputation ✓ Cout ✓ Temps ✓ Fiabilité	-l'algorithme utilisé est efficace et stable pour améliorer la qualité des solutions et résoudre les problèmes de composition de services à grande échelle et de sélection optimale (SCOS)	-Une limitation de l'application de cet algorithme dans des environnements de fabrication en nuage plus complexes et évolutifs	-La fabrication en nuage (CMfg)

Tableau 2:une comparaison des approches étudiées

4 Classification des approches de sélection :

Voici une classification des articles selon la solution de base utilisée :

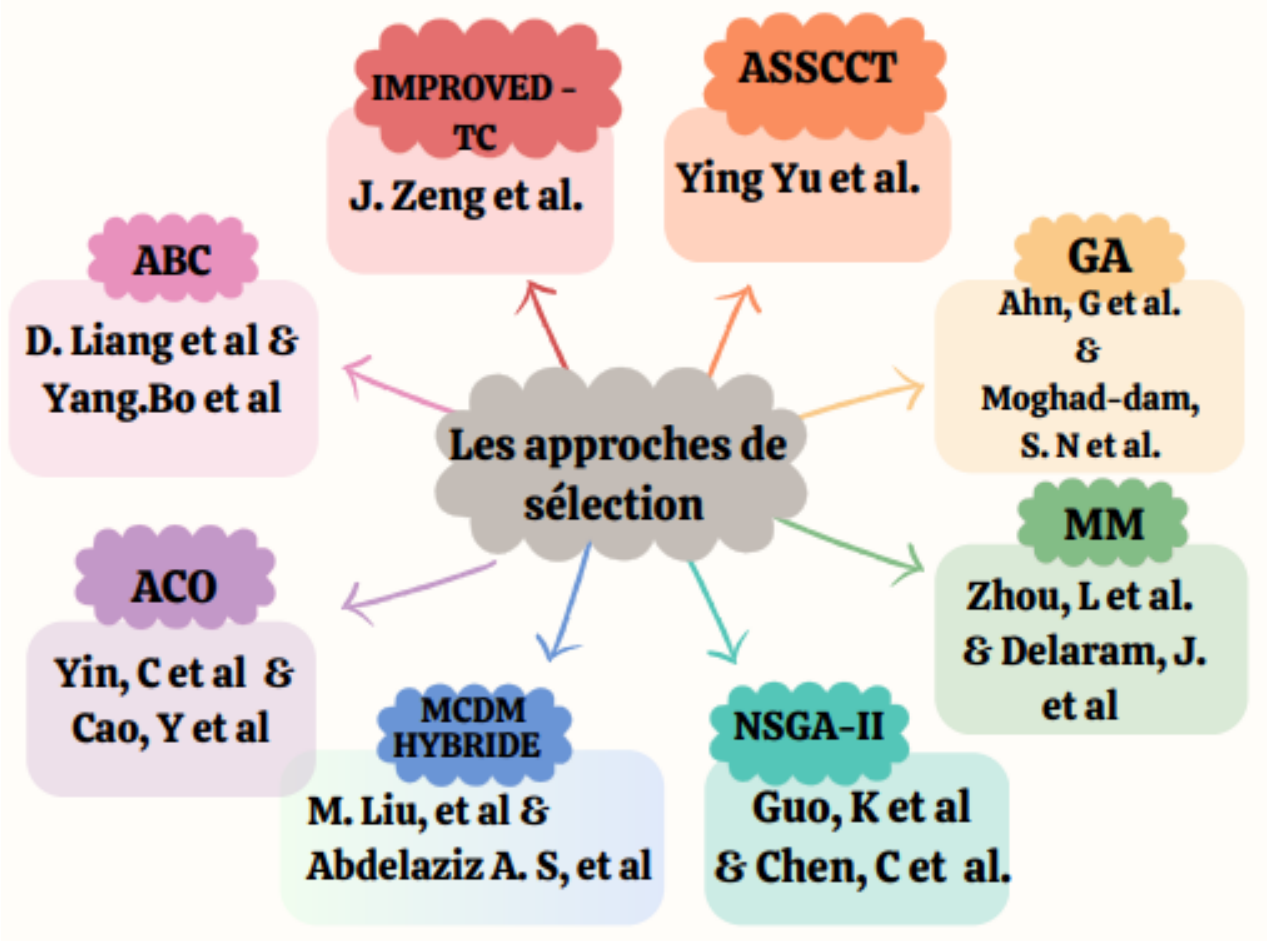


Figure 2: Classification des approches de sélection étudiée.

5 Conclusion :

En conclusion, le choix des services cloud est une étape important pour les entreprises souhaitant utiliser le cloud manufacturing pour délocaliser les processus de production et des stockages des données. Bien que de nombreuses méthodes des sélections de services cloud aient développées la plupart d'entre elles se concentrent sur l'optimisation de l'allocation des ressources de la qualité de service.

Dans ce chapitre, nous effectuons une analyse comparative des différentes méthodes actuelles de sélection des services Cloud Manufacturing, en mettant l'accent sur les critères de qualité de service (QoS). Cette classification nous permet de mettre en évidence les avantages et les inconvénients de chaque approche et algorithmes pour mieux positionner notre problématique par rapport aux solutions étudiées.

Dans le prochain chapitre, nous proposons un nouvel algorithme de sélection de service CMfg qui classe tous les services candidats et trouve la combinaison proche de l'optimale du service Cloud sous différents paramètres QoS et différentes contraintes QOS.

Chapitre 3 : Nos Approche de sélection des services cloud manufacturing sous différents paramètres QOS

1 Introduction :

Dans ce chapitre, nous mettrons en lumière la mise en œuvre et les performances de notre solution. Pour ce faire, nous présenterons le logiciel que nous avons développé pour résoudre les problèmes de la sélection optimale du service de fabrication en nuage. En nous appuyant sur les concepts abordés dans le chapitre précédent. De plus, nous décrirons les ressources matérielles et les outils de développement utilisés ainsi que des captures d'écran qui montrent l'interface de notre application web.

2 Le processus d'exécution d'une demande d'un client dans la plate-forme CMfg :

Le processus d'exécution d'une demande d'un client dans la plate-forme CMfg comprend plusieurs étapes qui figurent dans la figure 3. Tout d'abord, la plate-forme CMfg reçoit la demande du client concernant une tâche de fabrication spécifique. Ensuite, la demande est minutieusement analysée afin de comprendre les besoins et les spécifications du client, en tenant compte des exigences QoS telles que le Temps de réponse, prix, latence etc. La tâche estensuite décomposée en plusieurs sous-tâches plus petites, chacune pouvant être traitée individuellement. Pour chaque sous-tâche, la plate-forme CMfg effectue une recherche exhaustive de tous les services qualifiés disponibles dans son réseau. En se basant sur les exigences de qualité de service. La plate-forme CMfg sélectionne un pool de services candidatspour chaque sous-tâche. Ensuite, il procède à la composition des services en choisissant un ouplusieurs services de chaque pool de services candidats. Cela permet de créer un service composite qui répond aux besoins initiaux de la demande du client

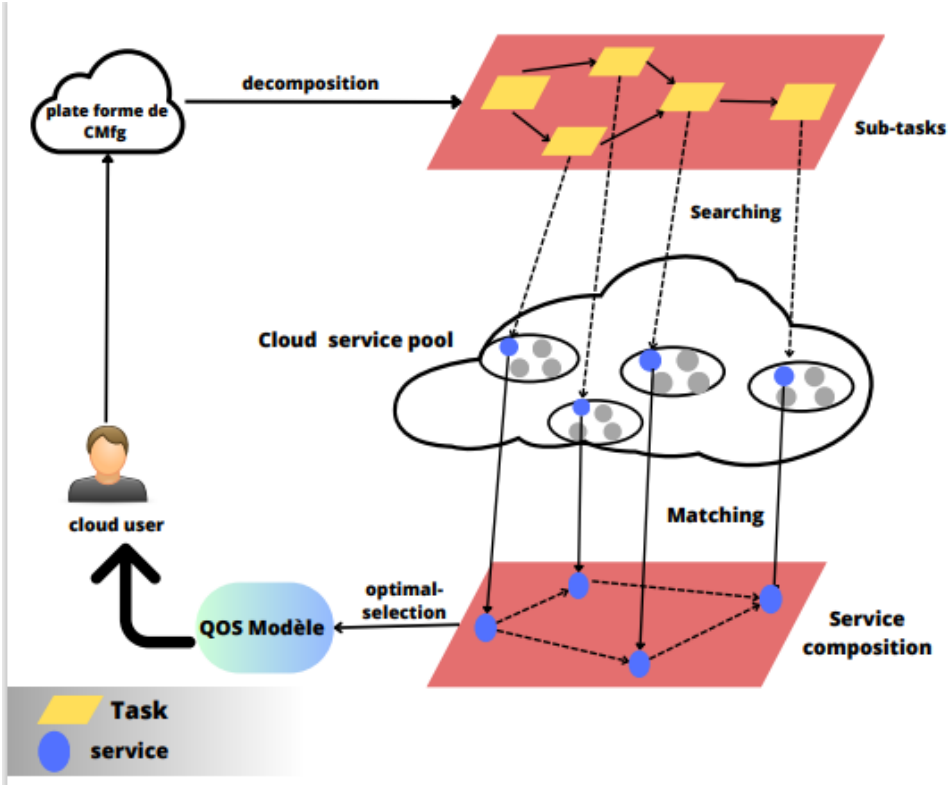


Figure 3:Processus de sélection optimale de la composition des services [32]

3 Les paramètres de qualité de services

Dans notre solution, huit paramètres de qualité de service (QoS) ont été pris en compte, qui peuvent être classés en deux types :

- **Les paramètres négatifs** : qui sont temps de réponse, coût et latence.
- **Les paramètres positifs** : qui sont disponibilité, débit, succès, fiabilité et documentation.

Les valeurs des paramètres positifs doivent être maximisées, tandis que les valeurs des paramètres négatifs doivent être minimisées.

4 L’approche globale :

L'approche globale adoptée vise à explorer toutes les combinaisons envisageables des compositions de services et à sélectionner la composition optimale. Cette méthode s'avère particulièrement efficace lorsque le problème est de petite taille. Toutefois, l'extensibilité de cette approche reste limitée en raison de la complexité croissante des algorithmes de recherche, qui augmente exponentiellement avec la taille du problème.

5 L’algorithme de sélection des services CMfg a des contraintes QOS prioritaire

Notre algorithme joue un rôle essentiel dans le processus de sélection du service CMfg en évaluant et classant tous les services candidats disponibles. Son objectif principal consiste à identifier la combinaison proche de l’optimale parmi les différentes options de services CMfg envisageables. Pour y parvenir, l'algorithme évite une approche exhaustive qui consisterait à explorer toutes les configurations possibles de plans d'exécution.

Il faut calculer les contraintes locales à chaque sous-tâche afin de sélectionner les services les plus appropriés pour chacune d'entre elles. Ces contraintes sont calculées en tenant compte des contraintes globales attribuées aux paramètres de qualité de service (QoS) tels que temps de réponse, coût, disponibilité, débit, succès, fiabilité, latence, documentation.

Ce modèle est représenté comme suit :

- Une tâche qui est décomposé en un ensemble de sous tache

$T= \{ST_1, ST_2, ST_3,\ldots, ST_n\}$

- Un ensemble de service candidats pour chaque sous tâche :

$ST_1= \{CS_{1.1}, CS_{1.2}, CS_{1.3}, \ldots, CS_{1.n}\}$

$ST_2= \{CS_{2.1}, CS_{2.2}, CS_{2.3}, \ldots, CS_{2.n}\}$

$ST_m= \{CS_{m.1}, CS_{m.2}, CS_{m.3}, \ldots, CS_{m.n}\}$

- Un ensemble de temps de réponse des services candidats

$T = \{T_{1.1},.. , T_{1.n} , T_{2.1} ,.. , T_{2.n} ,T_{3.1},.. , T_{3.n} , T_{m.1} ,.. ,T_{m.n}\}$

- Un ensemble de coût des services candidats

$C = \{C_{1.1},.. , C_{1.n} , C_{2.1} ,.. , C_{2.n} ,C_{3.1},.. , C_{3.n} , C_{m.1} ,.. ,C_{m.n}\}$

- Un ensemble de Latence des services candidats

$L = \{L_{1.1},.. , L_{1.n} , L_{2.1} ,.. , L_{2.n} ,L_{3.1},.. , L_{3.n} , L_{m.1} ,.. ,L_{m.n}\}$

- Un ensemble de disponibilité des services candidats

$A = \{A_{1.1},.. , A_{1.n} , A_{2.1} ,.. , A_{2.n} ,A_{3.1},.. , A_{3.n} , A_{m.1} ,.. ,A_{m.n}\}$

- Un ensemble de fiabilité des services candidats

$R = \{R_{1.1},.. , R_{1.n} , R_{2.1} ,.. , R_{2.n} ,R_{3.1},.. , R_{3.n} , R_{m.1} ,.. ,R_{m.n}\}$

- Un ensemble de succès des services candidats

$S = \{S_{1.1},.. , S_{1.n} , S_{2.1} ,.. , S_{2.n} ,S_{3.1},.. , S_{3.n} , S_{m.1} ,.. ,S_{m.n}\}$

- Un ensemble de débit des services candidats

$Th = \{Th_{1.1},.. , Th_{1.n} , Th_{2.1} ,.. , Th_{2.n} ,Th_{3.1},.. , Th_{3.n} , Th_{m.1} ,.. ,Th_{m.n}\}$

- Un ensemble de documentation des services candidats

$D = \{D_{1.1},.. , D_{1.n} , D_{2.1} ,.. , D_{2.n} ,D_{3.1},.. , D_{3.n} , D_{m.1} ,.. ,D_{m.n}\}$

Dans le figure , Un vecteur de priorité est utilisé pour évaluer l'importance relative des contraintes QoS, tel que QoS_1 doit avoir la plus haute priorité et QoS_8 doit avoir la priorité la plus basse. $Pr = \{QoS_1, QoS_2, QoS_3, QoS_4, QoS_5, QoS_6, QoS_7, QoS_8\}$.

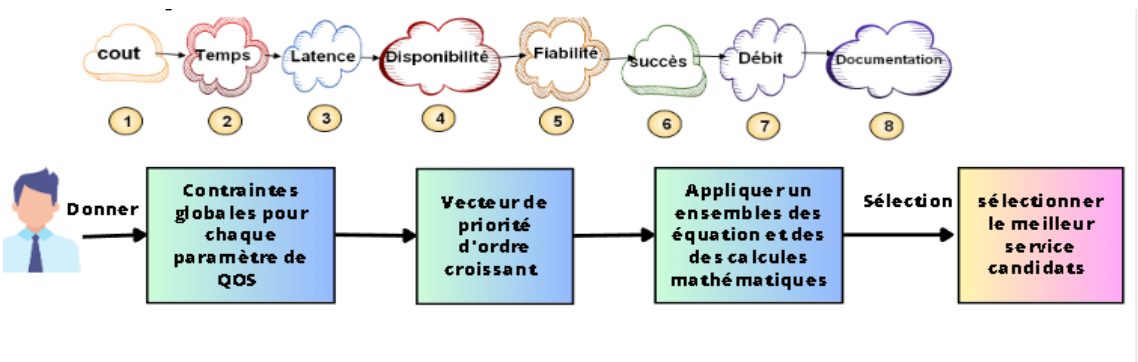


Figure 4:L'architecteur de notre premier solution par rapport le vecteur de priorité d'ordre décroissante.

