UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des sciences

Département d'informatique

MEMOIRE DE MASTER

En Informatique

Option : Systeme Informatique et Réseaux

THÈME :

Sélection des services dans le Cloud Manufacturing augmentés des contraintes temporelles

Réalisé par

HAMOUNI Mohammed Nadjib

BENMALEK Salah Eddine

Encadré par

Mme BEY Fella

18 Juillet 2021

Remercient

Nous remercions en premier lieu Dieu tout puissant de nous avoir donné la force et la volonté pour réaliser cette étude et tous les profs, qui ont guidé notre réflexion et on a accepté à nous rencontrer et répondre à nos questions durant notre étude intervenante et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques

Nous remercions nos très chers parents, qui ont toujours été à côté de nous

Nous sincères remercions à notre encadreuse **Mme Bey Fella** de nous avoir encadré, orienté, aidé et conseillé et de nous offrir l'opportunité de faire ce travail.

Nous exprimons toute nos reconnaissances et nos vifs remerciements aux **membres du jury** d'avoir voulu juger ce travail.

Dédicace

Je dédie cette modeste mémoire aux étres qui me sont les pluschers , je cite

mes parents les plus chers au monde , mon pére et ma mére que dieu les protége.

mes fréres alaa et akram , ma soeur widad

mes deux grand-mére

mes oncles et tantes mes cousins et cousines

notre encadreuse : Mme Bey Fella

tous mes amies aymen ,mahdi, abdou ,younes , toufik, salah , youcef,sadik

-Mohammed Nadjib

Dédicace

Je dédie ce travaille

A mes parents qui m'ont soutenue et encouragé durant ces année d'études

Qu'ils trouvent ici le témoignage de ma profonde reconnaissance

A mes frères mes grands et ceux qui ont partagé avec moi tous les

Moment d'émotion lors de la réalisation de ce travail. Ils m'ont

Chaleureusement supporté et encouragé tout au long de mon concours.

A ma famille, mes proches et à ceux qui me donnent de l'amour et de la

Vivacité.

A tous mes amis qui m'ont toujours encouragé, et à qui je souhaite plus de

Succès.

A tous ceux que j'aime.

-Salah Eddine

Résumé

L'objectif de sélection et composition des services dans le cloud manufacturing augmentés des contraintes temporelles est de déterminer la meilleure combinaison des services candidats en terme de temps d'exécution. Cette composition apparaîtra au client comme un service unique car elle est transparente pour lui.
L'un des problèmes majeurs dans la sélection et composition des services CMfg augmentés des contraintes temporelles est l'affectation d'un ensemble de contrainte temporelle prioritaires à un ensemble des services candidats. Nous proposons dans la première contribution un nouvel algorithme de sélection augmenté de contraintes temporelles prioritaires tout en considérant les dépendances entre ces contraintes. L'approche de sélection classe tous les services candidats et trouve la meilleure combinaison de service en évitant une recherche exhaustive dans tout le plan d'exécution.
Finalement, nous présentons plusieurs expériences et une évaluation exhaustive de performance pour évaluer l'efficacité de notre approche en utilisant un jeu de données (dataset).

Abstract

The objective of selection and composition of services in cloud manufacturing plus time constraints is to determine the best combination of candidate services in terms of execution time. This composition will appear to the customer as a unique service because it is transparent to him. One of the major problems in the selection and composition of CMfg services plus time constraints is the assignment of a set of priority time constraints to a set of candidate services. We propose in the first contribution a new selection algorithm augmented by priority temporal constraints while considering the dependencies between these constraints. The screening approach ranks all candidate departments and finds the best combination of departments by avoiding an exhaustive search throughout the execution plan.
Finally, we present several experiments and a comprehensive performance evaluation to assess the effectiveness of our approach using a dataset.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

Abréviation	**Définition**
ABC	Artificial Bee Colony
ACO	Ant Colony Optimizer
ANN	Artificial neural network
API	Application Programming Interface
Block-SC	Blockchain-based service composition
caABC	context-aware Artificial Bee Colony
CAD	Computer-aided design
CSSC	Correlation Sensitive Service Composition
CCOA	Chaos Control Algorithm
CMfg	Cloud Manufacturing
COOR	Combinatorial optimization of resources
DGA	Distributed genetic algorithm
DRaaS	Disaster Recovery as a Service
EA-SCOS	Energy Aware- Service Composition and Optimal Selection
EMOGWO	Grey Wolf Optimizer
ERP	Enterprise resource planning
MUM	Manufacturers to Users Mod
GA	genetic algorithm
GWO	Grey Wolf Optimizer
HABC	Hybrid Artificial Bee Colony
IaaS	Infrastructure as a Service
IHDETBO	Improved hybrid differential evolution and optimized based on teaching
LC	Local Constraint
MaaS	Manufacturing-as-a-Service

MBSPHE-CSCCM	Multi-Batch Subtasks Parallel-Hybrid Execution Cloud Service Composition for Cloud Manufacturing
MIaaS	Monitoring- Integration-As-A-Service
MO-SCOS	Multi Objectif- Service Composition and Optimal Selection
MPaaS	Mobile Platform as a Service
MROCl	Manufacturing resource optimization combinatorial
MSaaS	Managed software as a service
MSCOS	Manufacturing Service Composition and Optimal Selection
OBTL	Optimization Based on Teaching-Learning
OWL-S	Ontology web language for services
PaaS	Platform as a Service
PSO	Particle Swarm Optimization
Q-SAA	QoS-Sensitive sensor allocation algorithm
QoS	Quality of Service
QoS-EnCon	Quality of Service-Energy Consumption
RaaS	Recovery as a Service
SaaS	Software as a Service
SCOS	Service Composition and Optimal Selection
SFLA	Shuffled Frog-Leaping
SoM	Self-Organizing Map
SSAPTC	Service Selection Algorithm with Priority Time Constraint
TBBO	Biogeography-Based Optimisation
TC	Time Constraint
TQCS	Temps, Qualité, Coût et Service
VM	Virtual Machine
XaaS	X as a Service

Introduction Générale

Au cours de la dernière décennie, un nouveau paradigme informatique : le cloud computing a émergé grâce à la disponibilité de réseaux performants, d'ordinateurs à faible coût et de stockage ainsi que l'adoption généralisée de la virtualisation matérielle, Architecture Orientée Services (SOA) et informatique autonome et utilitaire. Cloud computing est un modèle de prestation de services et d'accès où il est dynamiquement évolutif et les ressources virtualisées sont fournies en tant que service avec une fiabilité et une évolutivité élevées et la disponibilité sur Internet. Le cloud computing introduit un nouveau mode d'exploitation et modèle commercial qui permet aux clients de ne payer que pour les ressources qu'ils utilisent réellement au lieu de faire de lourds investissements initiaux. Cela crée une toute nouvelle opportunité pour entreprises avec les avantages d'une performance plus élevée, d'un coût inférieur, d'une évolutivité élevée, disponibilité et accessibilité, et réduction des risques commerciaux et des dépenses de maintenance. Le cloud computing repose sur le partage des ressources pour assurer cohérence et économie d'échelle.
Le cloud manufacturing est également un modèle de fabrication en réseau intelligent qui englobe le cloud computing, visant à répondre aux demandes croissantes d'individualisation accrue des produits, de coopération mondiale plus large, d'innovation à forte intensité de connaissances et d'agilité accrue dans la réponse du marché.
CMfg a de nombreuses problèmes à étudier. Par exemple, la classification et virtualisation de ressource, description des tâches, et sélection et composition optimales de services, parmi lesquelles la dernière (sélection et composition optimales du service) semble être un des défis les plus importants.
Le problème de sélection optimale des services de fabrication joue un rôle important dans la recherche du meilleur service de fabrication pour des tâches spécifiques dans CMfg environnement.

Problématique

Proposer un nouvel algorithme de sélection du service Cloud qui classe tous les services

candidat et trouve la combinaison optimale du service Cloud en évitant une recherche exhaustive dans tous les plans d'exécution.
Le « Cloud Manufacturing » est un nouveau concept en expansion basé sur le Cloud qui permet avec un ordinateur et une connexion internet d'accéder à des services dont la production nécessite toute une infrastructure (logiciels, machines,. . .) généralement complexe et coûteuse. La différence avec le Cloud Manufacturing vient du fait que les services deviennent dorénavant des objets matériels. Comme exemple d'infrastructure Cloud Manufacturing, une usine avec des machines qui fabriquent des objets matériels. Cela vous permet donc de faire fabriquer n'importe quel objet sans pour autant posséder un quelconque moyen de production. Ceci est possible grâce à l'impression 3D qui permet en effet de fabriquer à la chaîne des produits.
La découverte et la sélection des services Cloud représentent un axe de recherche émergeant. Dans cette mémoire, nous nous intéressons à l'étude des contraintes temporelles dans la sélection des services Cloud.
L'un des problèmes cruciaux dans les systèmes Cloud Manufacturing (CMfg) est que de plus en plus de services proposés dans CMfg peuvent fournir les mêmes fonctionnalités mais présenter des performances différentes. Pour s'assurer que le Cloud Manufacturing répond aux exigences des tâches composites complexes, la sélection optimale du service CMfg (OSCMCS) devient un problème de plus en plus important. Notre mémoire contribue à la phase de composition et de sélection des services CMfg et propose un nouvel algorithme de sélection enrichi de contraintes de temps hiérarchisées tout en considérant les dépendances entre elle. Nous présentons les algorithmes de sélection par score des services Cloud qui classent tous les services concrets et trouvons la combinaison optimale du service CMfg en évitant une recherche exhaustive dans tous les plans d'exécution.

Organisation du mémoire

Généralité sur le cloud computing et le cloud manufacturing : ce chapitre aborde les définitions du cloud computing, son architecture et ses caractéristiques, les modèles de service, les types de cloud ainsi que le cloud manufacturing et sa relation avec le cloud computing.

Etat de l'art : ce chapitre aborde un état de l'art du domaine de la sélection du service CMfg en se basant sur les contraintes, suivi d'une classification de toutes les approches étudiées.

Approche de selection des services cloud manufacturing augmentée de contraintes temporelles : Dans ce chapitre, nous proposons une approche de sélection de services dans le cloud manufacturing augmentés des contraintes temporelles sous différentes priorités.

Realisation et experimentations : dans ce chapitre nous avons évalué notre approche en utilisant des ensembles de données de services cloud pour simuler les enregistrements de temps de réponse historiques de nombreux services fonctionnellement équivalents.