- **Les paramètres négatifs (Temps, coût et latence) :**

Il faut calculer les valeurs V_Qos(N)$_{[i][j]}$, qui représente la valuation des paramètres Qos(N)$_{[i][j]}$, minimisées on les comparants à la valeur maximal du paramètre Qos(N):

$$V_{Qos(N)_{[i][j]}} = \frac{Qos(N)_{[i][j]}}{MaxQos(N)} \quad \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \quad \forall\, N \in \{cost, time, latency\} (\mathbf{1})$$

$$\textbf{Où } MaxQos(N) = Max\left[Qos(N)_{[i][j]}\right] (\mathbf{2})$$

Ensuite, On veut calculer la moyenne de chaque sous tâche, en utilisant les valeur V_Qos(N)$_{[i][j]}$, pour chaque sous tâche.

$$AVG\_VQos(N)_{[i]} = \frac{\sum_{j=1}^{ci} V\_Qos(N)_{[i][j]}}{ci} \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \forall \in N\, \{cost, time, \ latency\} (\mathbf{3})$$

Puis, Dans **l'équation (4)** il faut calculer la valeur VC(N) qui représente la valuation des contraintes QOS en les comparants à la valeur Max_C(N), qui représente la somme de la valeur maximale du paramètre QOS de chaque sous tâche.

$$VC(N) = \frac{Constraint(N)}{Max\_C(N)} \quad \forall\, N \in \{time, cost, latency\} (4)$$

Où $Max\_C(N) = \sum_{i=1}^{n}\left(Qos(N)_{[i][j]}\right) \forall\, i \in \{1,n\}, \forall\, j\, \{1,ci\}, \forall\, N \in \{cost, time, latency\} (5)$

Ensuite, pour exprimer la valuation des paramètres QOS augmenté de contraintes QOS, il faut calculer VN_QOSC(N) pour chaque contrainte QOS en utilisant **l'équation (6)** :

$$VN_{QosC(N)_{[i]}} = \frac{VC(k) * AVG_{VQos(N)_{[i]}}}{\sum_{i=1}^{n} AVG_{VQos(N)_{[i]}}} \quad \forall\, i \in \{1,n\}, \forall\, N \in \{cost, time, latency\} (6)$$

Enfin, dans **l'équation (7)**, en multipliant chaque valeur de VN_QosC(N)$_{[i]}$ par la valeur de MaxN_C pour calculer la valeur de contrainte locale de chaque sous tâche

$$\text{LC (N)}_{[i]} = V\_QosC(N)_{[i]} * \text{MaxN\_C (N)} \quad \textbf{(7)}$$

- **Les paramètres positifs (disponibilité, fiabilité, succès, débit, documentation) :**

En premier lieu, il est nécessaire de procéder au calcul des valeurs$V_Qos(P)_{[i][j]}$ Ces valeurs reflètent la valuation des paramètres $Qos(P)_{[i][j]}$ minimisés lorsqu'ils sont comparés à la valeur maximale du paramètre QOS obtenue grâce à **l'équation (8).**

$$V\_Qos(P)_{[i][j]} = \frac{Qos(P)_{[i][j]}}{MaxQos(P)} \quad \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \quad \forall\, P \in \{A,R,S,Th,D\} \quad (\mathbf{8})$$

Où $MaxQos(P) = Max\left[Qos(P)_{[i][j]}\right] (\mathbf{9})$

Ensuite, en utilisant les valeurs V_Qos(P)$_{[i][j]}$ pour chaque sous tâche, l'équation **(10)** permet de calculer la moyenne de chaque sous tâche

$$AVG\_VQos(P)_{[i]} = \frac{\sum_{j=1}^{ci} V\_Qos(P)_{[i][j]}}{ci} \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \forall\, P \in \{A,R,S,Th,D\} (10)$$

Ensuite, la valeur VC(P) est calculée à l'aide de l'équation (11), ce qui représente la valuation des contraintes QOS par rapport à la valeur MaxP_C(P). Cette dernière correspond au produit de la valeur maximale du paramètre QOS de chaque sous-tâche.

$$VC(P) = \frac{Constraint(P)}{Max\_C(P)} \forall\, P \in \{A,R,S,Th,D\} \quad (11)$$

Où $MaxP\_C = \prod_{i=1}^{n}\left(Qos(P)_{[i][j]}\right) \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \forall\, P \in \{A,R,S,Th,D\}$ (12)

Par la suite, une valeur VP_QOSC(P) est calculée pour chaque contrainte QOS afin de représenter la valuation des paramètres QOS avec l'incorporation des contraintes QOS, en utilisant l'équation (13).

$$VP\_QosC(P)_{[i]} = AVG\_VQos(P)_{[i]} * \sqrt[ci]{\frac{VC(P)}{\prod_{i=1}^{n} AVG\_VQos(P)_{[i]}}} \forall\, i \in \{1,n\}, \forall\, P \in \{A,R\} \quad (13)$$

Finalement, le calcul de la valeur de contrainte locale de chaque sous-tâche est réalisé à l'aide de l'équation (14) :

$$LC(P)_{[i]} = VP\_QosC(P)_{[j]} * \sqrt[ci]{MaxP\_C} \quad (14)$$

Le calcul des valeurs DIFF, effectué à l'aide de **l'équation (15),** permet de quantifier la différence des valeurs QOS pour chaque CS par rapport à la nouvelle contrainte locale

$$DIFF(k)_{[i][j]} = Qos(k)_{[i][j]} - LC(k)_{[i]} \ \forall i \in [1,n], \forall j \in [1,Ci], \forall k \in [P,N] \quad (15)$$

Ensuite, il est nécessaire de vérifier si chaque service candidat satisfait la contrainte locale en fonction de la valeur de différence (DIFF). Si le paramètre QOS est positif et que la valeur de DIFF est positive, alors la contrainte locale est satisfaite.

Si la valeur $DIFF(k)_{[i][j]} > 0$ alors $ResAtt\_Qos(k)_{[i][j]} = 1$

Sinon $ResAtt\_Qos(k)_{[i][j]} = 0$

De même, si le paramètre QOS est négatif et que la valeur de DIFF est négative, alors la contrainte locale est satisfaite.

Si la valeur $DIFF(k)_{[i][j]} < 0$ alors $ResAtt\_Qos(k)_{[i][j]} = 1$

Sinon $ResAtt\_Qos(k)_{[i][j]} = 0$

En conclusion, si toutes les valeurs $ResAtt\_Qos(k)_{[i][j]}$ des paramètres QOS sont satisfaites, alors ces services sont sélectionnés comme la meilleure composition de service. Si il trouvent dans la même sous-tâche plusieurs CS avec $ResAtt\_Qos(k)_{[i][j]} = 1$, il est nécessaire de choisir un CS parmi eux. Ce choix sera effectué en respectant la priorité imposée par le client. Le vecteur de priorité QOS est défini comme suit : {Cost, Time, Latency, Availabilty, Reliability, Successability, Throughput, Documentation}, Le CS qui sera sélectionné est celui ayant la valeur de coût la plus basse.

5.1 Pseudo code de l'algorithme de la sélection de service CMfg a des contraintes QOS avec des priorités décroissantes

Algorithm parameters

Set of sub tasks STi = {ST_1,… ST_{1-n},ST_n}

Set of candidate service Cij = {$CS_{1.1}$, $CS_{2.1}$, …….CS_{nm}}

Set of constraints C = {C1, C2, C3, C4, C5, C6, C7, C8}

Set of global constraints GC={ GC [0], …,GC [2], GC [3],…, GC [7]}

Set of priority vector Pr = { Cost, Time, Latency, Availability , Reliability, Successability, Throughput, Documentation}

Begin

For$(i = 0 \rightarrow n)$

Compute $LC_{[i]}$ using eq.7 Or eq.14

End

For$(i = 0 \rightarrow n)$

For$(j = 0 \rightarrow Cj)$

$DIFF(N)_{[i][j]} = Qos(N)_{[i][j]} - LC_{[i]}$

$DIFF(P)_{[i][j]} = Qos(P)_{[i][j]} - LC_{[i]}$

If ($DIFF(N)_{[i][j]} < 0$) **then** $ResAtt\_Qos(N)_{[i][j]} = 1$ **else** $ResAtt\_Qos(N)_{[i][j]} = 0$

If ($DIFF(P)_{[i][j]} < 0$) **then** $ResAtt\_Qos(P)_{[i][j]} = 1$ **else** $ResAtt\_Qos(P)_{[i][j]} = 0$

End

End

For$(i = 0 \rightarrow n)$

For$(j = 0 \rightarrow Cj)$

If($ResAtt\_Qos(C)_{[i][j]}$ and $ResAtt\_Qos(T)_{[i][j]}$ and $ResAtt\_Qos(L)_{[i][j]}$ and $ResAtt\_Qos(A)_{[i][j]}$ and $ResAtt\_Qos(R)_{[i][j]}$ and $ResAtt\_Qos(S)_{[i][j]}$ and $ResAtt\_Qos(Th)_{[i][j]}$) and $ResAtt\_Qos(D)_{[i][j]}$) **then**

$BestCS_{[i][j]} = 1$

End

End

End

5.2 Exemple :

Afin d'illustrer l'algorithme proposé, nous prendrons comme exemple une composition de service comprenant trois sous-tâches séquentielles. Chaque sous-tâche est composée de trois services candidats, Parmi ces services candidats, trois paramètres QOS sont négatifs et cinq paramètres QOS sont positifs. Cette configuration est représentée dans les deux tableaux suivant :

subtask	vices candida	cost	time	latency	Availabilty	Reliability	Successability	Throughput	Documentatic
s1	CS11	93,76	302,75	187,75	0,89	0,73	0,9	7,1	0,32
	CS12	132,94	482	1	0,85	0,73	0,95	16	0,02
	CS13	90,57	3321,4	2,6	0,89	0,73	0,96	1,4	0,96
s2	CS21	77,18	126,17	22,77	0,98	0,67	1	12	0,89
	CS22	72,08	107	58,33	0,87	0,73	0,95	1,9	0,93
	CS23	111,99	107,57	18,21	0,8	0,67	0,81	1,7	0,61
s3	CS31	87,63	255	40,8	0,98	0,67	0,99	1,3	0,04
	CS32	91,6	136,71	11,57	0,76	0,6	0,76	2,8	0,08
	CS33	155	102,62	0,93	0,91	0,67	0,97	15,3	0,91
Global Contrain		**270**	**10000**	**100**	**0,4**	**0,2**	**0,8**	**1**	**0,001**

Tableau 3:Affectation des valeur QOS à chaque CS

Supposons que le service composite huit contraintes globales QOS définies comme suit : Cost<270, Time < 10, Latency<100, Availability>0,4Reliability>0,2, Successability>0,8, Throughput>1, Documentation>0,001

Pour chaque paramètre QOS on décompose la contrainte globale imposée en contrainte locale en utilisant les équations qu'on a vues précédemment. Pour plus de clarification on présentera un exemple sur les paramètres QOS coût et disponibilité.

- **Exemple sur le paramètre négatif coût :**

Tout d'abord, nous calculons les valeurs $V\_Qos(C)_{[i][j]}$ en utilisant les **équations (1)** et **(2)**. Ensuite, nous obtenons la moyenne$AVG_VQos(C)_{[i]}$pour chaque sous-tâche à l'aide de **l'équation (3)**. Ensuite, nous calculons la valeur VC(C) en les comparants à la valeur MaxN_C(C)selon **l'équation (4) et (5)**. Par la suite, nous déterminons la valeur VN_QOSC(C) pour chaque contrainte QOS en utilisant **l'équation (6)**. La valeur de la contrainte locale $LC(C)_{[i]}$ de chaque sous-tâche est calculée à l'aide de **l'équation (7)**.

Ensuite, nous appliquons **l'équation (15)** pour obtenir la valeur DIFF(C). Enfin, nous déterminons quel CS sera sélectionné pour la composition. Les calculs des contraintes locales pour le paramètre "Coût" sont définis dans le tableau suivant :

cost										
sous-tache	rvices candida	cost	Vcost	AVG	Vvcost	LC	Diff	Best		
s1	CS11	93,76	0,60490323	0,68230108	0,23466966	93,851438	-0,09143796	1	contrainte	270
	CS12	132,94	0,85767742				39,088562	0	Max	155
	CS13	90,57	0,58432258				-3,28143796	1	MaxQC	399,93
s2	CS21	77,18	0,49793548	0,56182796	0,19323431	77,2801972	-0,10019721	1	vc	0,67511815
	CS22	72,08	0,46503226				-5,20019721	1	sommes des moyennes (AVG)	1,96290323
	CS23	111,99	0,72251613				34,7098028	0		
s3	CS31	87,63	0,56535484	0,71877419	0,24721417	98,8683648	-11,2383648	1		
	CS32	91,6	0,59096774				-7,26836483	1	sommes des LC	270
	CS33	155	1				56,1316352	0		

Tableau 4:les résultats de l'application de l'algorithme de sélection du paramètre QOS "Cost"

- **Exemple sur le paramètre positif disponibilité :**

Tout d'abord, nous calculons les valeurs V_Qos(A)$_{[i][j]}$ en utilisant les **équations (8)** et **(9)**. Ensuite, nous obtenons la moyenne $AVG_VQos(A)_{[i]}$pour chaque sous-tâche à l'aide de **l'équation (10)**. Ensuite, nous calculons la valeur VC(A) en les comparants à la valeur MaxN_C(A)selon **l'équation (11) et (12)** Par la suite, nous déterminons la valeur VP_QOSC(A) pour chaque contrainte QOS en utilisant **l'équation (13)**. La valeur de la contrainte locale LC(A)$_{[i]}$ de chaque sous-tâche est calculée à l'aide de **l'équation (14)**. Ensuite, nous appliquons **l'équation (15)** pour obtenir la valeur DIFF(A). Enfin, nous déterminons quel CS sera sélectionné pour la composition.

Les calculs des contraintes locales pour le paramètre "disponibilité" sont définis dans le tableau suivant :

Availability										
									contrainte	0,4
subtasks	rvices candida	avaliability	Vavaliability	AVGAVB	EqF	LC	Diff	Best	Max	0,98
s1	CS11	0,89	0,90816327	0,89455782	0,51706478	0,73309442	0,15690558	1	MaxQC	2,85
	CS12	0,85	0,86734694				0,11690558	1	vc	0,14035088
	CS13	0,89	0,90816327				0,15690558	1	produits AVG	0,72678425
s2	CS21	0,98	1	0,90136054	0,52099684	0,73866928	0,24133072	1		
	CS22	0,87	0,8877551				0,13133072	1		
	CS23	0,8	0,81632653				0,06133072	1	la raçine	0,57801158
s3	CS31	0,98	1	0,90136054	0,52099684	0,73866928	0,24133072	1		
	CS32	0,76	0,7755102				0,02133072	1	produit des LC	0,4
	CS33	0,91	0,92857143				0,17133072	1		

Tableau 5: les résultats de l'application de l'algorithme de sélection du paramètre QOS "Availability"

Pour choisir la meilleure composition de services pour l'exécution de la demande du client, nous allons sélectionner le CS[i][j] dans chaque sous-tâche qui satisfait toutes les contraintes locales de chaque paramètre QOS.

CS	Cost	Rst	Response Time	Rst	Latency	Rst	Reliability	Rst	Availability	Rst	Throughput	Rst	Successability	Rst	Documentation	Rst	Result	Best	Ranking
CS 0.0	93.76	1	302.75	1	187.75	0	0.73	1	0.89	1	7.1	1	0.9	1	0.32	1	0	0	2
CS 0.1	132.94	0	482.0	1	1.0	1	0.73	1	0.85	1	16.0	1	0.95	1	0.02	0	0	0	3
CS 0.2	90.57	1	3321.4	1	2.6	1	0.73	1	0.89	1	1.4	1	0.96	1	0.96	1	1	1	1
CS 1.0	77.18	1	126.17	1	22.77	1	0.67	1	0.98	1	12.0	1	1.0	1	0.89	1	1	1	1
CS 1.1	72.08	1	107.0	1	58.33	0	0.73	1	0.87	1	1.9	1	0.95	1	0.93	1	0	0	2
CS 1.2	111.99	0	107.57	1	18.21	1	0.67	1	0.8	1	1.7	1	0.81	1	0.61	1	0	0	3
CS 2.0	87.63	1	255.0	1	40.8	0	0.67	1	0.98	1	1.3	1	0.99	1	0.04	0	0	0	3
CS 2.1	91.6	1	136.71	1	11.57	1	0.6	1	0.76	1	2.8	1	0.76	1	0.08	1	1	1	1
CS 2.2	155.0	0	102.62	1	0.93	1	0.67	1	0.91	1	15.3	1	0.97	1	0.91	1	0	0	2

Best Composition:

CS	Cost	Response Time	Latency	Reliability	Availability	Throughput	Successability	Documentation
CS 0.2	90.57	3321.4	2.6	0.73	0.89	1.4	0.96	0.96
CS 1.0	77.18	126.17	22.77	0.67	0.98	12.0	1.0	0.89
CS 2.1	91.6	136.71	11.57	0.6	0.76	2.8	0.76	0.08

Tableau 6:Tableau de notre premier algorithme de la sélectionné la meilleure composition du service

6 L’algorithme de priorité :

Ce procédé algorithmique revêt une importance cruciale dans la sélection de services CMfg qui satisfait un ensemble de contraintes QOS imposées par les clients. Son objectif principal est consisté à calculer la priorité de chaque contrainte temporelle des services candidats dans chaque sous tache.

Tout d'abord, on présenter l'ensemble des contraintes QOS en utilisant le vecteur TC = {TC_1,…., TC_t,...., TC_T } ,Ces derniers sont imposés dans une composition avec diverses priorités exprimées dans deux vecteur de priorité, un vecteur de priorité des paramètres négatives PrN[t] = { TC_C, TC_T, TC_L },Ou TC_c doit avoir la plus haute priorité et TCL doit avoir la priorité la plus basse, et un vecteur de priorité des paramètres positives PrN[t] = { TCA, TC_R, TC_S,TC_T,TC_D}, Ou TCA doit avoir la plus haute priorité et TCD doit avoir la priorité la plus basse. Chaque contrainte QOS a un vecteur d'affectation VA et une constante Const pour exprimer la contrainte, qui est définie en utilisant les équations suivantes :

- **Pour les paramètres négatifs :**

$$TC_C = \begin{cases} AV_C[\mathrm{i}], \forall \mathrm{i} \in \{1, \mathrm{n}\} \\ \leq \text{ Const C} \end{cases} \quad (16.1)$$

Où $AV_C(i) = \{ST_1, \dots, ST_i, \dots, ST_n\}$ /

$$\begin{cases} ST_i = 1, \text{Si } ST_i \text{ est inclus dans cette contrainte} \\ ST_i = 0, \text{sinon} \end{cases} \quad (16.2)$$

$$TC_T = \begin{cases} AV_T[\mathrm{i}], \forall \mathrm{i} \in \{1, \mathrm{n}\} \\ \leq \text{ Const T} \end{cases} \quad (17.1)$$

Où $AV_T(i) = \{ST_1, \dots, ST_i, \dots, ST_n\}$ /

$$\begin{cases} ST_i = 1, \text{Si } ST_i \text{ est inclus dans cette contrainte} \\ ST_i = 0, \text{sinon} \end{cases} \quad (17.2)$$

$$TC_L = \begin{cases} AV_L[\mathrm{i}], \forall \mathrm{i} \in \{1, \mathrm{n}\} \\ \leq \text{ Const L} \end{cases} \quad (18.1)$$

Où $AV_L(i) = \{ST_1, \dots, ST_i, \dots, ST_n\}$ / $\begin{cases} ST_i = 1, \text{Si } ST_i \text{ est inclus dans cette contrainte} \\ ST_i = 0, \text{sinon} \end{cases}$

(18.2)

- **Pour les paramètres positifs :**

$$TC_A = \begin{cases} AV_A[\text{i}], \forall \text{i} \in \{1, \text{n}\} \\ \geq \text{Const A} \end{cases} \quad (19.1)$$

Où $AV_A(i) = \{ST_1, \dots, ST_i, \dots, ST_n\}$ /

$\begin{cases} ST_i = 1, \text{Si } ST_i \text{ est inclus dans cette contrainte} \\ ST_i = 0, \text{sinon} \end{cases}$ (19.2)

$$TC_R = \begin{cases} AV_R[\text{i}], \forall \text{i} \in \{1, \text{n}\} \\ \geq \text{Const R} \end{cases} \quad (20.1)$$

Où $AV_R(i) =$

$\{ST_1, \dots, ST_i, \dots, ST_n\} - \begin{cases} ST_i = 1, \text{Si } ST_i \text{ est inclus dans cette contrainte} \\ ST_i = 0, \text{sinon} \end{cases}$ (20.2)

$$TC_S = \begin{cases} AV_S[\text{i}], \forall \text{i} \in \{1, \text{n}\} \\ \geq \text{Const S} \end{cases} \quad (21.1)$$

Où $AV_S(i) =$

$\{ST_1, \dots, ST_i, \dots, ST_n\} - \begin{cases} ST_i = 1, \text{Si } ST_i \text{ est inclus dans cette contrainte} \\ ST_i = 0, \text{sinon} \end{cases}$ (21.2)

$$TC_{Th} = \begin{cases} AV_{Th}[\text{i}], \forall \text{i} \in \{1, \text{n}\} \\ \geq \text{Const Th} \end{cases} \quad (22.1)$$

Où $AV_{Th}(i) = \{ST_1, \dots, ST_i, \dots, ST_n\}$ / $\begin{cases} ST_i = 1, \text{Si } ST_i \text{ est inclus dans cette contrainte} \\ ST_i = 0, \text{sinon} \end{cases}$ (22.2)

$$TC_D = \begin{cases} AV_D[\text{i}], \forall \text{i} \in \{1, \text{n}\} \\ \geq \text{Const D} \end{cases} \quad (23.1)$$

Où $AV_D(i) = \{ST_1, \dots, ST_i, \dots, ST_n\}$ /

$\begin{cases} ST_i = 1, \text{Si } ST_i \text{ est inclus dans cette contrainte} \\ ST_i = 0, \text{sinon} \end{cases}$ (23.2)

6.1 Pseudo code de l'algorithme de priorité avec des contraintes QOS négative

Negative algorithm parameters

Set of sub tasks STi = {ST1,… ST1-n ,STn}

Set of t QOS constraints TC={TC[0], TC[1], TC[02}

Set of affectation vectors AV={AV_c[i], AV_T[i], AV_L[i]} AV = boolean type

Set of priority vectors Ptc= {ptn_0[i], ptn_1[i], ptn_2[i]}

Begin

```
For(i = 0 → n)
        Sn[i]= AVc[i]+AVT[i]+AVL[i]
End
For(i = 0 → n)
    For(j = 0 → 3)
     If( PTN[i][j]<maxN[i]) then
       maxN[i]=PTN[i][j]          , j={C, T, L}
    End
End
For(i = 0 → n)
    For(j = 0 → 3)
     if (PTN[i][j]!=0) then
        aa=maxN[i]-PTN[i][j]
        ax=Math.pow( 2,aa)
        sumN=sumN+(1/ax)
     End
  pN[i]=Sn[i]/sumN    //calculer de pn
End
For(i = 0 → n)
    For(j = 0 → 3)
      ptn[i][j]=(pN[i]/Math.pow( 2,maxN[i]-1))*AV(N)[i][j]  , N={C, T, L}
    End
End
End
```

6.2 Pseudo code de l'algorithme de priorité avec des contraintes QOS positives

```
Set of sub tasks STi = {ST1,… ST1-n ,STn}
Set of QOS constraints  TC={TC[3], …,TC[7]}
Set of affectation vectors  AV={ AVA[i], AVR[i], AVS[i], AVTh[i], AVD[i]}, AV = boolean type
Set of priority vectors  Ptc= {ptp0[i], …, ptp4[i]}
Begin
For(i = 0 → n)
    Sp(i)= AVA[i]+AVR[i]+AVS[i]+AVTh[i]+AVD[i]
End
For(i = 0 → n)
    For(j = 0 → 5)
      If(PTP[i][j]> maxP[i] ) then
        maxP[i]=PTP [i][j]      , j= {A,  R , S , Th, D}
    End
End
For(i = 0 → n)
      For(j = 0 → 5)
      If(PTP[i][j]!=0) then
       sumP=sumP+(PTP[i][j]-1) ;
       End
       prod[i]=SIP[i]*(Math.pow(2,sumP));
       div[i]=1/ prod[i] ;
       pP[i]=Math.pow(prod[i],div[i]);
      End
End
For(i = 0 → n)
    For(j = 0 → 5)
        ptp[i][j]=(pP[i]/Math.pow( 2,maxN[i]-1))*AV(P)[i][j] , P={A,  R , S , Th, D}
    End
End
End
```

L'algorithme d'attribution de priorité reçoit en entrée un nombre de sous-tâches ST, un ensemble de sous-tâches ST, un ensemble de contraintes TC, un vecteur d'affectation pour chaque contrainte QOS, ainsi qu'un vecteur contenant les priorités des contraintes QOS pour chaque sous-tâche.

La détermination de la variable de la priorité maximale Pn de paramètre négatives, et la priorité maximale Pp de paramètre positives pour chaque sous-tâche (i) nécessite l'accomplissement d'une série d'étapes successives.

Tout d'abord, en se basant sur les vecteurs d'affectation, il est nécessaire d'attribuer les contraintes TC aux sous-tâches ST dans une matrice d'allocation. Cela signifie qu'à la fin de chaque ligne de la matrice, il doit exister une case Sn(i) contenant un nombre représentant la somme des contraintes négatives TC, et une case Sp(i) contenant un nombre représentant la somme des contraintes positives TC attribuées à la sous-tâche (i).

Ensuite, en utilisant le vecteur de priorité des TC, il faut rechercher la contrainte ayant la priorité la plus élevée dans chaque sous-tâche (i).

Puis, pour chaque sous-tâche (i), Les variable Pn Sont déterminant en effectuant la division de la variable Sn(i) par la variable sumN, et Les variable Pp Sont déterminant en effectuant la puissance de prod[i] de div[i].

Enfin, en prenant en compte la distance entre chaque contrainte QOS (TC), la priorité de chaque contrainte QOS négatives (TC) pour la sous-tâche (i) est calculée en divisant la valeur Pn par 2 élevé à la puissance de la priorité de la contrainte moins un, et la priorité de chaque contrainte QOS négatives (TC) pour la sous-tâche (i) est calculée en divisant la valeur Pp par 2 élevé à la puissance de la priorité de la contrainte moins un.

6.3 Exemple :

Le tableau montre un exemple numérique, qui contient cinq contraintes QOS positives avec un vecteur de classement des priorités pour chaque sous-tache selon la demande de client.

	AAV	ARV	ASV	AThV	ADV	PA	PR	PS	PTh	PD	SI
ST1	1	1	0	1	0	1	2	0	3	0	3
ST2	1	0	1	1	1	2	0	3	4	1	4
ST3	0	1	1	0	0	0	2	1	0	0	2

***T*ableau 7:Vecteurs d'allocation et de priorité des contraintes QOS positive affectées aux différents sous-taches.**

En utilisant l'algorithme de priorité pour les paramètres positives(Availability, Reliability, Successability, Throughput, et Documentiont), le degré de priorité de chaque contrainte temporelle pour chaque sous-tâche est calculé dans le tableau

	AAV	ARV	ASV	AThV	ADV	PA	PR	PS	PTh	PD	SI
ST1	1	1	0	1	0	1,71	0,85	0	0,42	0	3
ST2	1	0	1	1	1	1,06	0	0,53	0,26	2,13	4
ST3	0	1	1	0	0	0	0,66	1,33	0	0	2

Tableau 8 :Vecteurs de degré de priorité des contraintes QOS positive affectées aux différents sous-taches.

Le tableau montre un exemple numérique, qui contient trois contraintes QOS positives avec un vecteur de classement des priorités pour chaque sous-tache selon la demande de client.

	ACV	ATV	ALV	Pcost	Ptime	Platency	Si
ST1	1	1	1	1	2	3	3
ST2	0	1	1	0	1	2	2
ST3	1	0	1	2	0	1	2

Tableau 9:Vecteurs d'allocation et de priorité des contraintes QOS négative affectées aux différents sous-taches.

En utilisant l'algorithme de priorité pour les paramètres négatives (Cost, Time, Latency), et on suppose que on a trois sous-tache, le degré de priorité de chaque contrainte QOS pour chaque sous-tâche est calculé dans le tableau

	ACV	ATV	ALV	Pcost	Ptime	Platency	Si
ST1	1	1	1	1,71	0,85	0,42	3
ST2	0	1	1	0	1,33	0,66	2
ST3	1	0	1	0,66	0	1,33	2

Tableau 10 :Vecteurs de degré de priorité des contraintesQOS négative affectées aux différents sous-taches.

7 Approche globale augmenter de contraint QoS avec des priorités(AGACQP) :

Notre approche combine la recherche exhaustive de toutes les combinaisons possibles de compositions de services avec l'algorithme de priorité. Cet algorithme assigne une valeur de priorité à chaque paramètre de qualité de service qu'est affectée à chaque sous-tâche en fonction de leur importance respective. Cela permet de prendre des décisions éclairées pour sélectionner la composition de services qui répondra le mieux aux exigences spécifiques du client. En combinant la perspective globale de l'approche exhaustive avec l'évaluation prioritaire des paramètres QoS, cette approche vise à atteindre un équilibre optimal entre les performances globales et les critères de priorité dans le processus de composition des services.

Pour associer les services les plus pertinents à chaque sous tâche, nous appelons d'abord l'algorithme de priorité. Puis, en utilisant l'algorithme de priorité, la priorité globale de chaque contrains pour tous les sous-tâches est calculée, par **l'équation (24)**

$$pt(f) = \frac{\sum_{i=0}^{n} pt(f) + pt[i](f)}{n} \textbf{(24)}$$

Par la suite, **l'équation (25)** calcule les valeurs qui représente les contraintes globales multiplie par leurs priorités globales.

$$QoS\_P(f) = \prod_{f=0}^{8} QoS(f) * pt(f) \textbf{(25)}$$

Ensuite, nous calculons la somme des valeurs Cost, Time, Latency et Throughput multiplie par leur priorité de sous-tâches(i),en utilisant **l'équation(26)**

$$sum(f)[r] = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{m=0}^{n}((QoS(f)[k][i] * pt[k][f]) + (QoS(f)[k+1][j] * pt[k+1][f]) + (QoS(f)[k+2][m] * pt[k+2][f])) \ , r \in [0, s^{st}] \quad (\mathbf{26.1})$$

$$\mathbf{O}\grave{u} s = nombre\ des\ services\ condidats\ et\ st = nombre\ des\ sous - t\hat{a}ches \quad (\mathbf{26.2})$$

Au même temps, **l'équation (27)**permet de calculer le produit des valeurs Availability, Reliability, Successability et Documentati on multiplie par leur priorité de sous-tâches(i).

$$pro(f)[r] = \prod_{i=0}^{n}\prod_{j=0}^{n}\prod_{m=0}^{n}\big((QoS(f)[k][i] * pt[k][f]) * (QoS(f)[k+1][j] * pt[k+1][f]) * (QoS(f)[k+2][m] * pt[k+2][f])\big), r \in [0, s^{st}] \ (\mathbf{27.1})$$

$$\mathbf{O}\grave{u} s = nombre\ des\ services\ condidats\ et\ st = nombre\ des\ sous - t\hat{a}ches \quad (\mathbf{27.2})$$

Enfin, après avoir calculé les sommes et produits des valeurs de paramètres QoS de chaque sous tâche, il faut sélectionner les compositions les plus optimale pour tous les paramètres QoS.

Dans le cas des QoS négatives, si la somme de valeur QoS négative est plus petite que la contrainte globale avec priorité, donc la composition est sélectionnée.

$$\text{Si } (sum(f)[r] < QoS\_P(f))\ alors\ opt(f)[r] = 1$$

$sinon\ opt(f)[r] = 0$ Dans le cas des QoS positives, si la somme et le produit de valeur QoS positive est plus grande que la contrainte globale avec priorité, donc la composition est sélectionnée.

- $\text{Si } (sum(f)[r] > QoS\_P(f))\ alors\ opt(f)[r] = 1$

  $sinon\ opt(f)[r] = 0$
- $\text{Si } (pro(f)[r] > QoS\_P(f))\ alors\ opt(f)[r] = 1$

  $sinon\ opt(f)[r] = 0$

Finalement, pour l'obtention de la meilleure composition de service on calculera la valeur Best[r], en faisant un ET logique entreopt(f)[r] de tous les paramètres QoS.

Si (opt(C)[r] &opt(T)[r] &opt(L)[r] &opt(A)[r] &opt(R)[r] &opt(S)[r] &opt(Th)[r] &opt(D)[r]) = 1

Dans Le cas où on a plusieurs optimale composition, on sera obligé de choisir une composition parmi tous les optimales compositions, en utilisant **l'équations (28,29,30,31)** :

$$Neg[r] = \sum_{i=0}^{r} sum(C)[r] + sum(T)[r] + sum(L)[r] \ (\mathbf{28})$$

$$Pos[r] = \sum_{i=0}^{r} pro(A)[r] + pro(R)[r] + pro(S)[r] + sum(Th)[r] + pro(D)[r] (\mathbf{29})$$

$$neg = QoS\_P(C) + QoS\_P(T) + QoS\_P(L) (\mathbf{30})$$

$$pos = QoS\_P(A) + QoS\_P(R) + QoS\_P(S) + QoS\_P(Th) + QoS\_P(D) \qquad (\mathbf{31})$$

En conclusion, si la composition a le min Neg[r] tel que (Neg[r] <neg), et le max Pos[r] tel que (Pos[r] > pos), alors cette composition sera sélectionnée comme meilleure composition.

7.1 Pseudo code d'approche globale prioritaire avec des contraintes QOS

Algorithm parameters

Set of sub tasks STi = {ST1,…, ST1-n ,STn}

Set of candidate service Cij = {$CS_{1.1}$, $CS_{2.1}$, …., CS_{nm}}

Set of constraints C = {C1, C2, C3, C4, C5, C6, C7, C8}

Set of global constraints GC={ GC [0], …,GC [2], GC [3],…, GC [7]}

Set of affectation vectors AV={ AV_c[i], AV_T[i], AV_L[i], AV_A[i], AV_R[i], AV_S[i], AV_{Th}[i], AV_D[i]}, AV = boolean type

Set of priority vectors Ptc= { ptn_0[i],…, ptn_2[i]ptp_0[i], …, ptp_4[i]}

Begin

For$(i = 0 \rightarrow n)$

Call the priority algorithm

End

Int r=(int)Math.pow(m, n)

For$(i = 0 \rightarrow r)$

Compute sum(f)[r] and pro(f)[r] using eq.26 and eq.27

End

For$(i = 0 \rightarrow r)$

Compute sum(f)[r] and pro(f)[r] using eq.26 and eq.27

End

For$(i = 0 \rightarrow r)$

If(sum(f)[r]<QoS_P(f)) then opt(f)[r]=1 **Else** opt(f)[r]=0 ,f={C, T, L}

If(sum(Th)[r]>QoS_P(Th)) then opt(Th)[r]=1 **Else** opt(Th)[r]=0

If(pro(z)[r]>QoS_P(z)) then opt(z)[r]=1 **Else** opt(z)[r]=0 ,z={A, R, S, D}

End

For$(i = 0 \rightarrow r)$

If (opt(C)[r]=1 & opt(T)[r]=1 & opt(L)[r]=1 & opt(A)[r]=1 & opt(R)[r] =1& opt(S)[r]=1 & opt(Th)[r]=1 & opt(D)[r]=1) **then**

Best[r] = 1

End

End

En d'autres termes, l'algorithme prend en entrée un ensemble de contraintes QoS globales négatives et positives (temps de réponse, coût, latence, disponibilité, débit, succès, fiabilité, documentation) pour chaque service candidat, un ensemble de contraintes QoS, un ensemble de vecteurs d'affectation et un ensemble de vecteurs de priorité. Après avoir calculé les priorités attribuées à chaque sous-tâche à l'aide de l'algorithme de priorité.