Chapitre 1

Généralité sur le cloud computing et le cloud manufacturing

1.1 Introduction :

Le cloud computing est considéré comme l'évolution d'une variété de technologies qui se sont réunis pour changer l'approche des organisations pour construire leur infrastructure informatique. En fait, il n'y a rien de nouveau dans aucune des technologies qui sont utilisé dans le cloud computing où la plupart de ces technologies sont connues pendant très longtemps. Il s'agit de les rendre tous accessibles aux masses sous le nom cloud computing. Le cloud n'est pas simplement le dernier terme pour Internet, bien que Internet est une base nécessaire pour le cloud, le cloud est quelque chose de plus qu'Internet. Le cloud est l'endroit où vous allez utiliser la technologie quand vous en avez besoin, aussi longtemps que vous en avez besoin. Vous n'installez rien sur votre bureau, et vous ne payez pas pour la technologie lorsque vous ne l'utilisez pas.[13]
Le cloud computing est un nouveau style d'informatique dans lequel les ressources évolutives dynamiquement sont fournis sous forme de services virtualisés, cela permet aux fournisseurs de services et aux utilisateurs d'ajuster leur capacité de calcul en fonction de la quantité nécessaire à un temps donné ou pour une tâche donnée. [23]

1.2 Cloud computing :

Lorsque on branche un appareil électrique dans une prise, on ne soucie ni de la façon dont l'électricité est générée ni comment elle parvient à cette prise. Ceci est possible car l'électricité est virtualisée ; c'est-à-dire qu'il est facilement disponible à partir d'une prise murale qui cache des centrales électriques et un immense réseau de distribution. Lorsqu'il est étendu aux technologies de l'information, Ce concept signifie fournir des fonctions utiles tout en cachant ses fonctions internes, l'informatique elle-même, peut être considérée comme entièrement virtualisée, qui doit

permettre de construire des ordinateurs à partir de composants distribués, tels que des ressources de traitement, de stockage, de données et de logiciels. [12]
Les technologies telles que les groupes, les grilles et maintenant l'informatique dans les nuages sont toutes conçues pour permettre l'accès à de grandes quantités de puissance de calcul dans un environnement entièrement virtualisé, en agrégeant les ressources et en offrant une vue système unique.[7]
Le cloud computing a été inventé comme un terme générique pour décrire une catégorie de services informatiques sophistiquées à la demande, initialement proposée par des fournisseurs tels qu'Amazon, Google et Microsoft. Il désigne un modèle sur lequel l'infrastructure informatique est considérée comme un « cloud », à partir duquel les entreprises et les particuliers accèdent à des applications de n'importe où dans le monde à la demande.[8]
Le principe de base de ce modèle est d'offrir le calcul, le stockage et le logiciel en tant que service.

1.3 Architecture de cloud computing :

Cette section décrit les aspects architecturaux, commerciaux et divers modèles de fonctionnement du cloud computing.

1.3.1 Modèle en couches de cloud computing :

De manière générale, l'architecture d'un environnement de cloud computing peut être divisée en 4 couches : la couche matériel / centre de données, la couche infrastructure, la couche de plateforme et la couche d'application, comme le montre la figure 1.1 [43] nous décrivons chacune d'elles en détail :

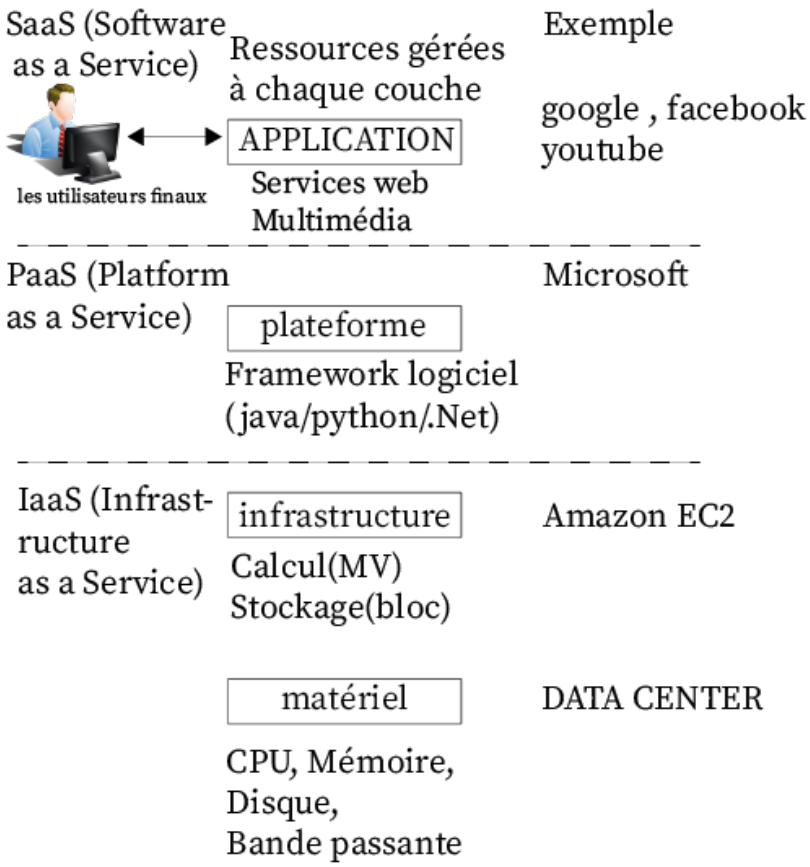


FIGURE 1.1 – l'architecture de cloud computing [43]

1.3.1.1 La couche matériel :

Cette couche est chargée de gérer les ressources physiques du cloud, y compris les serveurs physiques, les routeurs, les commutateurs, les systèmes d'alimentation et de refroidissement, en pratique, la couche matérielle est généralement implémentée dans les centres de données, un centre de données contient généralement des milliers de serveurs organisés en racks et interconnectés via des commutateurs, des routeurs ou d'autres outils, problèmes typiques au niveau de la couche matérielle incluent la configuration matérielle, la tolérance aux pannes, la gestion du trafic, les ressources d'alimentation et la gestion de refroidissement. [43]

1.3.1.2 La couche infrastructure :

Également appelée couche de virtualisation, la couche d'infrastructure crée un pool de stockage et des ressources informatiques en partitionnant les ressources physiques à l'aide des technologies de virtualisation, la couche d'infrastructure est une composante essentielle du cloud computing, car de nombreux fonctionnalités telles que l'affectation dynamique des ressources, sont uniquement mis à disposition grâce aux technologies de virtualisation. [43]

1.3.1.3 La couche de plateforme :

Construit au-dessus de couche infrastructure, la couche plate-forme se compose de systèmes d'exploitation et cadres d'application, le but de la couche de plate-forme est de minimiser le fardeau du déploiement direct des applications dans des conteneurs par exemple, Google App Engine fonctionne au niveau de la plate-forme pour fournir une prise en charge API (Application Programming Interface) pour la mise en œuvre du stockage, de la base de données et de la logique métier du Web typique applications. [43]

1.3.1.4 La couche d'application :

Au niveau le plus élevé de la hiérarchie, la couche application se compose des applications cloud proprement différentes des applications traditionnelles, les applications cloud peuvent tirer parti de la fonction de mise à l'échelle automatique pour meilleures performances, disponibilité et coûts d'exploitation réduits, par rapport aux environnements d'hébergement des services traditionnels comme les fermes de serveurs dédiés, l'architecture du l'informatique de cloud est plus modulaire. [43]

1.3.2 Modèles de service :

Dans le cloud computing, tout est traité comme un service (c'est-à-dire XaaS), par ex. SaaS, PaaS et IaaS. Ces services sont généralement fournis selon les normes de l'industrie interfaces telles que les services Web, SOA (Service-Oriented Architecture) ou REST (Representational State Transfer), prestations de service :

1.3.2.1 SaaS (Software as a Service) :

Dans ce modèle, les fournisseurs de services cloud sont responsables de l'exécution et la maintenance du logiciel d'application, du fonctionnement système et autres ressources. Le modèle SaaS apparaît au client en tant qu'interface d'application Web où Internet est utilisé pour fournir des services accessibles à l'aide d'un navigateur Web. Les applications hébergées comme Gmail et Google Docs peut être accédé via différents appareils tels que téléphones intelligents et ordinateurs portables, etc. Contrairement aux logiciels traditionnels, Le SaaS présente l'avantage que le client n'a pas besoin d'acheter des licences, installer, mettre à niveau, maintenir ou exécuter des logiciels sur son propre ordinateur. [29]

1.3.2.2 PaaS (Platform as a Service) :

Il s'agit d'un type de service cloud computing plus avancé. Dans PaaS, un fournisseur de services cloud propose, exécute et entretient à la fois le logiciel système (c'est-à-dire le système d'exploitation) et d'autres ressources informatiques. Les services PaaS comprennent la conception, développement et hébergement d'applications. Autres services inclure la collaboration,

l'intégration de base de données, la sécurité, le service Web intégration, mise à l'échelle, etc. Les utilisateurs n'ont pas à s'inquiéter de la disposition de leurs propres ressources matérielles et logicielles ou embaucher des experts pour la gestion de ces ressources. Ce schéma offre une flexibilité dans l'installation du logiciel sur le système, l'évolutivité est un autre avantage du PaaS. [29]

1.3.2.3 IaaS (Infrastructure as a Service) :

Dans (Iaas), le fournisseur de services en cloud fournit un ensemble de ressources informatiques virtualisées telles que CPU, mémoire, système d'exploitation, et les logiciels d'application ...etc, En utilisant des technologie de virtualisation pour convertir les ressources physiques en ressources logiques pouvant être provisionnées dynamiquement et publié par les clients au besoin. Certains des principaux entreprises offrant une infrastructure en tant que service comprennent : Serveurs Cloud Rackspace, Google, Amazon EC2, IBM et Verizon. [29]

1.3.2.4 RaaS (Recovery as a Service) :

Les solutions de récupération en tant que service RaaS aident les entreprises à remplacer leur sauvegarde, archivage, reprise après sinistre et des solutions de continuité d'activité en une seule Plate-forme. Les fournisseurs RaaS aident les entreprises à récupérer les centres de données, les serveurs (OS, applications, configuration et data) et les fichiers de base de données. RaaS aide les établissements commerciaux à réduire l'impact des temps d'arrêt en cas de catastrophe ou de situations similaires. RaaS est également appelé DRaaS (Disaster Recovery as a Service) WindStream est un exemple d'entreprises faisant du RaaS Affaires, etc. [29]