L'algorithme AGACQP calcule la somme et le produit des valeurs QoS négatives et positives en respectant les priorités des contraintes de chaque sous-tâche.

L'algorithme comparé la somme et le produit des valeurs QoS para pour les contraintes QoS globales avec priorité. Si la somme des valeurs QoS négatives est inférieure à la contrainte globale négative avec priorité, alors la composition de contraintes est sélectionnée. De même, si la somme et le produit des valeurs QoS positives sont supérieurs à la contrainte globale positive avec priorité, alors la composition de contraintes est sélectionnée.

L'algorithme vérifié si tous les compositions des services sélectionné dans chaque sous-tâche pour chaque contraints satisfont les contraintes QoS globales avec priorité.

8 Algorithme de sélection des services CMfg augmenter de contraint QoS avec des priorités :

En combiner l'algorithme de sélection des services CMfg a des contraintes QoS prioritaire et l'Algorithme de priorités, nous proposons une nouvelle approche pour la sélection du service CMfg, qui implique un algorithme capable de classifier tous les services candidats et de trouver une combinaison qui se rapproche de l'optimum en termes de services CMfg en respectant la priorité de chaque contrainte pour chaque sous tache.

Tous d'abord, en utilisant l'algorithme de priorité, la priorité globale de chaque contrains pour tous les sous-tâches est calculée, par **l'équation (32)**

$$pt(N) = \frac{\sum_{i=0}^{n} pt(N) + pt[i](N)}{n} (\mathbf{32})$$

- **les paramètres négatifs (temps , coût et latence) :**

Il faut calculer les valeurs V_Qos(N)$_{[i][j]}$, qui représente la valuation des paramètres Qos(N)$_{[i][j]}$ multiplier par la priorité de sous-tache(i), minimisées on les comparants à la valeur maximal du paramètre Qos(N)multiplier par la priorité de sous-tache(i):

$$V_{Qos(N)_{[i][j]}} = \frac{Qos(N)_{[i][j]} * ptn[i][N]}{MaxQos(N)} \quad \forall\, i \in \{1, n\}, \forall\, j \in \{1, ci\}, \quad \forall\, N \in \{cost, time, latency\} (\mathbf{33.1})$$

$$\mathbf{Où} MaxQos(N) = Max\left[Qos(N)_{[i][j]} * ptn[i][N]\right] (\mathbf{33.2})$$

Ensuite, On veut calculer la moyenne de chaque sous tâche, en utilisant les valeur V_Qos(N)$_{[i][j]}$, pour chaque sous tâche.

$$AVG\_VQos(N)_{[i]} = \frac{\sum_{j=1}^{ci} V\_Qos(N)_{[i][j]}}{ci} \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \forall \in N\,\{cost, time, \ latency\} (\mathbf{34})$$

Puis, Dans **l'équation (35)** il faut calculer la valeur VC(N) qui représente la valuation des contraintes QOS multiplier par la prioritéglobaleen les comparants à la valeur Max_C(N), qui représente la somme de la valeur maximale du paramètre QOSmultiplier par la priorité de chaque sous tâche.

$$VC(N) = \frac{Constraint(N) * pt(N)}{Max\_C(N)} \quad \forall\, N \in \{time, cost, latency\} (\mathbf{35.1})$$

Où $Max_{C(N)} = \sum_{i=1}^{n} \left(Qos(N)_{[i][j]} * ptn[i][N]\right) \forall\, i \in \{1,n\}, \forall\, j\, \{1,ci\}, \forall\, N\, \{cost, time, latency\} (\mathbf{35.2})$

Ensuite, pour exprimer la valuation des paramètres QOS augmenté de contraintes QOS avec des priorités, il faut calculer VN_QOSC(N) pour chaque contrainte QOS multiplier par la priorité en utilisant **l'équation (36)** :

$$VN\_QosC(N)_{[i]} = \frac{VC(k) * AVG\_VQos(N)_{[i]}}{\sum_{i=1}^{n} AVG\_VQos(N)_{[i]}} \quad \forall\, i \in \{1,n\}, \forall\, N\, \{cost, time, latency\} (\mathbf{36})$$

Enfin dans l'équation (37), en multipliant chaque valeur de VN_QosC(N)$_{[i]}$ par la valeur de MaxN_C pour calculer la valeur de contrainte locale avec priorité de chaque sous tâche

LC (N)$_{[i]}$ = $V\_QosC(N)_{[i]}$ * MaxN_C (N) **(37)**

- **Les paramètres positifs (disponibilité, fiabilité, succès, débit, documentation) :**

En premier lieu, il est nécessaire de procéder au calcul des valeurs $V\_Qos(P)_{[i][j]}$ Ces valeurs reflètent la valuation des paramètres $Qos(P)_{[i][j]}$multiplier par la priorité de sous-tache(i),minimisés lorsqu'ils sont comparés à la valeur maximale du paramètre QOS multiplier par la priorité de sous-tache(i), obtenue grâce à **l'équation (38).**

$$V\_Qos(P)_{[i][j]} = \frac{Qos(P)_{[i][j]} * ptp[i][P]}{MaxQos(P)} \quad \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \quad \forall\, P \in \{A,R,S,Th,D\} \ (\mathbf{38.1})$$

Où$MaxQos(P) = Max\left[Qos(P)_{[i][j]} * ptp[i][P]\right] (\mathbf{38.2})$

Ensuite, en utilisant les valeurs V_Qos(P)$_{[i][j]}$ pour chaque sous tâche, **l'équation (39)** permet de calculer la moyenne de chaque sous tâche .

$$AVG\_VQos(P)_{[i]} = \frac{\sum_{j=1}^{ci} V\_Qos(P)_{[i][j]}}{ci} \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \forall\, P \in \{A,R,S,Th,D\} (\mathbf{39})$$

Ensuite, la valeur VC(P) est calculée à l'aide de **l'équation (40)**, ce qui représente la valuation des contraintes QOS multiplier par la priorité globale, par rapport à la valeur MaxP_C(P). Cette dernière correspond au produit de la valeur maximale du paramètre QOS multiplier par la priorité de chaque sous-tâche.

$$VC(P) = \frac{Constraint(P) * pt(P)}{Max_{C(P)}} \forall\, P \in \{A,R,S,Th,D\} (\mathbf{40.1})$$

Où $MaxP\_C = \prod_{i=1}^{n} \left(Qos(P)_{[i][j]} * ptp[i][P]\right) \forall\, i \in \{1,n\}, \forall\, j \in \{1,ci\}, \forall\, P \in \{A,R,S,Th,D\} (\mathbf{40.2})$

Par la suite, une valeur VP_QOSC(P) est calculée pour chaque contrainte QOS multiplier par la priorité globale, afin de représenter la valuation des paramètres QOS multiplier par la priorité de sous-tache(i), avec l'incorporation des contraintes QOS avec priorité, en utilisant **l'équation (41)**.

$$VP\_QosC(N)_{[i]} = AVG_{VQos(P)_{[i]}} * \sqrt[ci]{\frac{VC(P)}{\prod_{i=1}^{n} AVG\_VQos(P)_{[i]}}} \forall\, i \in \{1,n\}, \forall\, P \in \{A,R,S,Th,D\} (\mathbf{41})$$

Finalement, le calcul de la valeur de contrainte locale avec priorité de chaque sous-tâche est réalisé à l'aide de **l'équation (42)** :

$$LC(P)_{[i]} = VP\_QosC(P)_{[j]} * \sqrt[ci]{MaxP\_C} \ (\mathbf{42})$$

Le calcul des valeurs DIFF, effectué à l'aide de **l'équation (43)**, permet de quantifier la différence des valeurs QOS avec priorité pour chaque CS par rapport à la nouvelle contrainte locale.

$$DIFF(N)_{[i][j]} = (Qos(N)_{[i][j]} * ptn[i][N]) - LC(N)_{[i]} \ \forall\, i \in [1,n], \forall\, j \in [1,Ci] (\mathbf{43.1})$$

$$DIFF(P)_{[i][j]} = (Qos(P)_{[i][j]} * ptp[i][P]) - LC(P)_{[i]} \ \forall\, i \in [1,n], \forall\, j \in [1,Ci] \ (\mathbf{43.2})$$

Ensuite, il est nécessaire de vérifier si chaque service candidat satisfait la contrainte locale en fonction de la valeur de différence (DIFF). Si le paramètre QOS est positif et que la valeur de DIFF est positive, alors la contrainte locale est satisfaite.

Si la valeur $DIFF(P)_{[i][j]}>0$ alors $ResAtt\_Qos(P)_{[i][j]} =1$

Sinon $ResAtt\_Qos(P)_{[i][j]} = 0$

De même, si le paramètre QOS est négatif et que la valeur de DIFF est négative, alors la contrainte locale est satisfaite.

Si la valeur $DIFF(N)_{[i][j]} < 0$ alors $ResAtt\_Qos(N)_{[i][j]} =1$

Sinon $ResAtt\_Qos(N)_{[i][j]} = 0$

En conclusion, si toutes les valeurs $ResAtt\_Qos(k)_{[i][j]}$ (k= {N, P}) des paramètres QOS multiplier par leur priorité sont satisfaites, alors ces services sont sélectionnés comme la meilleure composition de service. S'ils trouvent dans la même sous-tâche plusieurs CS avec $ResAtt\_Qos(k)_{[i][j]} = 1$, il est nécessaire de choisir un CS parmi eux. Ce choix sera effectué en respectant la priorité de chaque contrainte pour chaque sous-tache imposée par le client.

8.1 Pseudo code d'algorithme de la sélection de service CMfg a des contraintes QOS prioritaire :

Algorithm parameters

Set of sub tasks STi = {ST_1,… ST_{1-n} ,ST_n}

Set of candidate service Cij = {$CS_{1.1}$, $CS_{2.1}$, …….CS_{nm}}

Set of constraints C = {C1, C2, C3, C4, C5, C6, C7, C8}

Set of global constraints GC={ GC [0], …,GC [2], GC [3],…, GC [7]}

Set of affectation vectors AV={ AV_c[i], AV_T[i], AV_L[i], AV_A[i], AV_R[i], AV_S[i], AV_{Th}[i], AV_D[i]}, AV = boolean type

Set of priority vectors Ptc= { ptn_0[i],…, ptn_2[i]ptp_0[i], …, ptp_4[i]}

Begin

For($i = 0 \rightarrow n$)

Call the priority algorithm

End

For($i = 0 \rightarrow n$)

Compute $LC_{[i]}$ using eq.37 Or eq.42

End

For($i = 0 \rightarrow n$)

For($j = 0 \rightarrow Cj$)

$DIFF(N)_{[i][j]}$ = ($Qos(N)_{[i][j]}$ *ptn[i][N])- $LC_{[i]}$

$DIFF(P)_{[i][j]}$ = ($Qos(P)_{[i][j]}$ *ptn[i][P])- $LC_{[i]}$

If ($DIFF(N)_{[i][j]} < 0$) **then** $ResAtt\_Qos(N)_{[i][j]} = 1$ **else** $ResAtt\_Qos(N)_{[i][j]} = 0$

If ($DIFF(P)_{[i][j]} > 0$) **then** $ResAtt\_Qos(P)_{[i][j]} = 1$ **else** $ResAtt\_Qos(P)_{[i][j]} = 0$

End

End

For($i = 0 \rightarrow n$)

For($j = 0 \rightarrow Cj$)

If($ResAtt\_Qos(C)_{[i][j]}$ and $ResAtt\_Qos(T)_{[i][j]}$ and $ResAtt\_Qos(L)_{[i][j]}$ and $ResAtt\_Qos(A)_{[i][j]}$ and $ResAtt\_Qos(R)_{[i][j]}$ and $ResAtt\_Qos(S)_{[i][j]}$ and $ResAtt\_Qos(Th)_{[i][j]}$)] and $ResAtt\_Qos(D)_{[i][j]}$) **then**

$BestCS_{[i][j]} = 1$

End

End

End

Pour être plus précis, L'algorithme calcule la valeur DIFF, qui représente la différence entre la valeur du paramètre QOS et sa priorité dans chaque sous-tâche de chaque service candidat, et la valeur LC associée à chaque sous-tâche. Le choix de la meilleure sélection est

basé sur cette valeur DIFF. Si la valeur de la contrainte locale avec priorité est supérieure que la valeur de paramètre de QOS négatif avec ces priorités alors la valeur DIFF[i][j] < 0, donc le CS[i][j] est sélectionné, et la valeur ResAtt_Qos(k)[i][j] égale à 1. Si la valeur de LC avec priorité de sous-tache (i) est inférieur que la valeur de paramètre de QOS positif avec ces priorité alors la valeur DIFF[i][j]>0 , donc le CS[i][j] est sélectionné, et la valeur ResAtt_Qos(k)[i][j] égale à 1.

Après avoir comparé chaque valeur QOS[i][j]*ptn[i][j] et QOS[i][j]*ptp[i][j] a sa contrainte locale et vérifier sa consistance, on cherchera pour chaque ST[i] le CS[i][j] qui a respecté la LC pour tous les paramètres QOS on voyons leurs valeur de ResAtt_Qos(k)[i][j]. S'ils ont tous un 1 alors ce service sera choisi comme le meilleur CS[i][j] pour l'exécution de cette ST[i] et ainsi de suite jusqu'à ce qu'on terminera toutes les ST[i] du service demande.

9 Conclusion

Dans ce chapitre, nous avons décrit la conception de nos différentes solutions. Le premier algorithme est un algorithme de sélection des services CMfg sous différentes contraintes QoS", repose sur la décomposition des contraintes globales en contraintes locales. Son objectif est de minimiser le temps de sélection tout en garantissant une composition et une sélection proches de l'optimal, afin de choisir le meilleur service répondant aux besoins du client.

Le deuxième algorithme, nommé "Approche globale d'augmentation des contraintes QoS avec des priorités", se concentre sur la sélection de la composition la plus optimale de services, en tenant compte des priorités spécifiques du client.

De plus, le troisième algorithme, appelé "Algorithme de sélection des services CMfg avec augmentation des contraintes QoS et des priorités", repose également sur la décomposition des contraintes globales avec leurs priorités en contraintes locales. Son objectif est de minimiser le temps de sélection tout en garantissant une composition basée sur les priorités pour chaque sous-tâche, ainsi qu'une sélection proche de l'optimal, afin de choisir le meilleur service répondant aux besoins du client et en tenant compte des priorités spécifique du client.

Chapitre 4 : Réalisation et expérimentation

1 Introduction

Dans le but de démontrer les réalisations et les performances de nos solutions, ce chapitre se concentrera sur la présentation professionnelle de notre application web conçue pour résoudre les problèmes d'OSCMCS. Cette présentation s'appuiera sur les concepts abordés dans le chapitre précédent. En outre, nous mettrons en avant les outils utilisés et expliquerons le fonctionnement de notre logiciel de manière détaillée.

2 Outils et Plateformes utilisées

Dans cette section, nous allons présenter les plateformes utilisées afin de réaliser nos solutions. Notre travail est implémenté sur un ordinateur :

- Processeur Intel Core i5-8265U CPU @ 1.60GHz 1.80 GHz.
- RAM de 8 Go.
- Système d'exploitation : Windows 10, 64 bits processeur x64.

Le simulateur de sélection de la composition optimale des services CMfg sous différents contraintes QoS a été développé sur la plateforme Eclipse, en utilisant le langage Java EE, l'API Jstl, le serveur Apache Tomcat, et le dataset QWS (Quality of Web Service) [38].

2.1 Eclipse

Eclipse IDE est un environnement de développement intégré (EDI) avec un accès ouvert, extensible, universel et polyvalent. Il offre la possibilité de créer des projets de développement dans différents langages de programmation. L'EDI Eclipse est principalement développé en utilisant le langage Java, avec la bibliothèque graphique SWT d'IBM. De plus, grâce à des bibliothèques spécifiques, Java est également utilisé pour écrire des extensions. Toutes les fonctionnalités de l'EDI sont développées sous forme de plugins, ce qui permet une grande flexibilité et personnalisation. Il convient de noter que plusieurs logiciels commerciaux, tels qu'IBM, se basent sur cette plateforme libre pour développer leurs propres outils.[39]

2.2 Java EE

Java EE est une plateforme axée principalement sur les serveurs pour le développement et l'exécution d'applications distribuées, elle fournit à la fois les spécifications pour une

infrastructure de serveur d'applications et un ensemble d'API modulaires utilisables dans le développement d'applications distribuées. [40]

2.3 L'API Jstl

Abréviation de « Java Server Page Standard Tag Library », est un ensemble de tags personnalisés développé selon la spécification JSR 052. Il fournit des fonctionnalités couramment utilisées dans les JSP : Des tags de structure pour l'itération et les conditions, L'internationalisation, L'exécution de requêtes SQL et dans L'utilisation de documents XML Pour utiliser JSTL, vous avez besoin d'un conteneur d'applications web qui implémente l'API servlet 2.3 et l'API JSP 1.2. L'implémentation de référence de cette spécification, appelée JSTL-RI, est développée par le projet Taglibs du groupe Apache, sous le nom "Standard".[41]

2.4 Serveur Apache Tomcat

Apache Tomcat est un logiciel avec accès ouvert de serveur d'applications web spécialement conçu pour la programmation en Java. Il est développé et maintenu par Jakarta, le groupe de projets Java open source de la fondation Apache. L'objectif principal d'Apache Tomcat est de fournir un environnement d'hébergement et de déploiement pour les servlets Java. [42]

3 Présentation de l'interfaces des solutions proposées

Tous d'abord, nous avons décrit les interfaces de base comme suit : l'interface d'accueil Qui permet d'accéder à l'implémentation de nos deux solutions. Elle offre deux boutons :

Le premier bouton permet d'accéder à l'interface d'implémentation l'algorithme de la sélection de service CMfg a des contraintes QOS avec des priorités décroissantes et l'approche globale QOS avec des priorités décroissantes.

Ainsi que le deuxième bouton permet d'accéder à l'implémentation d'algorithme de la sélection de service CMfg a des contraintes QOS prioritaire et l'approche globale a des contraintes QOS prioritaire. Comme il est montré dans **la figure 5** :

Figure 5: Interface d'accueil

La **figure 6**, représente l'interface de notre première solution qui possède deux options, l'approche globale et la première solution

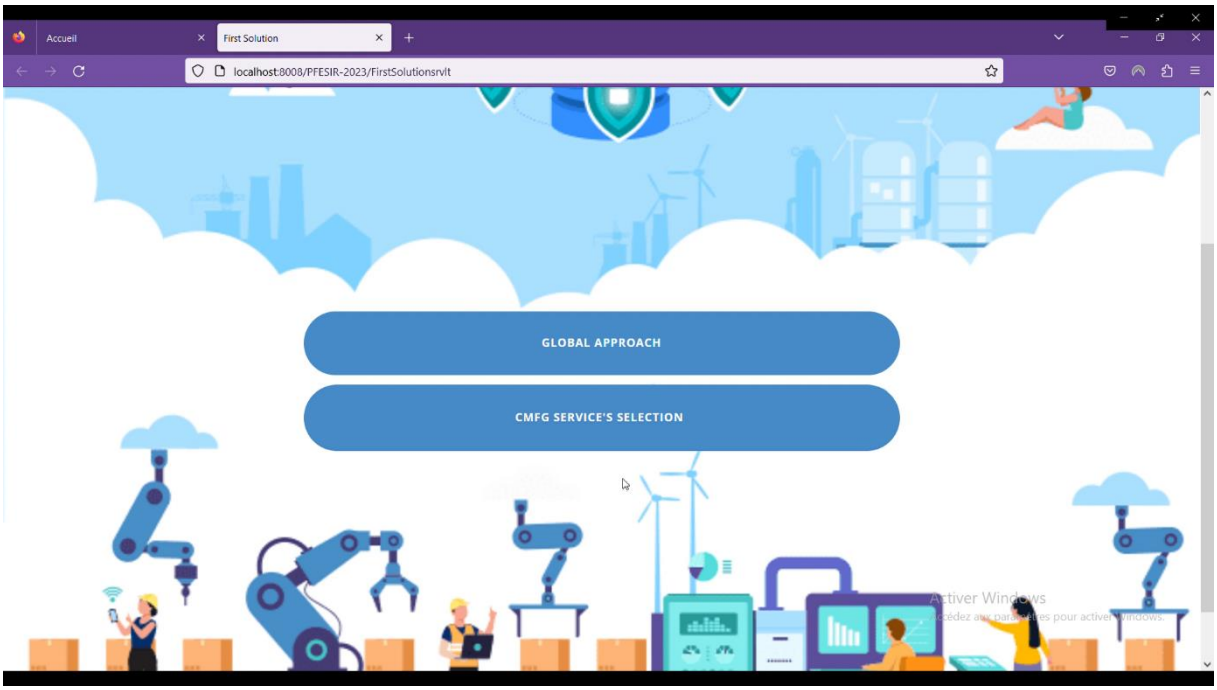


Figure 6: Interface de choix de premier solution.

Premièrement, l'utilisateur donnée les contraintes QOS globale, ainsi que le nombre de services candidats et les sous taches, Comme présenté dans **la figure 7 :**

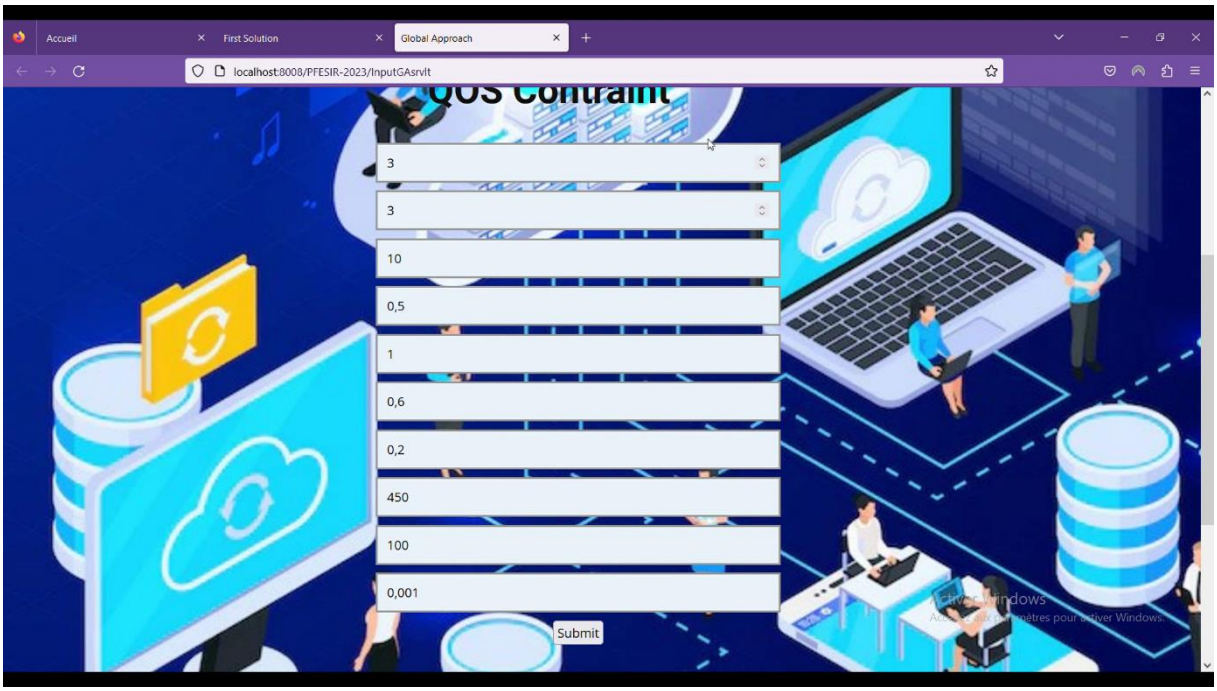
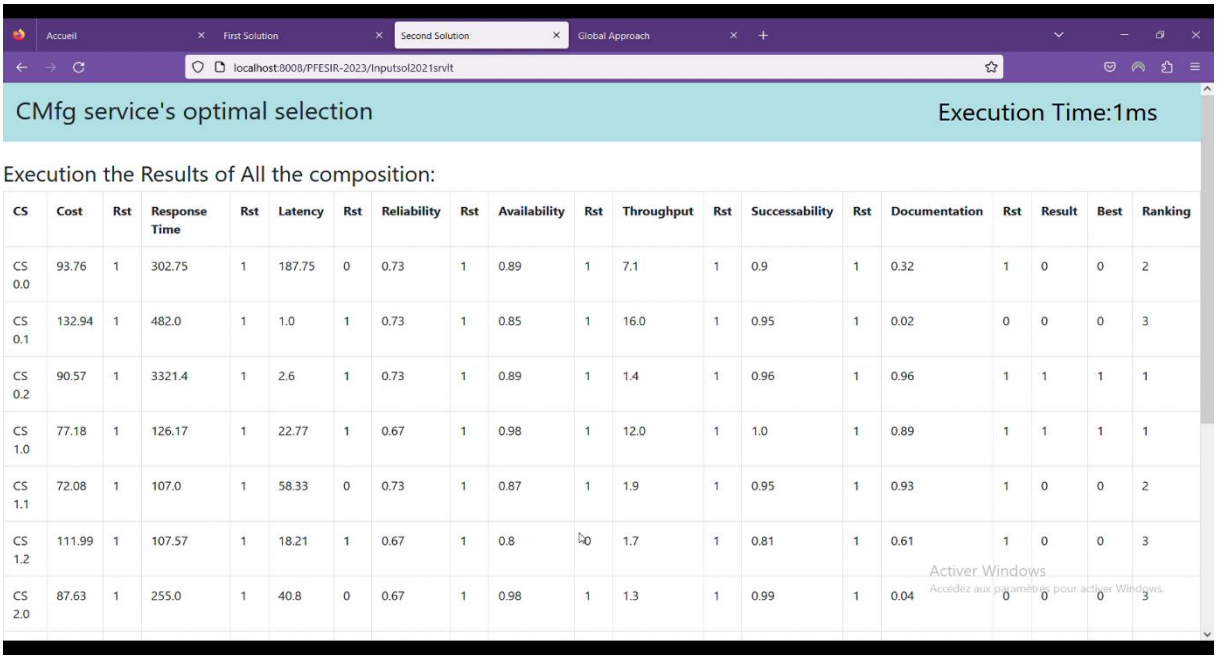


Figure 7: interface des contraintes QOS globale entrant par l'utilisateur

Ensuite ont cliquant sur le bouton « Submit ». Le vecteur statique de priorité, et toutes les équations de la première solution vues précédemment en détail dans **le chapitre 3** seront exécutés et tous les résultats seront affichés dans la liste des services candidats.



CS	Cost	Rst	Response Time	Rst	Latency	Rst	Reliability	Rst	Availability	Rst	Throughput	Rst	Successability	Rst	Documentation	Rst	Result	Best	Ranking
CS 0.0	93.76	1	302.75	1	187.75	0	0.73	1	0.89	1	7.1	1	0.9	1	0.32	1	0	0	2
CS 0.1	132.94	1	482.0	1	1.0	1	0.73	1	0.85	1	16.0	1	0.95	1	0.02	0	0	0	3
CS 0.2	90.57	1	3321.4	1	2.6	1	0.73	1	0.89	1	1.4	1	0.96	1	0.96	1	1	1	1
CS 1.0	77.18	1	126.17	1	22.77	1	0.67	1	0.98	1	12.0	1	1.0	1	0.89	1	1	1	1
CS 1.1	72.08	1	107.0	1	58.33	0	0.73	1	0.87	1	1.9	1	0.95	1	0.93	1	0	0	2
CS 1.2	111.99	1	107.57	1	18.21	1	0.67	1	0.8	0	1.7	1	0.81	1	0.61	1	0	0	3
CS 2.0	87.63	1	255.0	1	40.8	0	0.67	1	0.98	1	1.3	1	0.99	1	0.04	0	0	0	3

Figure 8:L'exécution de notre premier algorithme

Ainsi que la meilleure composition présenté dans **la figure 9**. Talque le meilleur service est choisi selon le vecteur de priorité et si tous les contraintes QOS pour chaque service soit satisfait

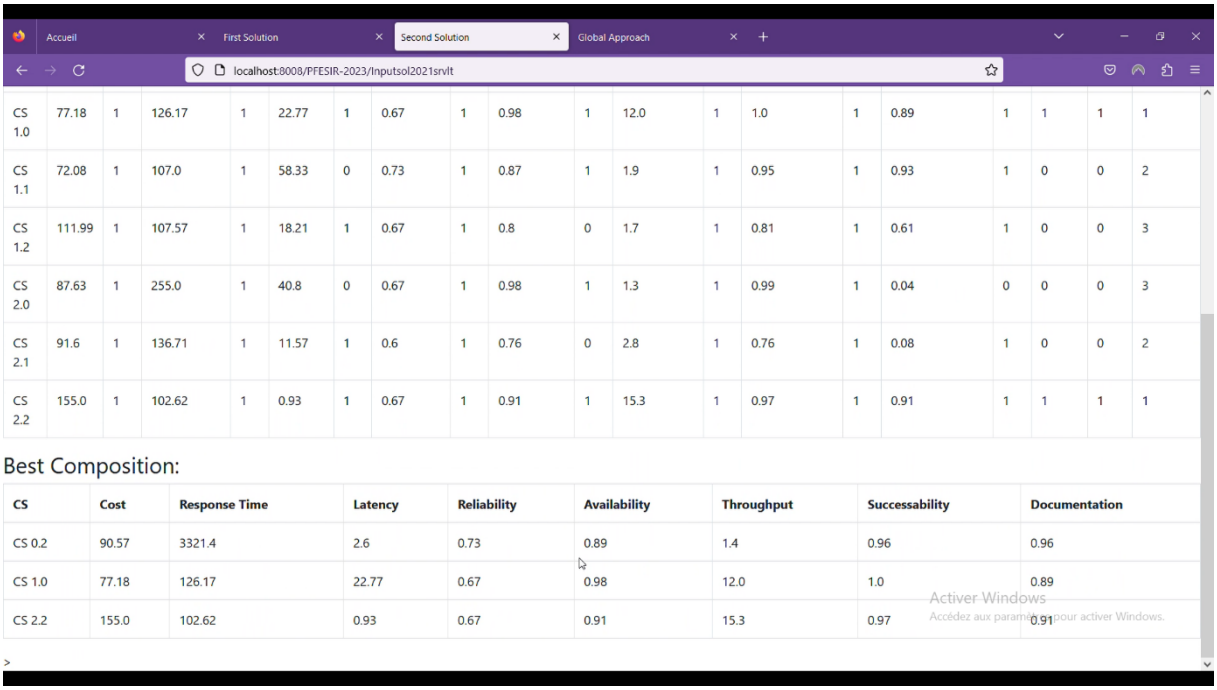


Figure 9:La sélection de la meilleure composition du service

Cette interface présente tous les meilleures compositions de l'approche globale modifiée, où nous avant utilisée le vecteur statique de priorité pour le classement des compositions

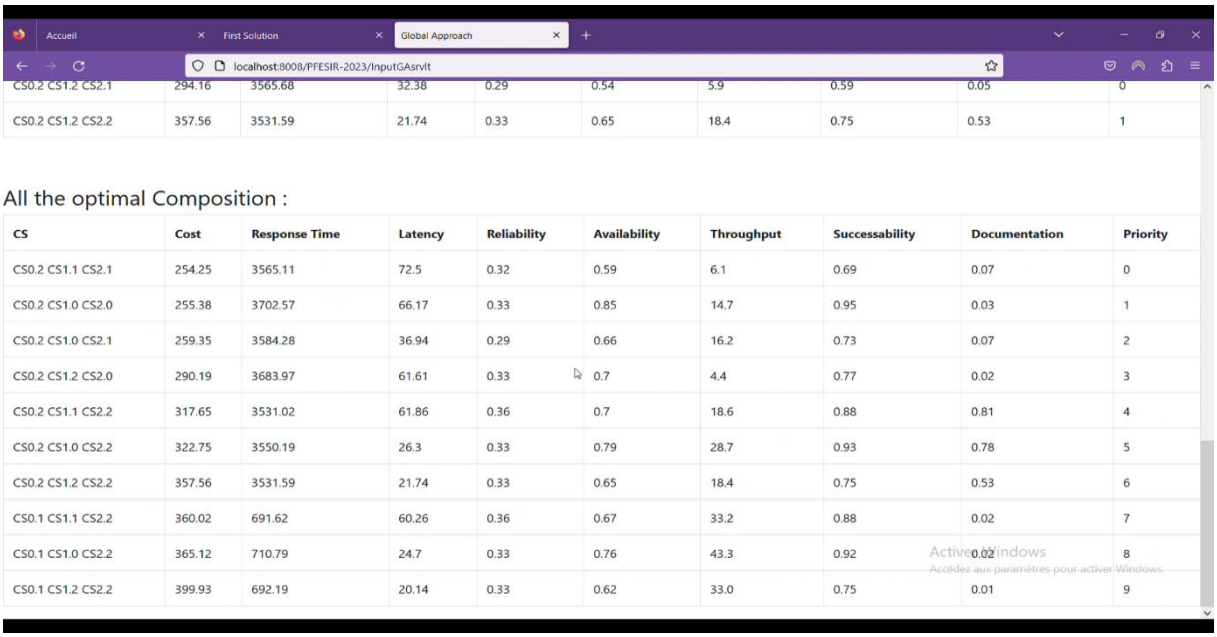


Figure 10: L'exécution d'approche globale modifié et La sélection des meilleures compositions du service.