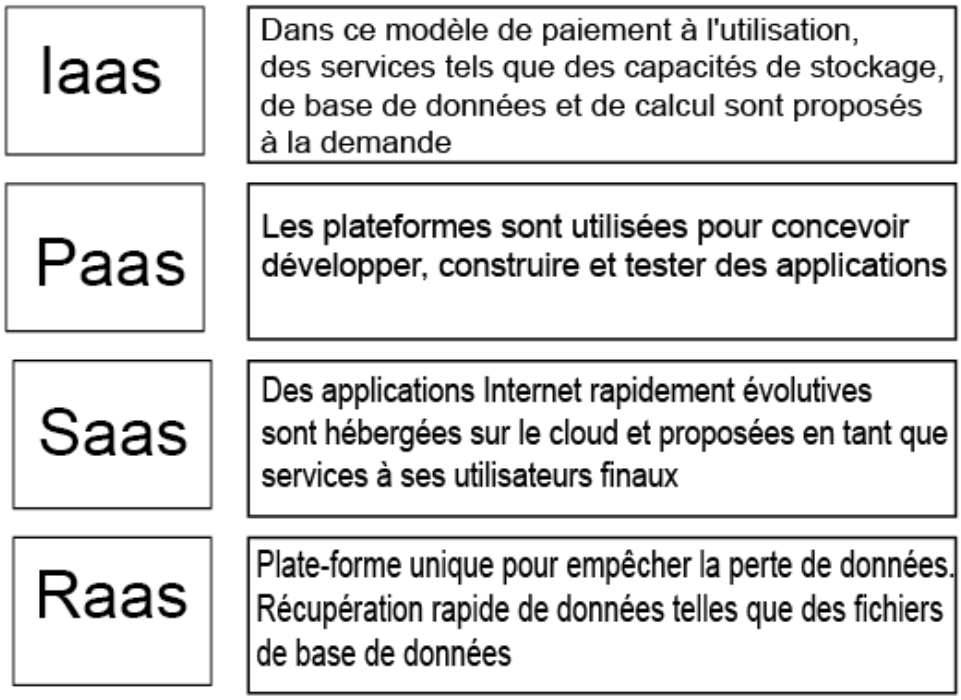


FIGURE 1.2 – Modèles des services[29]

1.4 Types de cloud :

Il existe de nombreux problèmes à prendre en compte lors du déplacement d'une application d'entreprise vers l'environnement cloud. Par exemple, certains fournisseurs de services souhaitent surtout réduire les coûts d'exploitation, tandis que d'autres préfèrent une fiabilité et une sécurité élevées. En conséquence, il existe différents types de cloud, chacun avec ses propres avantages et inconvénients :

1.4.1 Cloud public :

Cloud dans lequel les fournisseurs de services offrent leurs ressources en tant que services au grand public. Les cloud publics offrent plusieurs avantages clés aux fournisseurs de services, notamment l'absence d'investissement initial en capital sur l'infrastructure et transfert des risques vers les fournisseurs d'infrastructure. Cependant, les clouds publics manquent de contrôle précis sur les données, le réseau et paramètres de sécurité, ce qui nuit à leur efficacité dans de nombreux scénarios commerciaux. [43]

1.4.2 Cloud privé :

Également appelés cloud internes, privés les cloud sont conçus pour une utilisation exclusive par une seule organisation. Un cloud privé peut être créé et géré par l'organisation ou par des fournisseurs externes. Un cloud privé offre le plus haut degré de contrôle sur les performances,

la fiabilité et la sécurité. Cependant, ils sont souvent critiqués pour être similaires aux fermes de serveurs propriétaires traditionnels et ne fournissent pas des avantages tels que l'absence de coûts d'investissement initiaux. [43]

1.4.3 Cloud hybride :

Un cloud hybride est une combinaison des modèles des cloud privé et public qui tentent de remédier aux limitations de chaque approche. Dans un cloud hybride, une partie de l'infrastructure de service s'exécute dans des cloud privés tandis que la partie restante s'exécute dans les cloud publics. Les cloud hybrides offrent plus de flexibilité que les cloud publics et privés. Plus précisément, ils fournissent un contrôle et une sécurité plus stricts sur les données des applications par rapport aux cloud publics, tout en facilitant toujours l'expansion et la contraction des services à la demande. Du côté négatif, la conception d'un cloud hybride nécessite de déterminer soigneusement la meilleure répartition entre les composants de cloud public et privé. [43]

1.5 Caractéristique de cloud computing :

Le cloud computing est composé de cinq caractéristiques essentielles :

1.5.1 Libre-service à la demande :

Un consommateur peut provisionner unilatéralement les capacités informatiques telles que l'heure du serveur et le stockage réseau, selon les besoins automatiquement sans nécessiter d'interaction humaine avec chaque fournisseur de services.

1.5.2 Large accès au réseau :

Les consommateurs peuvent accéder aux ressources cloud sur Internet à tout moment et à partir de n'importe où (c'est-à-dire omniprésent) à travers différents types de dispositifs (par exemple, téléphones mobiles et ordinateurs portables). [29]

1.5.3 Mise en commun des ressources :

Les ressources informatiques du fournisseur sont mises en commun pour servir plusieurs consommateurs utilisant un modèle multi-tenant, avec différentes ressources virtuelles affectées et réaffectées dynamiquement en fonction des demandes de consommateur. Il y a un sentiment d'indépendance de l'emplacement en ce que le client n'a généralement aucun contrôle ou connaissance de l'emplacement exact des ressources fournies, mais peut être en mesure de spécifier l'emplacement à un niveau d'abstraction plus élevé. (Par exemple, pays, état ou centre de données). Des exemples de ressources incluent le stockage, traitement, mémoire et bande passante réseau. [29]

1.5.4 Élasticité rapide :

Les capacités peuvent être provisionnées et libérées élastiquement, en certains cas automatiquement, pour évoluer rapidement vers l'extérieur et vers l'intérieur proportionnellement à la demande. Pour le consommateur, les capacités disponibles pour l'approvisionnement souvent semblent illimitées et peuvent être voulus en toute quantité à tout moment. [29]

1.5.5 Service mesuré :

Les systèmes cloud contrôlent et optimisent automatiquement les ressources utiliser en tirant parti d'une capacité de mesure à un certain niveau d'abstraction approprié au type de service (par exemple, stockage, traitement et bande passante). L'utilisation des ressources peut être surveillée, contrôlée et rapportée, en fournissant la transparence tant pour les fournisseurs que pour les consommateurs du service utilisé. [29]

1.6 Les domaines d'applications de cloud computing :

Le cloud computing est l'un des domaines les plus dominants, par conséquent le cloud computing a son importance dans les domaines suivants :

1.6.1 Apprentissage en ligne :

En ce qui concerne l'éducation, ce domaine fournit un environnement attrayant pour les étudiants, les chercheurs et les professeurs membres. Les étudiants, les membres du corps professoral et les chercheurs peuvent se connecter au cloud de leur organisation et accéder aux données et des informations à partir de là. [29]

1.6.2 Gouvernance électronique :

Le gouvernement peut améliorer son fonctionnement en introduisant Cloud computing. De cette façon, les services fournis par les diverses organisations gouvernementales peuvent être livrées dans un manière améliorée et plus sophistiquée. Cloud computing réduira également le fardeau de la gestion, de l'installation et mise à niveau des applications. [29]

1.6.3 Planification des ressources d'entreprise (ERP) :

Lorsque l'activité d'une organisation se développe, l'utilisation du cloud dans ERP voit le jour. Le travail de gestion des applications, des ressources humaines, la paie, etc. devient cher et complexe. Pour surmonter ces problèmes, les fournisseurs de services peuvent installer l'ERP dans le cloud lui-même. [29]

1.7 Les avantages du cloud computing :

Un démarrage rapide : Le cloud computing permet de tester le plan économique rapidement, à coûts réduits et avec facilité.
L'agilité pour l'entreprise : Résolution des problèmes de gestion informatique simplement sans avoir à vous engager à long terme.
Une sélection plus rapide des ressources : Réduisons le temps de recherche pour les développeurs sur le paramétrage des applications.
Pas de dépenses de capital : Plus besoin des locaux pour élargir les infrastructures informatiques.

1.8 Le cloud manufacturing :

De nos jours, la mondialisation de la fabrication, la personnalisation des produits, la réduction des délais de mise sur le marché et la fabrication verte et à faible émission de carbone ont posé de grands défis pour les entreprises de fabrication. Dans ce contexte, il devient de plus en plus important pour que les entreprises de fabrication utilisent efficacement les ressources de fabrication mondiales et collaborent efficacement avec des partenaires commerciaux, ce qui nécessite la mise en place d'une infrastructure flexible pour faciliter l'intégration efficace des ressources de fabrication internes et externes et la coopération et partage des ressources entre différentes entreprises.
De nombreux modes / systèmes et technologies de fabrication avancés, comme la fabrication agile, la fabrication virtuelle, le fournisseur de services d'application et la grille de fabrication, ont a été proposé pour résoudre ce problème. Cependant, il y a certains problèmes, tels que l'évolutivité, l'agilité et l'interopérabilité, avec les modes et systèmes de fabrication traditionnels, c.-à-d. de mécanismes de gestion centralisés, manque d'interopérabilité des systèmes d'information de l'entreprise et le manque de structures et de procédures adaptatives.
Parallèlement, le développement rapide des technologies de l'information telles que le cloud computing, l'IoT (Internet of things), les technologies orientées services fournissent le modèle opérationnel et les technologies pour surmonter les problèmes rencontrés par l'industrie manufacturière. Dans ce contexte, le concept de CMfg a été proposé. [30]
Le cloud manufacturing, en tant que nouvelle technologie axée sur les services, vise à fournir à la demande des services de fabrication sur Internet en facilitant la collaboration entre les différents producteurs avec ressources et capacités de fabrication. [6]
En fait, CMfg déménage vers l'offre de ressources et de capacités industrielles en tant que services sur Internet, semblable à ce qui est maintenant disponible en cloud computing pour le stockage, le traitement logiciels Power ou Pay-as-you-go. Le cloud manufacturing a été proposée pour fournir à la demande services de fabrication en réseau grâce à la collaboration de fournisseurs de services distribués pour satisfaire les exigences des demandeurs de service, dans

ce paradigme, diverses ressources de fabrication sont interconnectées via l'internet des objets. Ces ressources peuvent être proposées sous forme de services cloud de fabrication, et peuvent être contrôlés et géré intelligemment par la plate-forme cloud. En fait, La plateforme CMfg exécute les tâches des demandeurs en décomposant intelligemment les tâches, récupérant les ressources fonctionnellement similaires et invoquant, et composant les services pour atteindre l'optimum état.