La figure, représente l'interface de notre deuxième solution qui possède deux options, l'approche globale modifiée et la deuxième solution

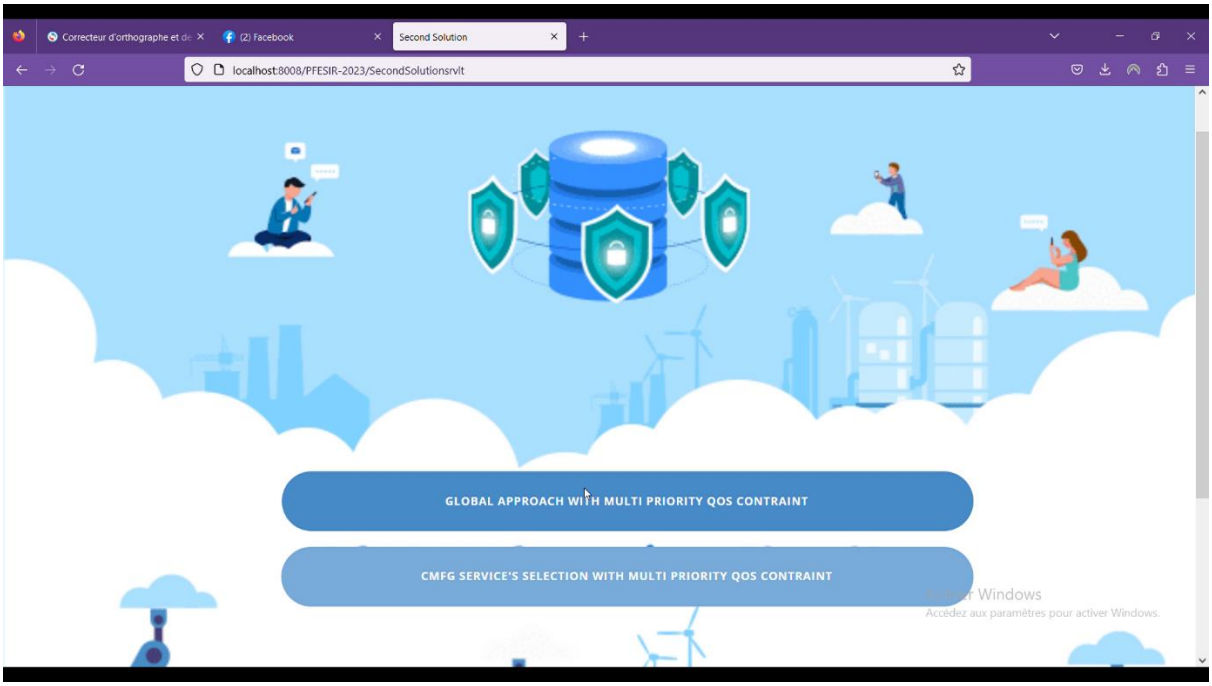


Figure 11: Interface de choix de deuxième

Cette interface représente les contraintes QOS globale, le nombre de services candidats et les sous taches, ainsi que les vecteurs d'affectations et les vecteurs de priorité pour chaque contraintes QOS. Qui est donnée par l'utilisateur dans la figure

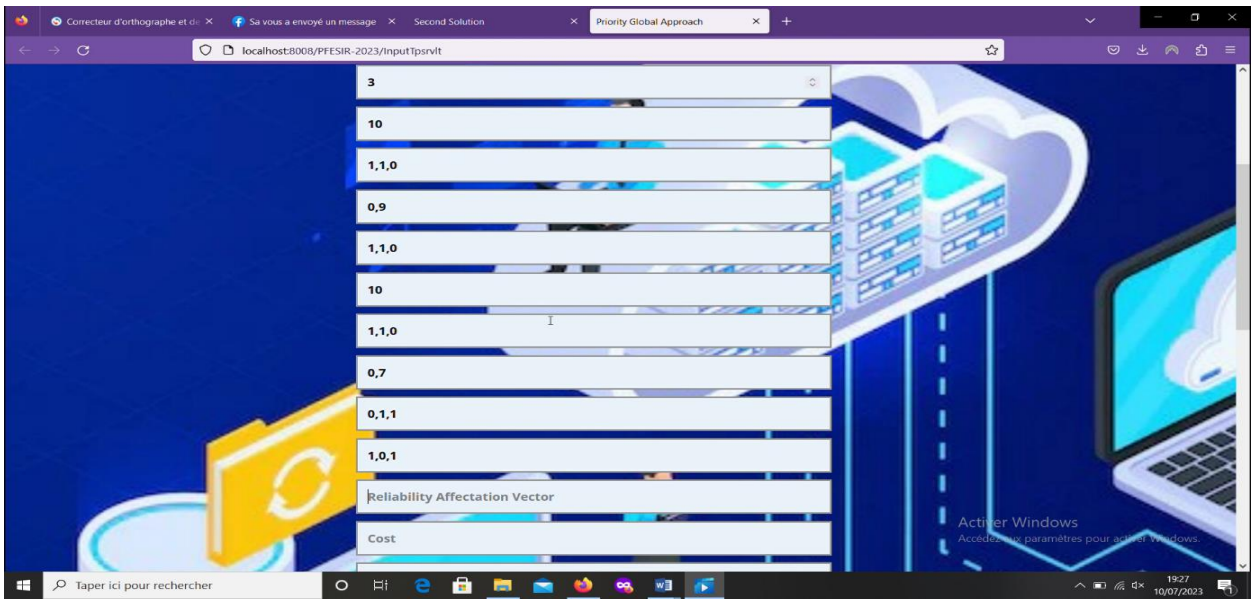


Figure 12:interface des contraintes QOS globale et les vecteurs de priorité entrant par l'utilisateur.

Le bouton radio est pour que l'utilisateur choisie si les vecteurs de priorité donnée par lui soit des priorités calculer ou pas. Comme présenté dans **la figure 13**

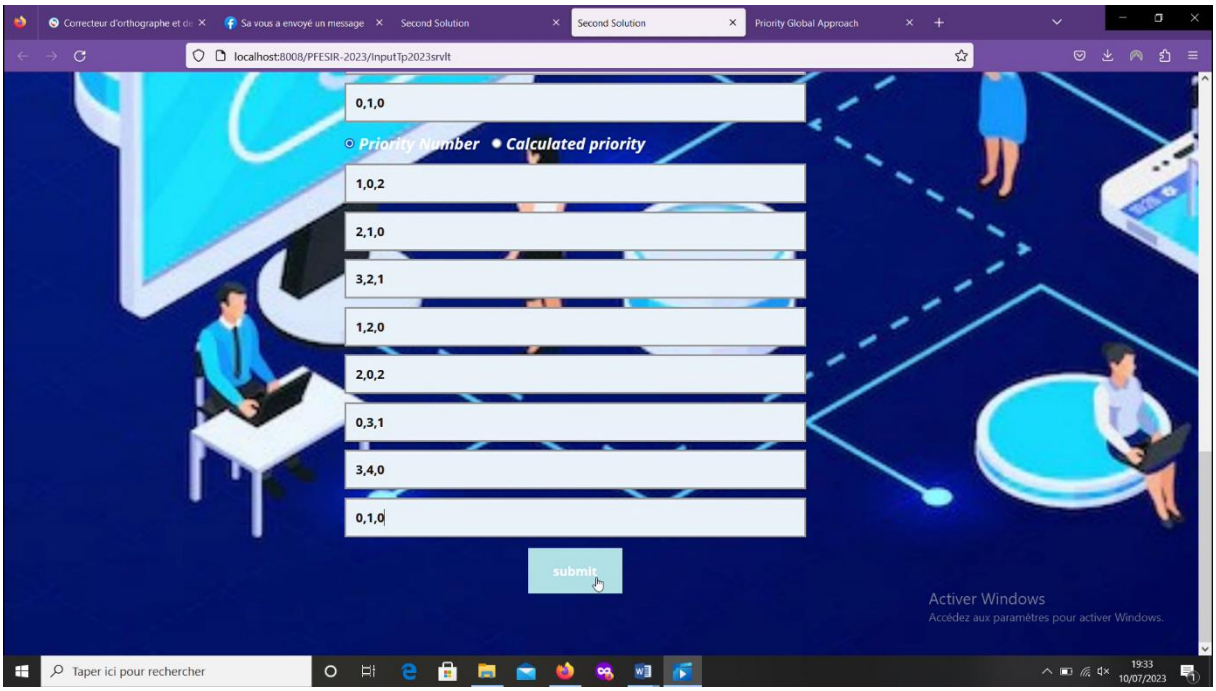
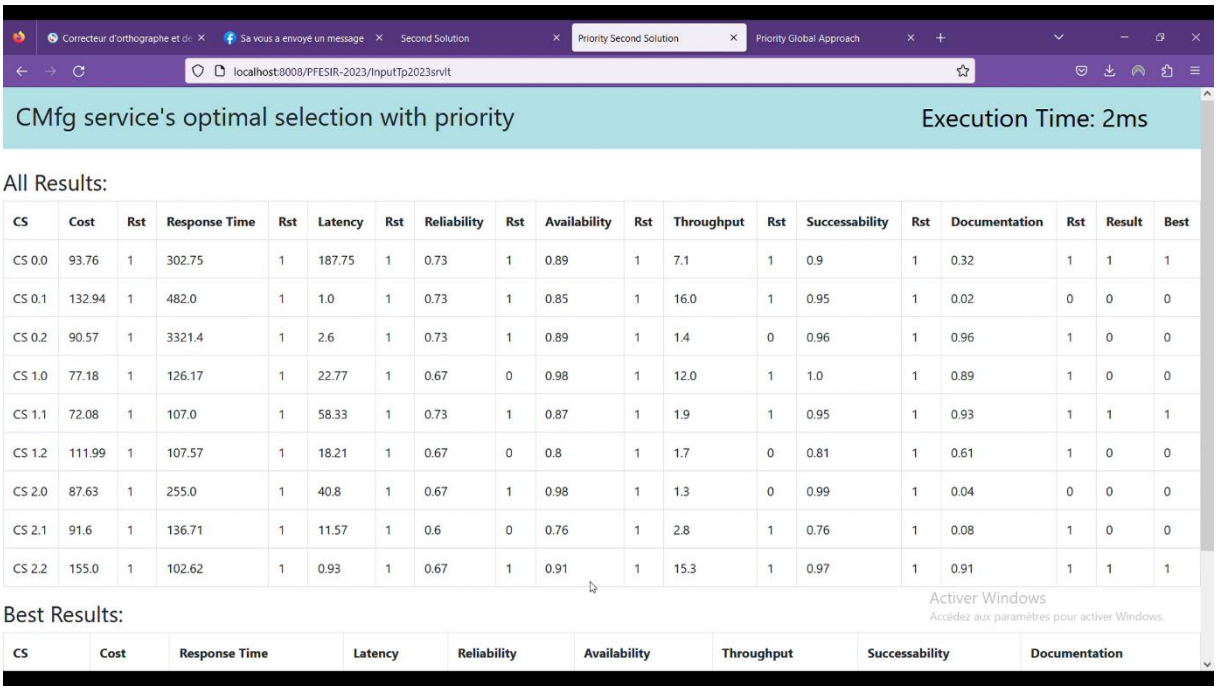


Figure 13:interface des vecteurs d'affectation pour chaque QOS entrant par l'utilisateur.

Dans la **figure 14**, en cliquant sur le bouton « Submit ». L'algorithme de priorité, et toutes les équations de la deuxième solution vues précédemment en détail dans **le chapitre 3** seront exécutés et tous les résultats seront affichés dans la liste des services candidats.

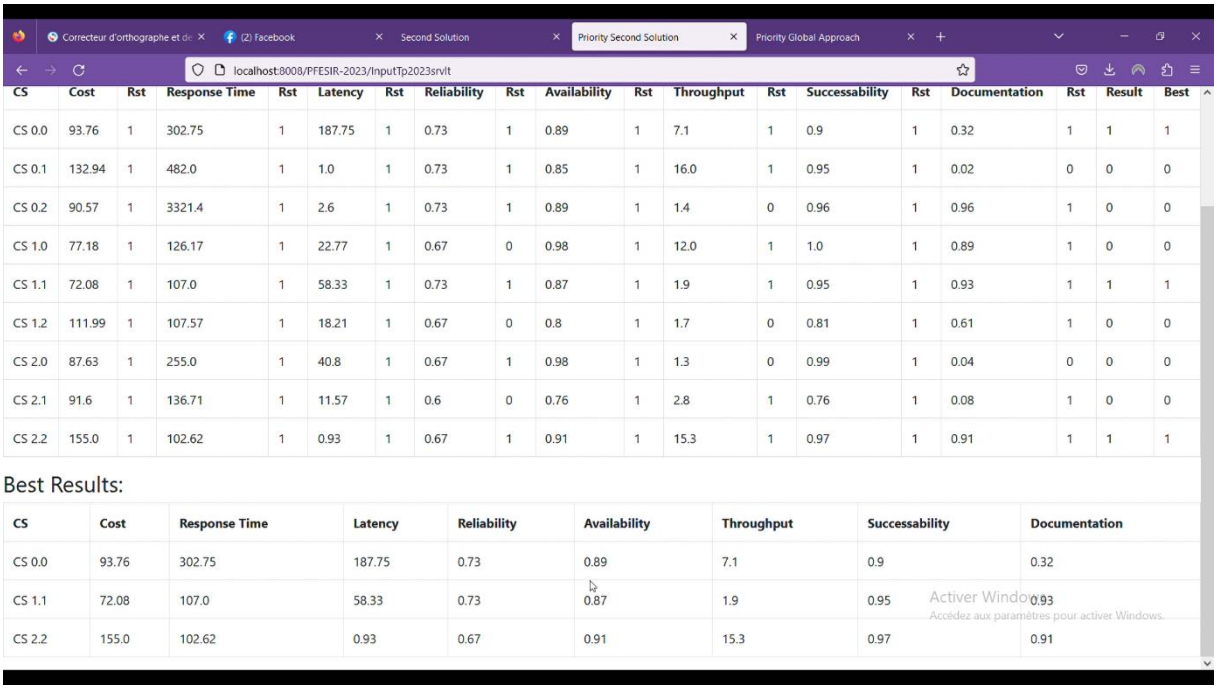


CS	Cost	Rst	Response Time	Rst	Latency	Rst	Reliability	Rst	Availability	Rst	Throughput	Rst	Successability	Rst	Documentation	Rst	Result	Best
CS 0.0	93.76	1	302.75	1	187.75	1	0.73	1	0.89	1	7.1	1	0.9	1	0.32	1	1	1
CS 0.1	132.94	1	482.0	1	1.0	1	0.73	1	0.85	1	16.0	1	0.95	1	0.02	0	0	0
CS 0.2	90.57	1	3321.4	1	2.6	1	0.73	1	0.89	1	1.4	0	0.96	1	0.96	1	0	0
CS 1.0	77.18	1	126.17	1	22.77	1	0.67	0	0.98	1	12.0	1	1.0	1	0.89	1	0	0
CS 1.1	72.08	1	107.0	1	58.33	1	0.73	1	0.87	1	1.9	1	0.95	1	0.93	1	1	1
CS 1.2	111.99	1	107.57	1	18.21	1	0.67	0	0.8	1	1.7	0	0.81	1	0.61	1	0	0
CS 2.0	87.63	1	255.0	1	40.8	1	0.67	1	0.98	1	1.3	0	0.99	1	0.04	0	0	0
CS 2.1	91.6	1	136.71	1	11.57	1	0.6	0	0.76	1	2.8	1	0.76	1	0.08	1	0	0
CS 2.2	155.0	1	102.62	1	0.93	1	0.67	1	0.91	1	15.3	1	0.97	1	0.91	1	1	1

CS	Cost	Response Time	Latency	Reliability	Availability	Throughput	Successability	Documentation

Figure 14:L'exécution de notre deuxième algorithme

Ainsi que la meilleure composition présenté dans la figure. Talque le meilleur service est choisi si tous les contraintes QOS locales prioritaires pour chaque service soit satisfait