La figure 1.3 [30] montre schématiquement comment tout le cycle de vie du produit peut être contrôlé sous la gestion des services de fabrication

Et aussi montré le début de vie, le milieu de vie et la fin de vie de service CMfg, la gestion des services de fabrication peut être divisé en étapes suivantes : étape de génération de service, étape pré-demande de service, étape demande de service et l'étape après demande de service. [30]

L'étape de génération de service dans le début de vie couvre le processus d'association des différentes ressources et capacités physiques de fabrications aux services virtuel. Il contient la perception et la connexion de ressources, transmission de données de perception, filtrage et traitement de données, virtualisation de ressources, modélisation de services et description numérique. À partir du moment où plusieurs services sont générés, le service d'agrégation émergera progressivement. Il révèle le résultat que tous les services sont regroupés et intégré par diverses corrélations et relations abstraites (c'est-à-dire les fonctions, une variété d'attributs, etc.).[30]

L'étape de pré-demande du service dans le milieu de vie, est l'échelle portée par les demandes du service généré à être mis en service. Les opérations sont la recherche et correspondance des services, l'évaluation des services, sélection et composition des services, planification des services et transaction des services.[30]

Au stade de la demande du service dans le milieu de vie, se déroule l'exécution et la surveillance des services, la tolérance aux pannes des services et la logistique des services pour la mise en œuvre des tâches. Selon les différentes exigences de la mise en œuvre des tâches, la logistique pourrait être le transport des résultats après l'exécution du service, ainsi que le transport des matériaux dans le processus d'exécution du service. Quant à l'étape d'après demande du service, il s'agit principalement de la libération de service et la désagrégation après l'exécution en fin de vie. [30]

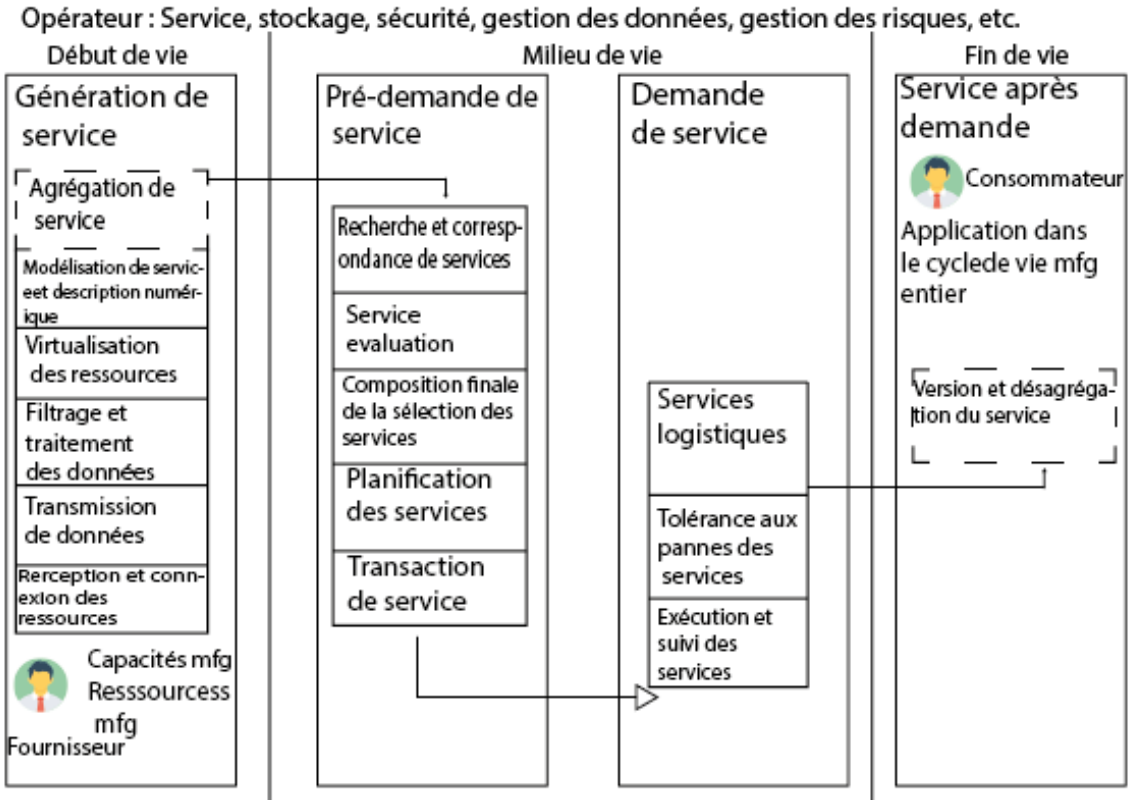


FIGURE 1.3 – plateforme de services CMfg [30]

1.9 Architecture de cloud manufacturing :

L'architecture de cloud manufacturing se compose de six couches :

1.9.1 Couche physique :

Cette couche est la somme de toutes les ressources de fabrications et les capacités de fabrication, y compris équipement de fabrication, centre de traitement, documentation technique, serveurs, logiciels et d'autres auxiliaires technologiques, etc., qui se dispersent dans le terminal cloud, ces ressources physiques de fabrication sont éparpillées partout sur le terminal cloud, qui se connecte à la plate-forme cloud de fabrication à travers les correspondantes interfaces, équipements et middleware. [26]

1.9.2 Couche d'interface orientée ressources :

Grâce à l'IOT et au terminal cloud intégré technologie, etc…, cette couche conçoit le protocole pour planifier les ressources de manière raisonnable en temps réel. Selon une certaine forme standard, il transmet diverses ressources de fabrication et capacités de fabrication du terminal cloud et les organise, pour accéder, décomposer, grouper, emballer, stocker, ainsi qu'une

série de traitement de la technologie et intégration des ressources. Il fixe également un prix et règle les comptes conformément avec le contenu technologique de ses ressources, pour répartir raisonnablement les intérêts des ressources. [26]

1.9.3 Couche de ressources virtuelles :

Cette couche analyse et traite principalement les ressources de fabrication qui se sont rassemblées en terminal de cloud manufacturing. Dans le virtuel de masse base de données, cette couche classe ces ressources en groupes et les conditionne. Après cela, elle analyse, décompose, classe et stocke les informations traitées à l'aide d'outils virtuels, pour fournir une capacité de ressources pour les requêtes de la couche d'application. [26]

1.9.4 Couche des services de base :

Cette couche est la couche la plus critique. C'est l'intermédiaire a connecté le fournisseur en amont et l'utilisateur en aval, mais aussi la clé si le la plate-forme de cloud manufacturing tierce peut fonctionner en douceur tout le temps ou non. Elle contient des services cloud de recherche, le service de combinaison cloud, la gestion des correspondances, la gestion des algorithmes, les mécanismes de tolérance aux pannes, etc. Grâce au calcul haute performance rapide et stable, il complète la planification des ressources pour rapidement répondre à la demande de service du terminal cloud. [26]

1.9.5 Couche d'interface orientée service :

Cette couche fournit principalement des interfaces de service pour demandeurs de service sous une certaine forme d'organisation. Il peut être une interface commune comme un portail Web, ou interface spéciale répondant à la demande d'une spécifique application de fabrication individuelle, industrielle ou domaine. Pendant ce temps, cette couche comprend également l'accès à la gestion de l'autorité et des paiements. [26]

1.9.6 La couche d'application :

Cette couche est adaptée aux besoins de tous les utilisateurs du terminal cloud. Tous les membres du système de fabrication cloud, dans le cadre de leur autorité, peut faire une demande par le biais de leur moyens pratiques à tout moment et n'importe où et interrogez des informations sur les portails Web, ou demande des services aux serveurs concernés par terminaux mobiles, PC terminaux ou terminaux spéciaux [26]

1.10 La relation entre le cloud manufacturing et le cloud computing :

La relation entre le cloud manufacturing et le cloud computing peut globalement être définie à partir de trois points de vue : ressource, applications et technologies : [21]

	Cloud computing	**Cloud manufacturing**
Ressources	-Ressources informatiques générales par exemple : CPU , GPU	-Ressources informatiques générales par exemple : CPU, GPU -Ressources informatiques orientées fabrication -Des ressources de fabrication pures par exemple robot
Applications	-Applications informatique générale	-Applications informatiques générales -Applications informatiques orientées Fabrication -Applications de fabrication pures
Technologies	-Technologies cloud générales axées sur les ressources informatiques	-Technologies cloud générales axées sur les ressources informatiques -Technologies de cloud manufacturing des ressources informatiques orientées -Des ressources de fabrication pures, ainsi que de nombreuses autres technologies habilitantes

TABLE 1.1 – La relation entre le cloud manufacturing et le cloud computing [21]

1.11 Conclusion

Dans ce chapitre nous avons présenté le cloud computing et le cloud manufacturing, nous avons montré pour chacun d'eux l'architecture et les caractéristiques, au final nous avons défini la relation entre eux.

Chapitre 2

Etat de l'art

2.1 Introduction :

Tout d'abord, il doit être clair que le cloud manufacturing est un concept qui émigre le concept,le modèle de fonctionnement et technologies du cloud computing à la fabrication, le cloud manufacturing est donc appelé la version de fabrication de cloud computing.[21]
Cloud computing adopte le concept de tout en tant que service (c'est-à-dire XaaS), y compris SaaS, PaaS et IaaS. Prenant le concept de cloud computing, le cloud manufacturing adopte le concept de « fabrication en tant que service » MaaS (Manufacturing as a Service), c'est-à-dire que toutes les ressources et les capacités de fabrication regrouper dans le cycle de vie du produit sont encapsulées en services. Il existe également trois modèles de service pour MaaS dans le cloud manufacturing, (MSaaS), (MPaaS)et(MIaaS).[21]
Ce chapitre aborde un état de l'art du domaine de la sélection du service CMfg en se basant sur les contraintes, suivi d'une classification de toutes les approches étudiées qui nous a permis d'une part de déduire les avantages et les inconvénients de chaque approche, et d'autre part de mieux déterminer notre problématique par rapport aux solutions étudier afin de perfectionner notre approche.