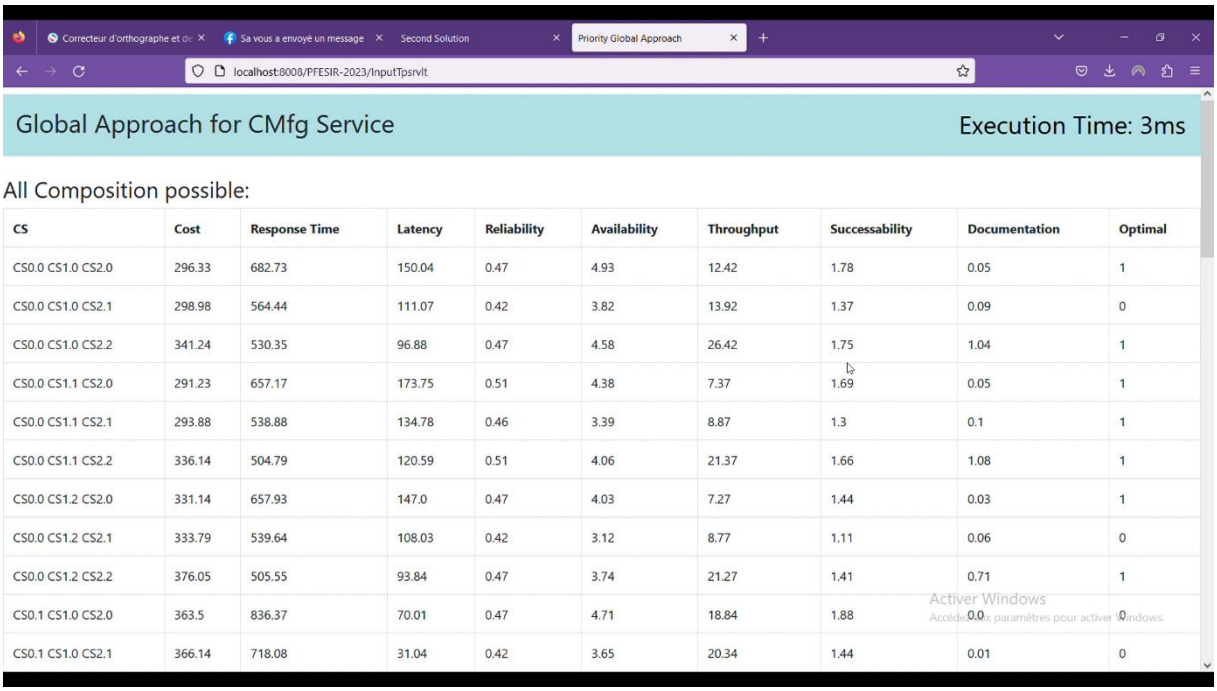
CS	Cost	Rst	Response Time	Rst	Latency	Rst	Reliability	Rst	Availability	Rst	Throughput	Rst	Successability	Rst	Documentation	Rst	Result	Best
CS 0.0	93.76	1	302.75	1	187.75	1	0.73	1	0.89	1	7.1	1	0.9	1	0.32	1	1	1
CS 0.1	132.94	1	482.0	1	1.0	1	0.73	1	0.85	1	16.0	1	0.95	1	0.02	0	0	0
CS 0.2	90.57	1	3321.4	1	2.6	1	0.73	1	0.89	1	1.4	0	0.96	1	0.96	1	0	0
CS 1.0	77.18	1	126.17	1	22.77	1	0.67	0	0.98	1	12.0	1	1.0	1	0.89	1	0	0
CS 1.1	72.08	1	107.0	1	58.33	1	0.73	1	0.87	1	1.9	1	0.95	1	0.93	1	1	1
CS 1.2	111.99	1	107.57	1	18.21	1	0.67	0	0.8	1	1.7	0	0.81	1	0.61	1	0	0
CS 2.0	87.63	1	255.0	1	40.8	1	0.67	1	0.98	1	1.3	0	0.99	1	0.04	0	0	0
CS 2.1	91.6	1	136.71	1	11.57	1	0.6	0	0.76	1	2.8	1	0.76	1	0.08	1	0	0
CS 2.2	155.0	1	102.62	1	0.93	1	0.67	1	0.91	1	15.3	1	0.97	1	0.91	1	1	1

Best Results:

CS	Cost	Response Time	Latency	Reliability	Availability	Throughput	Successability	Documentation
CS 0.0	93.76	302.75	187.75	0.73	0.89	7.1	0.9	0.32
CS 1.1	72.08	107.0	58.33	0.73	0.87	1.9	0.95	0.93
CS 2.2	155.0	102.62	0.93	0.67	0.91	15.3	0.97	0.91

Figure 15:La sélection de la meilleure composition du service de notre deuxième solution

Cette interface présente tous compositions de l'approche globale modifiée, comme présenté dans la figure

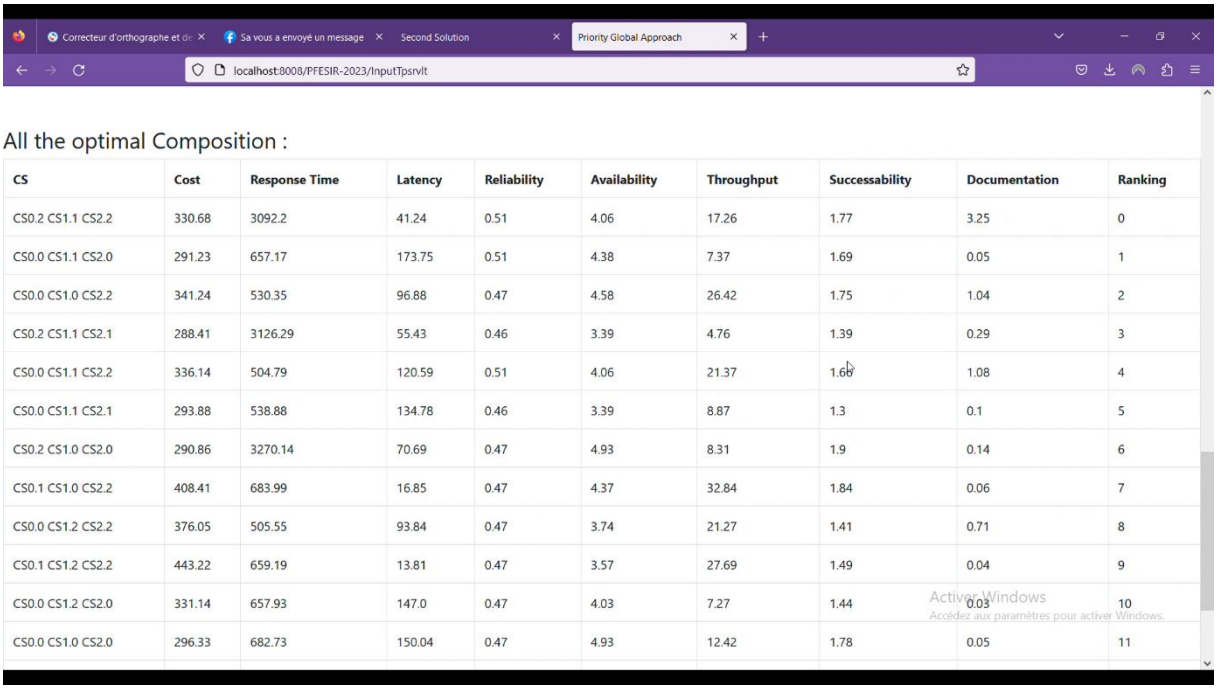


All Composition possible:

CS	Cost	Response Time	Latency	Reliability	Availability	Throughput	Successability	Documentation	Optimal
CS0.0 CS1.0 CS2.0	296.33	682.73	150.04	0.47	4.93	12.42	1.78	0.05	1
CS0.0 CS1.0 CS2.1	298.98	564.44	111.07	0.42	3.82	13.92	1.37	0.09	0
CS0.0 CS1.0 CS2.2	341.24	530.35	96.88	0.47	4.58	26.42	1.75	1.04	1
CS0.0 CS1.1 CS2.0	291.23	657.17	173.75	0.51	4.38	7.37	1.69	0.05	1
CS0.0 CS1.1 CS2.1	293.88	538.88	134.78	0.46	3.39	8.87	1.3	0.1	1
CS0.0 CS1.1 CS2.2	336.14	504.79	120.59	0.51	4.06	21.37	1.66	1.08	1
CS0.0 CS1.2 CS2.0	331.14	657.93	147.0	0.47	4.03	7.27	1.44	0.03	1
CS0.0 CS1.2 CS2.1	333.79	539.64	108.03	0.42	3.12	8.77	1.11	0.06	0
CS0.0 CS1.2 CS2.2	376.05	505.55	93.84	0.47	3.74	21.27	1.41	0.71	1
CS0.1 CS1.0 CS2.0	363.5	836.37	70.01	0.47	4.71	18.84	1.88	0.0	0
CS0.1 CS1.0 CS2.1	366.14	718.08	31.04	0.42	3.65	20.34	1.44	0.01	0

Figure 16:L'exécution d'approche globale modifié

Cette interface présente tous les meilleures compositions de l'approche globale modifiée, où nous avant utilisée l'algorithme de priorité pour le classement des compositions



All the optimal Composition :

CS	Cost	Response Time	Latency	Reliability	Availability	Throughput	Successability	Documentation	Ranking
CS0.2 CS1.1 CS2.2	330.68	3092.2	41.24	0.51	4.06	17.26	1.77	3.25	0
CS0.0 CS1.1 CS2.0	291.23	657.17	173.75	0.51	4.38	7.37	1.69	0.05	1
CS0.0 CS1.0 CS2.2	341.24	530.35	96.88	0.47	4.58	26.42	1.75	1.04	2
CS0.2 CS1.1 CS2.1	288.41	3126.29	55.43	0.46	3.39	4.76	1.39	0.29	3
CS0.0 CS1.1 CS2.2	336.14	504.79	120.59	0.51	4.06	21.37	1.66	1.08	4
CS0.0 CS1.1 CS2.1	293.88	538.88	134.78	0.46	3.39	8.87	1.3	0.1	5
CS0.2 CS1.0 CS2.0	290.86	3270.14	70.69	0.47	4.93	8.31	1.9	0.14	6
CS0.1 CS1.0 CS2.2	408.41	683.99	16.85	0.47	4.37	32.84	1.84	0.06	7
CS0.0 CS1.2 CS2.2	376.05	505.55	93.84	0.47	3.74	21.27	1.41	0.71	8
CS0.1 CS1.2 CS2.2	443.22	659.19	13.81	0.47	3.57	27.69	1.49	0.04	9
CS0.0 CS1.2 CS2.0	331.14	657.93	147.0	0.47	4.03	7.27	1.44	0.03	10
CS0.0 CS1.0 CS2.0	296.33	682.73	150.04	0.47	4.93	12.42	1.78	0.05	11

Figure 17:La sélection des meilleures compositions du service.

4 Discussion de performance de notre algorithme

Pour évaluer l'efficacité et l'optimalité de l'algorithmes proposé pour résoudre le problème d'OSCMCS, une comparaison des performances a été réalisée en utilisant nos deux solutions. Les scénarios suivants ont été utilisés comme exemples pour cette évaluation :

Scénario 1 : Composé de 6 cas de test, le nombre de sous-tâches N est fixé à 10 avec un nombre de services candidats M dans chaque sous tâche qui est égale au début à 50 CS avec un incrément de 50.

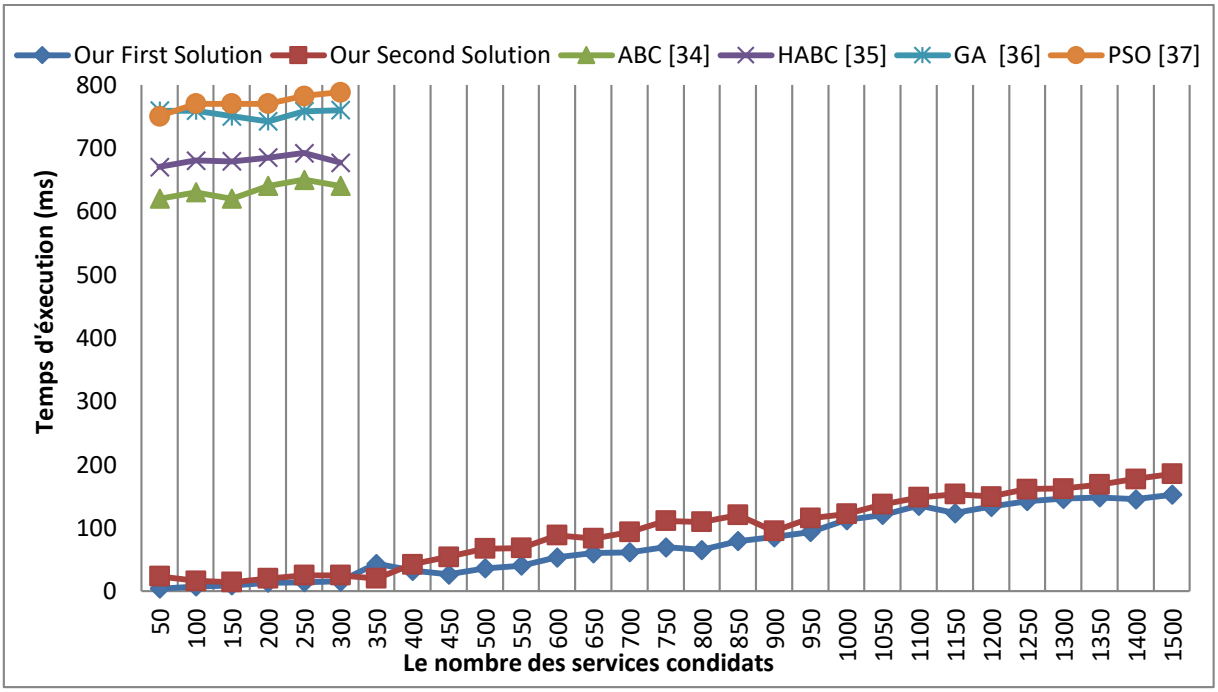


Figure 18: La comparaison de performance entre nos solution par rapport le nombre des services candidats

Résultats : Le diagramme 1 illustre la comparaison des performances de nos solutions par rapport à différentes approches ABC [34], HABC [35], GA [36] et PSO [37]) pour la sélection quasi-optimale des services CMfg. Dans notre première solution, le temps nécessaire pour sélectionner les services est de 152 ms pour 1500 CS et pour notre deuxième solution prend 185 ms, tandis que l'approche ABC prend 640 ms, l'approche HABC prend 677 ms, l'approche GA prend 780 ms et l'approche PSO prend 788 ms pour 300 CS. Ces résultats mettent en évidence la rapidité de nos solutions par rapport les autres approches, ainsi que notre première

solution par rapport aux notre deuxième solution. Et cela est dû à l'algorithme de priorité et les calcule des priorités qui nous avons ajouté aux équations de notre deuxième solution.

Scénario 2 : Composé de 7 cas de test, le nombre de services candidats M est fixé à 50 avec un nombre de sous tâches N qui est égale au début à 10 avec un incrément de 10.

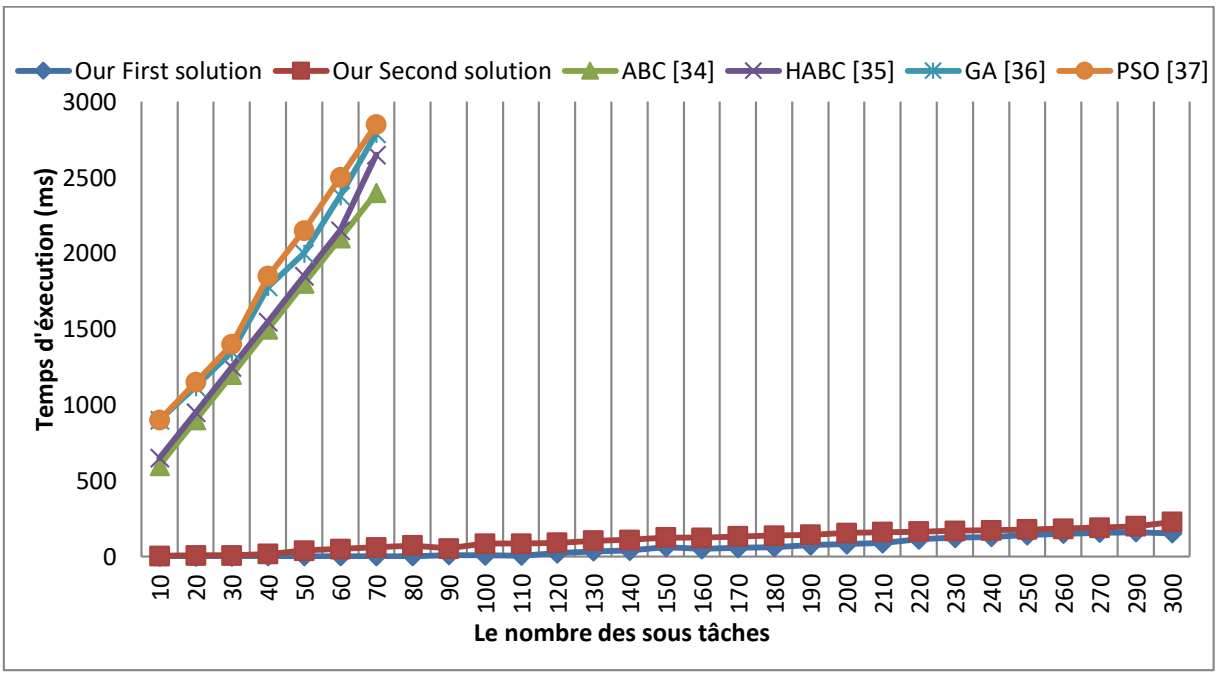


Figure 19:La comparaison de performance de nos solution par rapport le nombre des sous taches.