2.2 Sélection des services cloud :

La sélection du service cloud est un processus qui permet de découvrir et de sélectionner les services cloud appropriés prestataires des services capable de répondre à leurs besoins sur la base de diverses contraintes telles que : les fonctionnelles exigences, QoS et Accords de Niveau de Service, en raison de l'absence des normes exprimant ces exigences, il est difficile d'évaluer la qualité des services (provenant de différents fournisseurs) pour répondre à toutes les exigences des utilisateurs, ainsi c'est important de concevoir des procédures de sélection de service appropriées qui évaluent les services cloud et sélectionnent les services les plus adaptés. [11]

2.3 La qualité de services (QoS) :

Il existe des mesures, présentées par la qualité de service. De plus, les QoS représentent l'aspect non-fonctionnel d'un service, ces valeurs permettent la sélection des services pertinents aux demandes des utilisateurs. Nous allons présenter quelques mesures QoS :

2.3.1 Temps de réponse :

L'efficacité de la disponibilité d'un service peut être mesurée en termes du temps de réponse, c'est-à-dire dans le cas de l'IaaS, à quelle vitesse les services peuvent être mis à la disposition de l'utilisation. Par exemple, si un utilisateur demande une machine virtuelle d'un fournisseur de Cloud, puis la réponse « temps du service » représentera le temps mis par le fournisseur de Cloud pour servir cette demande. Cela inclut le provisionnement de la VM, le démarrage de la VM, l'attribution d'une adresse IP et le démarrage du déploiement de l'application. Le temps de réponse du service dépend de divers sous-facteurs tels que temps de réponse moyen, temps de réponse maximum promis par le fournisseur de services [14]
Le temps de réponse est calculé d'après la formule suivante :

$$\text{Temps de reponse} = \sum_{i}^{N} \dot{Ti} \ /n$$

Ti : Représente l'utilisateur i a demandé un service et n est le nombre total des demandes de service.

2.3.2 La fiabilité :

Reflète la façon dont un service fonctionne sans faiblesse pendant un temps et une condition donnés. Par conséquent, il est défini en fonction sur le délai moyen de faiblesse promis par le fournisseur de Cloud et échecs antérieurs accuser par les utilisateurs. Il est mesuré par : [14]

$$\text{La fiabilité} = \left(1 - \frac{Nombre\ d\ échec}{n}\right) * P_{mttf}$$

Nombre d'échec est le nombre d'utilisateurs qui ont accuser un échec dans un intervalle de temps inférieur à celui prévu par le fournisseur du cloud
N est le nombre d'utilisateur.
P mttf :représente le temps d'échec estimé qui est défini par le fournisseur.

2.3.3 La réputation :

La réputation d'un plan d'exécution est la moyenne des réputations des services qui participent dans le plan. Il mesure la fiabilité du service sur la base de l'expérience de l'utilisateur lors de l'utilisation du service [28]

$$\text{Réputation} = \frac{\sum_{r=1}^{N} F_r}{N}$$

Fr : est le rang donné par le rième utilisateur pour un service.
N : est le nombre d'utilisateurs qui ont utilisé le service.

2.3.4 La disponibilité :

La disponibilité est le pourcentage de temps auquel un client peut accéder au service. Il est donné par [14] :

$$\text{Disponibilité} = \frac{MTBF}{MTBF+MTTR}$$

MTBF se réfère au temps moyen de service avant l'état d'échec,
MTTR est le temps moyen de réparation.

2.3.5 Pertinence :

Pertinence est définie comme le degré de satisfaction des exigences d'un client par un fournisseur de Cloud. Il y a deux sous cas avant de pouvoir définir la Pertinence. Premièrement, si après avoir filtré le Cloud fournisseurs, il existe plusieurs fournisseurs Cloud qui satisfont toutes les exigences essentielles et non essentielles du client, alors tous conviennent. Sinon, si le filtrage aboutit à une liste de fournisseur de service vide puis les fournisseurs qui satisfont les caractéristiques essentielles sont choisis [14]

$$\text{Pertinence} = \frac{Nombr\ e\ de\ caractéristique\ non\ essentielles\ fournies\ par\ le\ service}{Nombre\ de\ caractéristique\ non\ essentielles\ requises\ par\ le\ client}$$

$$\text{Pertinence} = \begin{cases} 1\ , & Si\ toutes\ les\ caractéristiques\ sont\ satisfaites \\ 0\ , & dans\ le\ cas\ inverse \end{cases}$$

2.3.6 Coût :

Est le montant qui doit payer le demandeur pour exécuter un service. [28]

2.4 Le problème de selection de services dans le cloud :

L'un des problèmes cruciaux des systèmes CMfg est que de plus en plus d'offres de services CMfg peuvent offrir les mêmes fonctionnalités mais diffèrent par leurs performances. Pour s'assurer que le service CMfg répond aux exigences des tâches, la sélection optimale du service CMfg devient un problème de plus en plus important. Notre solution contribue à la phase

de sélection des services CMfg et propose un nouvel algorithme de sélection augmenté des contraintes de temps tout en considérant les dépendances entre elles. Dans un premier temps, nous proposons un algorithme de sélection par score de service CMfg qui classent tous les candidats et trouvent la combinaison proche de l'optimale des services CMfg en évitant une recherche exhaustive dans tous les plans d'exécution, comme montré l'article [17] .

2.5 Sélection de services composites :

Avec l'introduction de la technologie cloud dans la fabrication, les ressources matérielles et logicielles peuvent être dynamiquement encapsulées en tant que services cloud. Pour simplifier l'illustration, nous utilisons "La conception et la production d'une voiture personnalisée dans un cloud environnement" comme cas pour décrire le processus de composition du service de la soumission de la tâche à la fin. Depuis la fabrication mission est complexe et implique des centaines de sous-systèmes et des milliers de pièces, nous utilisons un modèle simplifié, comme indiqué dans Fig 2.1 [46], la mission de fabrication simplifiée est divisée en plusieurs sous-tâches : (1) ST1 : Analyse technique et théorique, (2) ST2 : Modélisation CAO, (3) ST3 : Simulation de vérification, (4) ST4 : Traitement du cadre, (5) ST5 : Traitement de la roue, (6) ST6 : Traitement du moteur, (7) ST7 : Production de pièces, (8) ST8 :Assemblage des composants, (9) ST9 : Assemblage final du véhicule, (10) ST10 : Post-traitement, et (11)ST11 : Expérimentation et livraison Au cours du processus, la mise en œuvre de chaque sous-tâche à des exigences particulières, par exemple : la tâche de modélisation CAO (ST2) a sa fonctionnalité et ses exigences de qualité de service par exemple : les types de données, les paramètres de conception, le prix de l'offre, les tolérances de modélisation, Dimensions, etc, la plate-forme CMfg affine les services avec des propriétés de fonction similaires mais différentes en termes de qualité de service, tels que SolidWorks, Master-CAM, etc , de plus, ST7 peut être exécuté par de nombreux services d'usinage par exemple la vitesse, la précision, la fiabilité, la similarité des fonctions, le crédit, le temps et le coût, en outre, chaque type de service a ses spécifications spéciales pour le traitement d'un objet, telles que le type de matériau, la dureté, la coupe taille, trempe ou traitement thermique cyclique. Dans ce cas, la mise en correspondance des services est effectuée en premier pour générer le service candidat, ensembles, puis comment choisir le plus approprié dans chaque ensemble et former le service composite pose un défi de taille pour un optimiseur, en particulier lorsque des candidats de service à grande échelle sont fournis

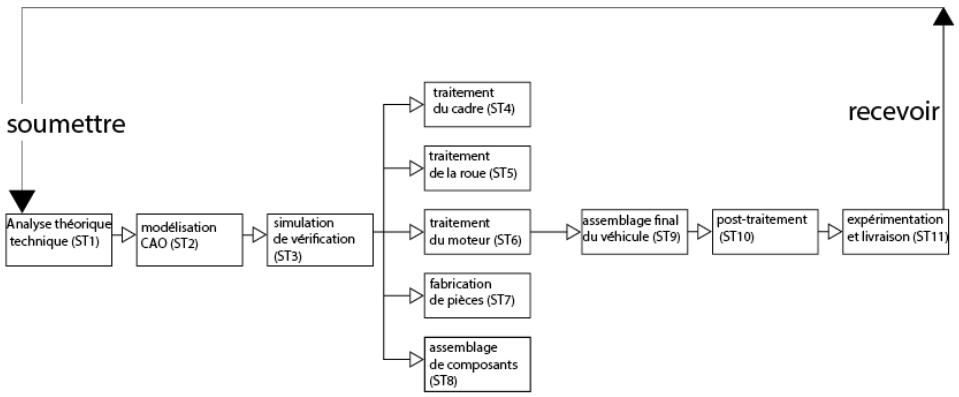


FIGURE 2.1 – La conception et la production d'une voiture personnalisée dans le cloud [46]

2.6 Structure d'exécution des services :

II existe 3 structures d'exécution de service :

2.6.1 Parallèle :

Plusieurs services sont exécutés en même temps. [4]

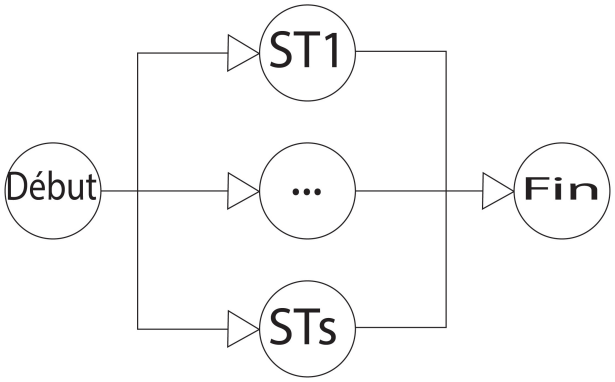


FIGURE 2.2 – Structure d'exécution de service : Parallèle

2.6.2 Séquentielle :

Les services sont exécutés l'un après l'autre. [4]

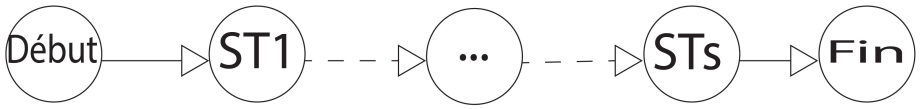


FIGURE 2.3 – Structure d'exécution de service :Séquentielle

2.6.3 Boucle :

Un ou plusieurs services sont exécutés de manière répétitive N fois. [4]

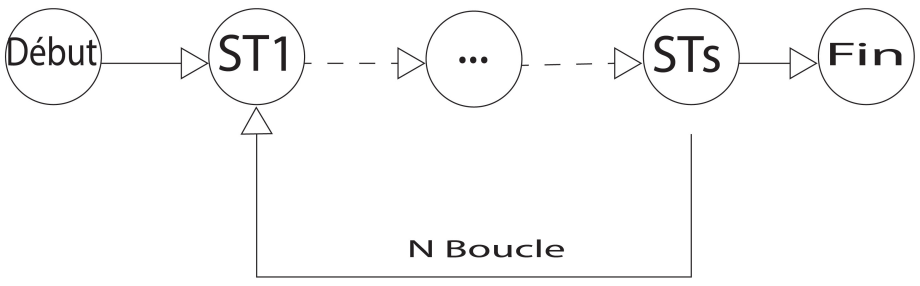


FIGURE 2.4 – Structure d'exécution de service :Boucle

2.6.4 Sélective :

Sélectionnez un service parmi plusieurs services. [4]

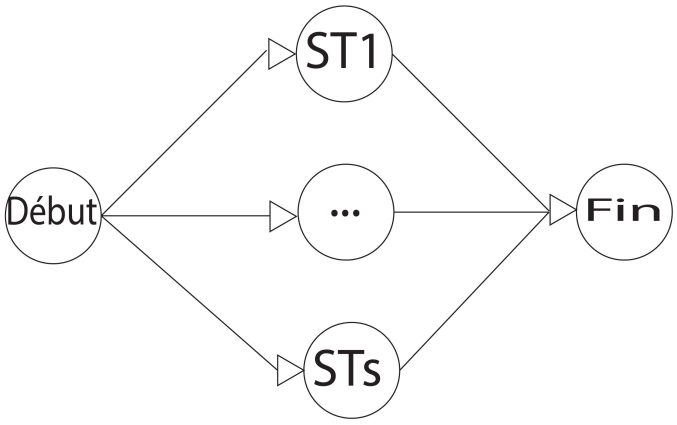


FIGURE 2.5 – Structure d'exécution de service :Sélective

2.7 Cycle de vie de composition de services :

2.7.0.1 Phase de décomposition de la tâche :

La demande de tâche des utilisateurs est décomposée en sous-tâches et les services sont construits par correspondance de fonction et de flux. [42]

2.7.0.2 Phase de sélection de services :

Les services composites abstraits sont cartographiés en services de ressources concrets, et un plan de composition de services optimal est sélectionné. [42]

2.7.0.3 Phase de composition des services :

La composition optimale des services est exécuté en liant et en invoquant le service de ressources concret, et le résultat final est sorti. [42]

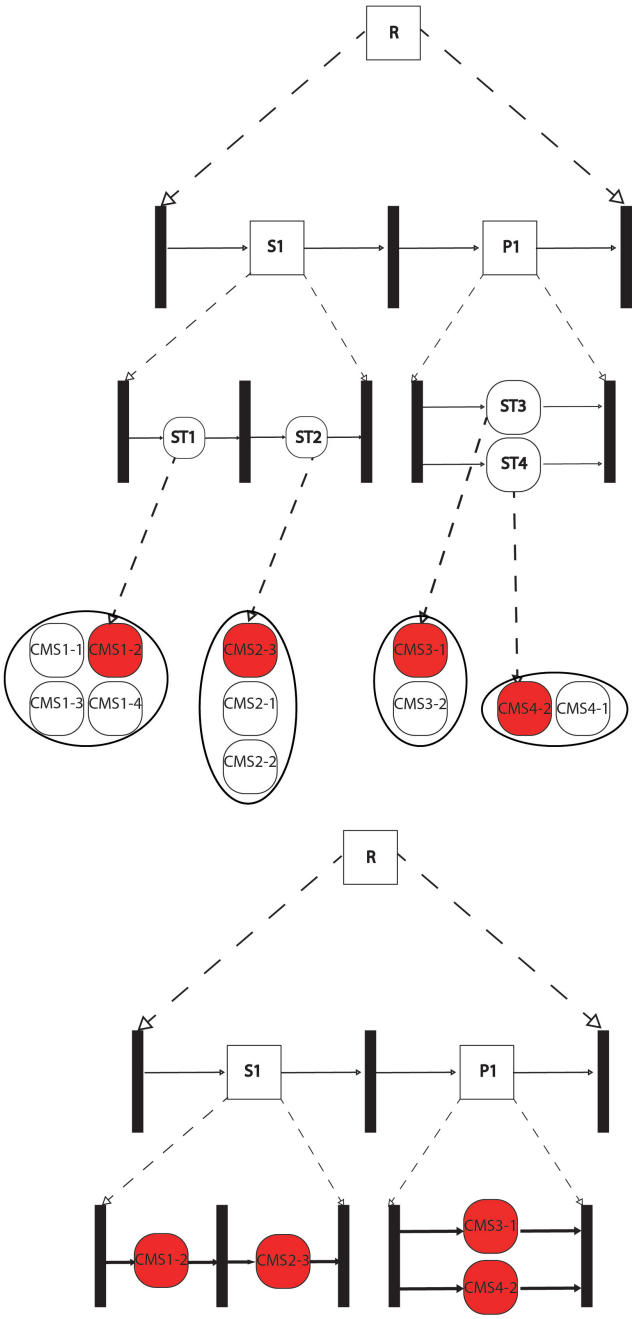


FIGURE 2.6 – Cycle de vie de composition de services

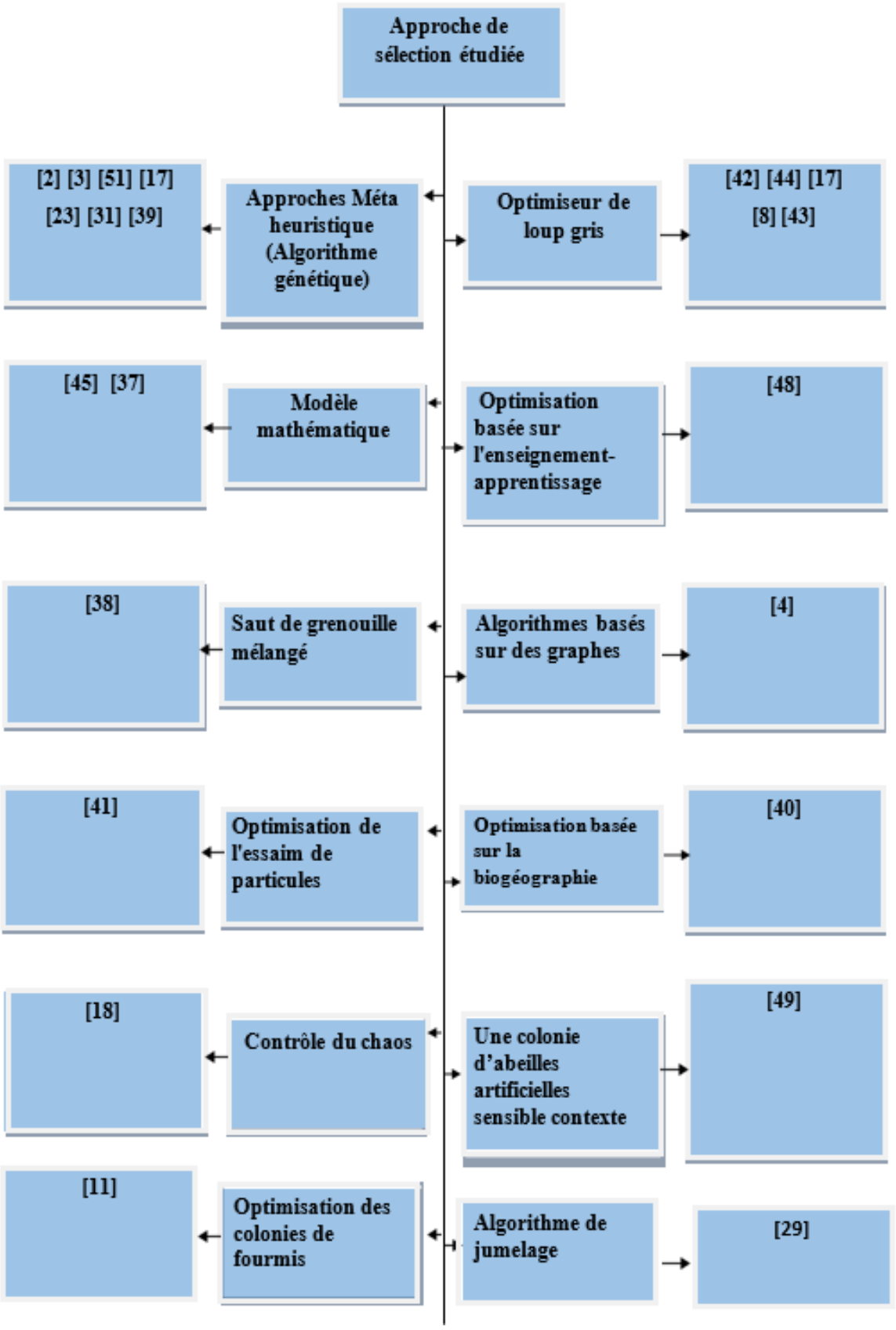
2.8 Les approches de sélection :

2.8.1 Introduction :

Cet section permet de faire un état de l'art du domaine de la sélection des services cloud Mfg et d'identifier leurs limites. Cette partie de l'état de l'art est dédiée à l'étude des travaux relatifs à la problématique de selection.

2.8.2 Classification et présentation des approches de sélections :

Cet section permet d'élaborer un état de l'art qui contient les nouvelles approches de sélection de service CMfg augmenté des différents types de contraintes. Suivi d'une classification de toutes les approches étudiées qui nous a permis d'une part de dégager les avantages et les inconvénients de chaque approche, et d'autre part de mieux positionner notre problématique par rapport aux solution étudier afin d'améliorer notre approche.
Nous discutons dans cette section les travaux liés au domaine de sélection des services Web impliqués dans la composition. Plusieurs solutions sont utilisées pour sélectionner la meilleure composition de service. La classification suivante résume les approches de sélection étudieés.