Résultats : La comparaison des performances de nos solutions par rapport aux approches ABC [34], HABC [35], GA [36] et PSO [37] pour la sélection quasi-optimale des services CMfg est présentée dans le diagramme 2. Dans notre première solution, le temps nécessaire pour sélectionner les services est de 154 ms pour la sélection de 300 ST avec 50 CS et pour notre deuxième solution prend 228 ms. En revanche, l'approche ABC prend 2400 ms, l'approche HABC prend 2650 ms, l'approche GA prend 2790 ms et l'approche PSO prend 2850 ms pour 70 sous tache. Ces résultats mettent en évidence la rapidité de nos solutions par rapport les autres approches, ainsi que notre première solution par rapport aux notre deuxième solution. Et cela est dû à l'algorithme de priorité et les calcule des priorités qui nous avons ajouté aux équations de notre deuxième solution.

5 Comparaison de l'optimalité de notre première approche par rapport à l'approche globale dans la sélection de services :

Afin d'évaluer l'optimalité de notre première solution, nous avons comparé le résultat obtenu par notre approche avec le résultat obtenu par l'approche globale qui donne des résultats de sélection optimaux. Pour cela, en comparant la somme des valeurs des paramètres QOS des services candidat sélectionner par notre algorithme dans chaque sous tâche avec la somme des valeurs des paramètres QOS des services candidat sélectionner par l'approche globale dans chaque sous tâche, La valeur de l'optimalité est calculée comme suite :

$$\boldsymbol{Optimalit}\text{é} = \frac{\sum_{i=1}^{n} Cost\_A[i] + \sum_{i=1}^{n} Time\_A[i] + \sum_{i=1}^{n} Latency\_A[i] + \prod_{i=1}^{n} Avail\_A[i] + \prod_{i=1}^{n} Reli\_A[i] + \sum_{i=1}^{n} Throu\_A[i] + \prod_{i=1}^{n} Succ\_A[i]}{\sum_{i=1}^{n} Cost\_G[i] + \sum_{i=1}^{n} Time\_G[i] + \sum_{i=1}^{n} Latency\_G[i] + \prod_{i=1}^{n} Avail\_G[i] + \prod_{i=1}^{n} Reli\_G[i] + \sum_{i=1}^{n} Throu\_A[i] + \prod_{i=1}^{n} Succ\_G[i]}$$

$$\frac{\prod_{i=1}^{n} Docum\_A[i]}{\prod_{i=1}^{n} Docum\_G[i]} * 100 \qquad (\mathbf{44})$$

L'équation (44) permet de comparer les résultats de sélection obtenue par notre approche à celle de l'approche globale car elle a des résultats de sélection optimale.

Scénario 3 : Dans ce scénario, nous avons évalué l'optimalité de notre premier solution en fixant le nombre de sous-tâches à 3 et en faisant varier le nombre de services candidats de 3 à 33. Les résultats d'optimalité dans plusieurs cas de test sont présentés dans le diagramme 4, où l'on observe l'impact de l'augmentation du nombre de services candidats.

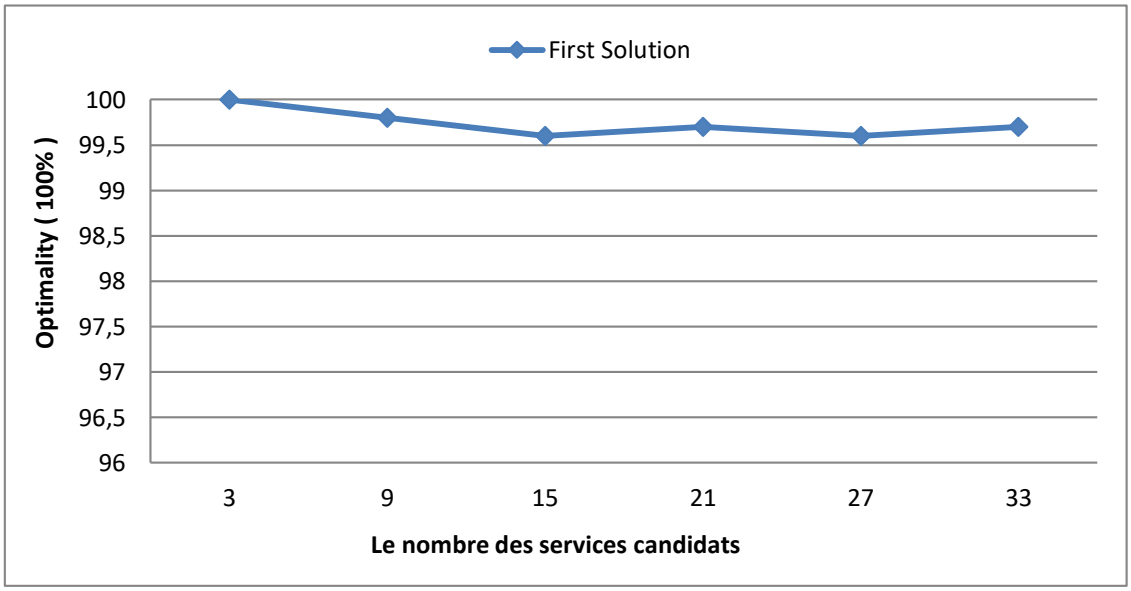


Figure 20:L'optimalité notre premier solution avec l'augmentation du nombre de service candidat.

6 Conclusion

Au fil de ce dernier chapitre, Nous avons décrit ci-dessus l'implémentation de notre algorithme conçu pour résoudre le problème de sélection optimale des services CMfg.dans le but d'aider le client à trouver le service qui répond le mieux à ses besoins. Suivant ce que nous avons présenté dans la section de discussion de performance de nos algorithmes les résultats ont clairement montré l'efficacité et l'optimalité de nos solutions.

Conclusion générale :

La composition de services vise à créer automatiquement des services complexes en utilisant des services atomiques, à travers le processus de sélection. La sélection de services consiste à choisir le service le plus approprié pour accomplir une fonctionnalité spécifique, en prenant en compte les exigences et les préférences de l'utilisateur.

Pour remédier à ce problème, nous avons proposé un nouvel algorithme de sélection des services CMfg qui tient compte de différentes contraintes QOS. Cet algorithme permet non seulement la sélection de services fiables, mais il garantit également la satisfaction des exigences de l'utilisateur. De plus ce modèle permet l'optimisation de plusieurs contraintes qui prend en compte les contraintes de demandeurs de services.

Pour valider notre algorithme, nous avons développé un programme en Java basé sur la décomposition de la contrainte globale imposée par le client en contraintes locales. Notre solution a démontré de meilleures performances en termes de fiabilité et de qualité de service, en maximisant la disponibilité, la fiabilité, la documentation, le débit et le succès. tout en réduisant le coûts, la latence et le temps de réponse.

Il convient de noter que tout travail de développement et de recherche constitue une base pour de futures améliorations et enrichissements, dans le but d'obtenir un processus plus performant. En conclusion, nous souhaitons mettre en évidence quelques perspectives qui pourraient enrichir notre algorithme à l'avenir :

- ✓ Implémenter notre solution dans un environnement réel en utilisant une plate-forme d'expérimentation.
- ✓ Considérer davantage de paramètres QOS, car notre algorithme peut être facilement modifié pour inclure d'autres critères.
- ✓ Tenir compte de la mobilité des services dans le processus de sélection.

En envisageant ces perspectives, nous espérons que notre algorithme pourra être renforcé et amélioré, contribuant ainsi au développement continu dans le domaine de la sélection de services CMfg.

Les références:

[1] Mell, P., & Grance, T. (2013). The NIST Definition of Cloud Computing (NIST Special Publication 800-145).

[2] Fisher, O., Watson, N., Porcu, L., Bacon, D., Rigley, M., & Gomes, R. L. (2018). Cloud manufacturing as a sustainable process manufacturing route. Journal of Manufacturing Systems, 47, 53-68.

[3] Wu, D., Greer, M. J., Rosen, D. W., & Schaefer, D. (2013). Cloud Manufacturing: Drivers, Current Status, and Future Trends. Volume 2: Systems; Micro and Nano Technologies; Sustainable Manufacturing.

[4] Wang, P., Gao, R. X., & Fan, Z. (2015). Cloud Computing for Cloud Manufacturing: Benefits and Limitations. Journal of Manufacturing Science and Engineering, 137(4), 044002.

[5] Qin, S. J. (2014). Process data analytics in the era of big data. AIChE Journal, 60(9), 3092-3100.

[6] Gao, J., Yao, Y., Zhu, V. C. Y., Sun, L., & Lin, L. (2009). Service-oriented manufacturing: a new product pattern and manufacturing paradigm. Journal of Intelligent Manufacturing, 22(3), 435-446.

[7] Tao, F., Zhang, L., Venkatesh, V. C., Luo, Y., & Cheng, Y. (2011). Cloud manufacturing: a computing and service-oriented manufacturing model. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 225(10), 1969-1976.

[8] Qu, L. (2016). Credible service selection in cloud environments (Doctoral dissertation, Macquarie University, Faculty of Science and Engineering, Department of Computing).

[9] Yuan, M., Zhou, Z., Cai, X., Sun, C., & Gu, W. (2020). Service composition model and method in cloud manufacturing. Robotics and Computer-Integrated Manufacturing, 61.

[10] Debabeche, O. (2020). LA COMPOSITION DE SERVICES CLOUD PAR LES ALGORITHMES EVOLUTIONNAIRES (Doctoral dissertation).

[11] Bagga, P., Joshi, A., & Hans, R. (2017). QoS based Web Service Selection and Multi-Criteria Decision Making Methods.

[12] Qin, S. J., Chen, Y., & Mu, X. (2012). An Optimal Service Selection with Constraints Based on QoS. , 25(none), 2050–2057.

[13] Li, X., Yu, S., & Chu, J. (2018). Optimal selection of manufacturing services in cloud manufacturing: A novel hybrid MCDM approach based on rough ANP and rough TOPSIS. Journal of Intelligent & Fuzzy Systems, 34(6), 4041–4056.

[14] Liu, M., Shao, Y., Yu, C., & Yu, J. (2020). A Heterogeneous QoS-Based Cloud Service Selection Approach Using Entropy Weight and GRA-ELECTRE III. Mathematical Problems in Engineering, 2020, 1–17.

[15] Abdelaziz, A. S., Harb, H., Zaghloul, A., & Salem, A. (2023). An Enhanced MCDM Model for Cloud Service Provider Selection. International Journal of Advanced Computer Science and Applications, 14(2).

[16] Loubiere, P. (2016). Amélioration des métaheuristiques d'optimisation à l'aide de l'analyse de sensibilité [Improvement of optimization metaheuristics using sensitivity analysis]. Thèse de doctorat, Paris Est.

[17] Liang, D., Wang, J., Bhamra, R., Lu, L., & Li, Y. (2022). A Multi-Service Composition Model for Tasks in Cloud Manufacturing Based on VS-ABC Algorithm. Mathematics, 10(21), 3968.

[18] Yang, B., Wang, S., Li, S., & Jin, T. (2020). A robust service composition and optimal selection method for cloud manufacturing. International Journal of Production Research, 1–19.

[19] Hao, J. K., & Middendorf, M. (2012). An NSGA-II Algorithm for the Green Vehicle Routing Problem. Lecture Notes in Computer Science, 7245, 1–13.

[20] Guo, K., Li, J., & Niu, M. (2023). Multi-agent interests service composition optimization in cloud manufacturing environment. IEEE Access, 15.

[21] Chen, C., Yu, J., Lu, J., Su, X., Zhang, J., Feng, C., & Ji, W. (2023). Service Composition and Optimal Selection of Low-Carbon Cloud Manufacturing Based on NSGA-II-SA Algorithm. Processes, 11(2), 340.

[22] Costanzo, A., Luong, T. V., & Marill, G. (2006). Optimisation par colonies de fourmis [Ant colony optimization].

[23] Yin, C., Li, S., & Li, X. (2023). A matching-synergy degree-based optimization selection method for cloud manufacturing service composition. Research Square.

[24] Cao, Y., Wang, S., Kang, L., & Gao, Y. (2016). A TQCS-based service selection and scheduling strategy in cloud manufacturing. The International Journal of Advanced Manufacturing Technology, 82(1-4), 235-251.

[25]Techno-Science.net. (s. d.). Modèle mathématique : définition et explications. Récupéré le 1 juin 2023, de https://www.techno-science.net/definition/2880.html

[26] Zhou, L., Zhang, L., Ren, L., & Wang, J. (2019). Real-Time Scheduling of Cloud Manufacturing Services Based on Dynamic Data-Driven Simulation. IEEE Transactions on Industrial Informatics, 15(9), 5042-5051.

[27] Delaram, J., & Valilai, O. F. (2018). A Mathematical Model for Task Scheduling in Cloud Manufacturing Systems focusing on Global Logistics. Procedia Manufacturing, 17, 387-394.

[28] Ahn, G., & Hur, S. J. (2021). Multio bjective Real-Time Scheduling of Tasks in Cloud Manufacturing with Genetic Algorithm. Mathematical Problems in Engineering, 2021, 1-10.

[29] Moghaddam, S. N., Akbaripour, H., & Houshmand, M. (2021). Integrated forward and reverse logistics in cloud manufacturing: an agent-based multi-layer architecture and optimization via genetic algorithm. Production Engineering, 15(6), 801-819.

[30] Yu, Y., Li, S., & Ma, J. (2021). Time-aware cloud manufacturing service selection using unknown QoS prediction and uncertain user preferences. Concurrent Engineering.

[31] Zeng, J., Yao, J., Gao, M., & Wen, J. (2022). A service composition method using improved hybrid teaching learning optimization algorithm in cloud manufacturing. Journal of Cloud Computing, 11(1).

[32] Jian, L., Youling, C., Long, W., Lidan, Z., & Yufei, N. (2018). An approach for service composition optimization considering service correlation via a parallel max-min ant system based on the case library. International Journal of Computer Integrated Manufacturing, 31(12), 1174-1188.

[33] Qi, L., Dou, W., Zhang, X., & Chen, J. (2012). A QoS-aware composition method supporting cross-platform service invocation in cloud environment. Journal of Software, 78(5), 0-0.

[34] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

[35] Han, Y. Y., Liang, J. J., Pan, Q. K., Li, J. Q., Sang, H. Y., & Cao, N. N. (2013). Effective hybrid discrete artificial bee colony algorithms for the total flowtime minimization in the blocking flowshop problem. International Journal of Advanced Manufacturing Technology, 67(1-4), 397-414.

[36] Wang, D., Yang, Y., & Mi, Z. (2015). A genetic-based approach to web service composition in geo-distributed cloud environment. Computers & Electrical.

[37] Tao, F., Zhao, D., Hu, Y., & Zhou, Z. (2010). Correlation-aware resource service composition and optimal-selection in manufacturing grid. European Journal of Operational Research, 201(1), 129–143.

[38] Al-Masri, E., & Mahmoud, Q. H. (2007, May). Discovering the best web service. In Proceedings of the 16th international conference on World Wide Web (pp. 1257-1258).

[39] Eclipse (logiciel) - Définition et Explications. (s. d.). *Techno-Science.net*. Récupéré le 1 juin 2023, de https://www.techno-science.net/glossaire-definition/Eclipse-logiciel.html

[40] Doudoux, J. M. (s. d.-a). *Développons en Java - Java Entreprise Edition*. Récupéré le 1 juin 2023, de https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee

[41] Doudoux, J. M. (s. d.-c). *Développons en Java - JSTL (Java server page Standard Tag Library)*. Récupéré le 1 juin 2023, de https://www.jmdoudoux.fr/java/dej/chap-jstl

[42] Syloé. (2022, 12 octobre). *Apache Tomcat - Glossaire Syloé - Syloe, Devops & Cloud*. Récupéré le 1 juin 2023, de https://www.syloe.com/glossaire/apache-tomcat