Approche de sélection étudiée
Approches Méta heuristique (Algorithme génétique)
[2] [3] [51] [17] [23] [31] [39]
Optimiseur de loup gris
[42] [44] [17] [8] [43]
Modèle mathématique
[45] [37]
Optimisation basée sur l'enseignement-apprentissage
[48]
Saut de grenouille mélangé
[38]
Algorithmes basés sur des graphes
[4]
Optimisation de l'essaim de particules
[41]
Optimisation basée sur la biogéographie
[40]
Contrôle du chaos
[18]
Une colonie d'abeilles artificielles sensible contexte
[49]
Optimisation des colonies de fourmis
[11]
Algorithme de jumelage
[29]

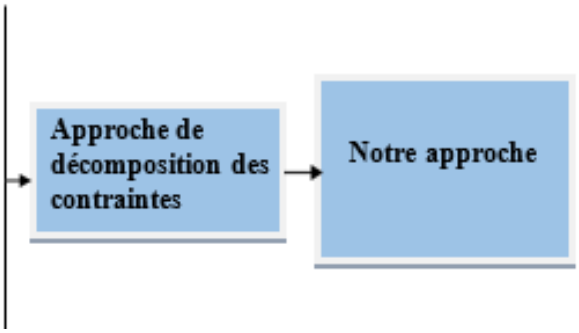


FIGURE 2.7 – Classification des approches de sélection étudiées

2.9 Les Algorithmes de sélection :

2.9.1 Algorithme génétique :

Les algorithmes génétiques sont des approches de recherche heuristiques applicables à une large gamme de problème d'optimisation. Cette flexibilité les rend attrayants pour de nombreux problèmes d'optimisation en pratique. L'évolution est la base des algorithmes génétiques. Ils ont développé à des structures complexes qui permettent la survie dans différents types d'environnements.
La solution proposé dans [2] présente un modèle mathématique qui sélectionne un ensemble optimal de composition de services de fabrication et de logistique afin de réduire les coûts d'exploitation et de logistique dans une perspective opérationnelle tout en répondant à une nouvelle idée d'entropie cloud configurée de la logistique et les fournisseurs d'opérations.
La solution proposé dans [1] remet en question le mécanisme centralisé du problème de composition de services et présente une nouvelle plate-forme intitulée Blockchain-based service composition model (Block-SC) basée sur la technologie blockchain. Les capacités de la plate-forme proposée sont remarquables de deux points de vue ; tout d'abord, il fournit un mécanisme de collaboration des fournisseurs de services avec une approche orientée services et à partir de la deuxième perspective, l'optimalité du problème de composition de service est profondément affectée en tenant compte du comportement dynamique de cloud manufacturing.
L'article [20] vise les problèmes dans lesquels il existe une collocation entre les services et les tâches de fabrication. Premièrement, les expressions mathématiques du degré de collocation des services, l'entropie de la composition et leurs facteurs d'influence liés à la composition du service sont analysés. Ensuite, un service mathématique multi objectif de composition et d'optimisation de cloud manufacturing est établi.
Dans l'article [48] un nouveau schéma de composition des services de fabrication nommé Multi-Batch Subtasks Parallel-Hybrid Execution Cloud Service Composition for Cloud Manufacturing (MBSPHE-CSCCM) est proposée, une telle composition est l'un des problèmes d'optimisation

de combinaison les plus difficiles avec une complexité NP-difficile. Pour régler le problème, une nouvelle méthode d'optimisation proposé nommée évolution différentielle hybride améliorée et optimisée

Dans l'article [15] un modèle multi-objectif est d'abord construit avec des priorités objectives différentes. Ensuite, une méthode en deux phases basée sur l'ordre de priorité de satisfaction0 est conçue pour résoudre ce problème.

L'article [35] propose une stratégie de sélection des ressources de fabrication basée sur un algorithme génétique distribué pour optimisation de ressource de fabrication combinatoire (MROC) dans CMfg. ils ont divisé le DGA en plusieurs sections et optimisé le processus, qui non seulement garantissait la vitesse de l'algorithme, mais aussi élargi la plage de recherche et améliore la précision

L'étude [27] présente un nouveau Mode Fabricants aux Utilisateurs (FAU) pour la fabrication dans le cloud, visant à résoudre le problème de base de composition du service de fabrication de sélection optimale (MSCOS). Le mode FAU étend les zones de service et améliore ses capacités d'allocation dynamique optimale des ressources par un gestion et un fonctionnement efficaces et flexibles des services.

2.9.2 Optimiseur de loup gris (GWO) :

C'est une méthode récente d'optimisation intelligente inspirée du comportement de chasse des loups gris. Dans l'algorithme GWO, le paramètre 'alpha' est diminué de 2 à 0 pour équilibrer respectivement l'exploitation et l'exploration. Un nouveau paramètre variant dans le temps d'une diminution linéaire est utilisé pour améliorer les performances de l'algorithme GWO. Afin d'améliorer la convergence globale, lors de la génération de la première population, la méthode des bons points est utilisée. [22]

L'article [38] étudie le problème du SCOS basé sur la QoS et la consommation d'énergie (QoS-EnCon). Premièrement, le modèle de composition de services multi-objectifs a été établi l'évaluation de la QoS et de la consommation d'énergie ont été étudiées.

L'article [39] présente un optimiseur de loup gris multi-objectif amélioré (EMOGWO) pour le problème de composition de services multi-objectifs et de sélection optimale (MO-SCOS) dans le cloud manufacturing, dans laquelle la qualité de service et la consommation d'énergie sont considérées à partir des perspectives de fabrication durable.

L'article [40] propose un modèle de service économe en énergie de composition et de sélection optimale (EA-SCOS) pour garantir une haute qualité et une faible consommation d'énergie pendant les tâches. Le modèle mathématique de consommation d'énergie basé sur la qualité de service (QoS) est établi. Afin de résoudre efficacement le problème EA-SCOS, un algorithme nommé optimiseur de loup gris (GWO), est introduit.

Dans l'article [15] la préférence linguistique est considérée, et un nouveau modèle en deux

phases basé sur un degré satisfaisant est proposé .
L'étude [6] propose une nouvelle approche hybride basée sur l'algorithme d'optimisation le loup gris récemment développé (GWO) et opérateurs évolutionnaires de l'algorithme génétique. Le croisement intégré et les opérateurs de mutation assurent une double fonctionnalité : ils permettent d'adapter la structure continue de GWO à un problème combinatoire .

2.9.3 Modèle mathématique :

La modélisation mathématique, qui consiste à passer d'une situation réelle à un modèle, travailler avec ce modèle et l'utiliser pour comprendre et développer ou résoudre les problèmes du monde réel, et depuis longtemps été une caractéristique des cours d'ingénierie, de science et la technologie. [10]
Dans l'article [41] le schéma de sélection optimal de la composition du service est proposé en utilisant le gris relationnel méthode d'analyse (GM) et la validité du schéma de sélection optimal est vérifiée par un exemple de fabrication de moule.
Dans l'article [33] un modèle de composition de service sensible à la corrélation (CASC) a été établi en considérant deux types de corrélations de service, c'est-à-dire la corrélation orientée vers la compossibilité et la corrélation orientée vers la qualité. La corrélation orientée vers la compossibilité a été mappée à un ensemble de contraintes, tandis que la corrélation orientée qualité était quantifiée dans les objectifs du modèle de service de composition.

2.9.4 Optimisation basée sur l'enseignement-apprentissage :

Cette méthode agit sur l'effet de l'influence d'un enseignant sur les apprenants. Comme d'autres algorithmes inspirés de la nature, cette optimisation est également une méthode et utilise une population de solutions pour passer à la solution globale.
Afin de construire une composition de service innovante et stable. L'article [44] propose un nouveau modèle composite, qui utilise une sélection à double sens selon leur coopération pour recommander les partenaires les plus appropriés.

2.9.5 Algorithme de saut de grenouille mélangé (SFLA) :

Une méta-heuristique mimétique appelée algorithme SFLA (shuffled frog-leaping) a été développée pour résoudre des problèmes d'optimisation combinatoire. La SFLA est une recherche coopérative basée sur la population métaphore inspirée de la mimétique naturelle. L'algorithme contient des éléments de recherche locale et globale d'échange d'informations
Dans l'article [34] tout d'abord, un modèle de regroupement et de recherche d'un service d'externalisation web basé sur le Langage Web d'ontologie pour les services (OWL-S) et un réseau de neurones artificiels (ANN) sont établi. Ensuite, un algorithme amélioré de saut de

grenouille mélangé (SFLA) est développé pour résoudre le problème de découverte des services et optimisation combinatoire des ressources d'externalisation (COOR).

2.9.6 Algorithmes basés sur des graphes :

Ces dernières années, les coupes de graphes sont devenues une technique d'optimisation puissante pour minimiser les fonctions énergétiques qui surviennent dans les problèmes de vision de bas niveau. L'importance des graphiques est dans la capture des phénomènes du monde réel, ce qui les rend des machines puissantes pour la détection d'anomalies. [5]
Dans L'article [3] une tâche est considérée comme un graphe orienté acyclique, et proposent des algorithmes basés sur des graphes pour obtenir le coût, le temps d'exécution, la qualité et la fiabilité d'une tâche ayant plusieurs modèles de composition.

2.9.7 Optimisation de l'essaim de particules :

L'optimisation des essaims de particules telle que développée par les auteurs comprend un concept très simple, et les paradigmes peuvent être implémentés en quelques lignes de code informatique. Il ne nécessite que des primitifs opérateurs mathématiques, et il est peu coûteux en termes de calcul en termes d'exigences de mémoire et de la vitesse. [19]
L'document [37] se concentre sur le processus SCOS avec des considérations de durabilité au niveau opérationnel de CMfg. Plus précisément, il opérationnalise la méthode d'évaluation d'auto-organisatrice Carte(SoM) en termes de rentabilité économique, environnementale et les aspects sociaux et résout le problème SCOS dans un tel nouveau contexte.

2.9.8 Algorithme d'optimisation basée sur la biogéographie :

Dans l'article [36] un nouveau problème d'optimisation multi-objectifs basé sur la qualité de service des tâches urgentes de CMfg est présentée, et deux méthodes de recomposition de service sont proposées.

2.9.9 Un algorithme optimal de contrôle du chaos :

L'article [16] étudie le problème de la sélection et la composition optimale des services cloud (SCOS) dans le CMfg. Un nouvel algorithme optimal de contrôle du chaos (CCOA) est conçu pour résoudre le problème du SCOS.

2.9.10 Une colonie d'abeilles artificielles sensibles au contexte :

Le papier [45] propose d'abord une colonie d'abeilles artificielles sensibles au contexte (caABC) algorithme basé sur le principe de l'ABC(Artificial Bee Colony) et des fonctionnalités

de service dans l'environnement cloud. Ensuite, le caABC amélioré à évolution différentielle, est conçu pour augmenter encore les performances de recherche d'ABC.

2.9.11 Optimisation des colonies de fourmis (ACO) :

Dans l'article [9] une sélection de services et modèle d'ordonnancement est établi, avec les critères TQCS (temps, qualité, coût et service) à l'étude. Une floue théorie de la prise de décision est adoptée pour transformer les valeurs TQCS en degrés de supériorité relative. Ceci est différent de la méthode traditionnelle de pondération linéaire dans la plupart des recherches antérieures.

2.9.12 Algorithme de jumelage :

Dans l'article [25] les auteurs proposent une façon d'utiliser les modèles de Web sémantique et de qualité de service (QoS) pour effectuer la correspondance, la sélection et Composition pour répondre aux exigences des utilisateurs finaux.

2.10 Tableau comparatif :

Dans la table 2.1 nous présentons pour chaque approche de sélection les différentes points positives et négatives de chaque article :

Article	Avantage	Inconvenients
Algorithme genetique		
[2]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité).	Seule la structure séquentielle de composition est Considérée dans cette approche Moins d'attention a été accordée au problème d'optimalité.
[1]	Plus d'attention a été accordée à la Complexité du temps et aux problèmes d'optimalité. Plusieurs contraintes sont imposées dans cette approche (fiabilité, réputation, considération de qualité).	Seule la structure parallele de composition est Considérée dans cette approche.
[**?**]	Plusieurs contraintes sont imposées dans cette approche (temps, coût, degré de synergie et d'entropie de composition).	Moins d'attention a été accordée à la Complexité du temps et aux problèmes d'optimalité. Gére uniquement la composition séquentielle du service
[48]	Plusieurs contraintes sont imposées dans cette approche (temps, coût, disponibilité, réputation).	Moins d'attention a été accordée à la Complexité du temps et aux problèmes d'optimalité Seule la structure parallèle de composition est considérée dans cette approche.
[15]		Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité. Aucune contrainte n'est imposée dans cette approche

TABLE 2.1 – Tableau comparatif

[35]	Plusieurs contraintes sont imposées dans cette approche(utilité, confiance, énergie).	Seules la structure parallèle de composition est considérée dans cette approche. Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité.
[27]	Plusieurs contraintes sont imposées dans cette approche (temps, fiabilité, coût et capacité) Tous les structures de composition sont considérées dans cette approche (séquentielle, parallèle, boucle, Sélective). Plus d'attention a été accordée à la Complexité du temps et aux problèmes d'optimalité.	
Optimiseur de loup gris		
[39]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, disponibilité, fiabilité). Plus d'attention a été accordée à la Complexité du temps et aux problèmes d'optimalité.	Seule la structure parallèle de composition est considérée dans cette approche
[40]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, disponibilité, fiabilité). Plus d'attention a été accordée à la Complexité du temps et aux problèmes d'optimalité.	Seule la structure séquentielle de composition est considérée dans cette approche.

[15]	Plusieurs contraintes sont imposées dans cette approche (coût, temps).	Seule la structure séquentielle de composition est considérée dans cette approche. Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité.
[6]	Tous les structures de composition sont considérées dans cette approche Plusieurs contraintes sont imposées dans cette approche (coût, temps, disponibilité, fiabilité).	Moins d'attention a été accordée au problème d'optimalité.
Modèle mathématique		
[41]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité). Tous les structures de composition sont considérées dans cette approche.	Moins d'attention a été accordée au problème d'optimalité
[33]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, disponibilité, fiabilité). Plus d'attention a été accordée à la complexité du temps et aux problème d'optimalité.	Seule la structure séquentielle de composition est considérée dans cette approche.

Optimisation basée sur l'enseignement apprentissage		
[44]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité).	Seule la structure séquentielle de composition est considérée dans cette approche. Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité
Algorithme de saut de grenouille mélangé		
[34]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, qualité, crédibilité, énergie)	Seule la structure séquentielle de composition est considérée dans cette approche. Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité
Algorithmes basés sur des graphes		
[3]	Tous les structures de composition sont considérées dans cette approche (séquentielle, parallèle, boucle, Sélective). Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité).	Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité

Optimisation de l'essaim de particules		
[37]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité). Plus d'attention a été accordée à la complexité du temps et aux problème d'optimalité.	Seule la structure séquentielle de composition est considérée dans cette approche.
Algorithme d'optimisation basée sur la biogéographie (TBBO)		
[36]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité). Tous les structures de composition sont considérées dans cette approche (séquentielle, parallèle, boucle, Sélective).	Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité.
Un algorithme optimal de contrôle du chaos		
[16]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité). Plus d'attention a été accordée à la complexité du temps et aux problème d'optimalité	Seule la structure séquentielle de composition est considérée dans cette approche.

Une colonie d'abeilles artificielles sensibles au contexte		
[45]	Plusieurs contraintes sont imposées dans cette approche (temps, réputation, disponibilité). Plus d'attention a été accordée à la complexité du temps et aux problème d'optimalité	
Algorithme d'optimisation multi-objectifs		
[38]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité).	Seule la structure séquentielle de composition est considérée dans cette approche. Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité.
Algorithme d'allocation de capteur sensible à la QoS (Q-SAA)		

[24]	Plusieurs contraintes sont imposées dans cette approche (coût, temps, fiabilité). Plus d'attention a été accordée à la complexité du temps et aux problème d'optimalité	Seules les structures séquentielles et Parallèle de composition sont considérées dans cette approche.
Optimisation des eolonics de fourmis (ACO)		
[9]	Plusieurs contraintes sont imposées dans cette approche (temps, qualité et coût et service).	Seule la structure en boucle de composition est considérée dans cette approche. Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité
Algorithme de matchmaking		
[25]	Plusieurs contraintes sont imposées dans cette approche (disponibilité, débit, temps de réponse et coût).	Seule la structure séquentielle de composition est considérée dans cette approche. Moins d'attention a été accordée à la complexité du temps et aux problèmes d'optimalité.

2.11 Tableau de Simulation :

Dans la table 2.2 nous avons montré pour chaque article l'outil de simulation utilisé :

Article	Outil de simulation
[2]	MATLAB
[1]	N/A
[**?**]	MATLAB
[48]	N/A
[15]	N/A
[35]	Simulation SINOMA
[39]	Simulation Parote
[40]	MATLAB
[15]	N/A
[6]	MATLAB
[41]	N/A
[33]	Monto Carlo simulation
[44]	Développé par Java
[34]	Simulation avec des sous-serveurs
[3]	N/A
[37]	MATLAB
[36]	Langage de programmation C#
[16]	SOM
[45]	MATLAB
[38]	Simulation Pareto
[27]	MATLAB
[24]	MATLAB
[9]	N/A
[25]	N/A

TABLE 2.2 – Tableau de simulation

2.12 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur la composition et la sélection de services cloud dans une composition augmentée de contraintes. Plus précisément, pour chacune de ses approches nous avons mis en évidence les avantages et les inconvénients de chaque type d'approches.

Chapitre 3

APPROCHE DE SELECTION DES SERVICES CLOUD MANUFACTURING AUGMENTÉE DE CONTRAINTES TEMPORELLES :

3.1 Introduction

Dans ce chapitre, nous proposons tout d'abord une approche de sélection de services dans le cloud manufacturing augmentés des contraintes temporelles sous différentes priorités.
L'ensemble des sous-taches fait référence aux fonctions requises par la requête cloud. Le service candidat est l'instanciation de la sous-tache. La sous-tache définit la fonction de la requête cloud. Plusieurs fournisseurs peuvent fournir des services candidat équivalents avec les mêmes fonctions. Plusieurs services candidat disponibles peuvent remplir cette fonction, et ils sont équivalents et peuvent se remplacer. Le choix entre eux peut se faire à travers des caractéristiques non fonctionnelles, telles que le temps de réponse, qui est un facteur important dans le choix d'un service candidat.

3.2 Contrainte temporelle multiple avec différentes priorités dans la sélection des services CMfg

En supposant un ensemble de n classes de sous-tache, les services cloud accessibles sont regroupés en fonction de leurs attributs fonctionnels, mais ils peuvent différer en termes d'attributs non fonctionnels (temps de réponse). Notre objectif est de sélectionner un ensemble de services candidats qui répondent aux contraintes de temps imposées par les clients, les fournisseurs et la composition. Chaque service candidat un attribut de temps de réponse émis par le fournisseur de services.

Nous utilisons le vecteur*TC = {TC_1 ,. . . , TC_t ,. . . ., TC_T }* pour représenter l'ensemble des contraintes temporelle imposées dans une composition avec différentes priorités exprimées dans le vecteur de priorité *Pr[t] = { TC_H,. . . , TC_t, TC_f,. . . , TC_L }*. Ou TC_H doit avoir la plus haute priorité et TC_L doit avoir la priorité la plus basse.TC_t doit avoir une priorité plus élevée que TC_f. Chaque contrainte temporelle a un vecteur d'affectation VA_t tel que défini dans eq.3.1 et une constante *Const* pour exprimer la contrainte.

$$TC_t = \begin{cases} AV_t[i], \forall i \in \{1, n\} \\ \leq Const_t \end{cases} \tag{3.1}$$

Où

$$AV_t(i) = \{ST_1, \ldots, ST_i, \ldots, ST_n\} \begin{cases} ST_i = 1, \ Si\ ST_i\ est\ inclus\ dans\ cette\ contrainte \\ ST_i = 0, \ \ sinon \end{cases} \tag{3.2}$$

Chaque contrainte temporelle peut être allouée à un service ou à un groupe de services, qui est représenté par une matrice d'allocation. Notre objectif est de gérer un ensemble de contraintes de temps avec des priorités différentes en même temps, sans avoir à traiter chaque contrainte de temps séparément et à considérer les corrélations et dépendances possibles entre elles. Afin de pondérer l'importance des contraintes de temps, nous utilisons un vecteur de priorité, qui est un excellent moyen de hiérarchiser les contraintes de temps des sous-taches.

3.2.1 Algotithme de priorité

La figure 3.1 montre en détail l'algorithme d'attribution de priorité. Afin de donner le poids d'importance de chaque contrainte assignée aux sous-taches dépendants et indépendants.

Les valeurs des priorité sont affecté automatique par l'algorithme suivant la distance ou le degré de priorité pour chaque contrainte temporelle affecté de façon equtable et suivant le degré de priorité entre les CT.

Input : paramètres de l'algorithme

n : nombre de sous-tache

m : nombre de Contraintes temporaries

ptc[i] : {ptc [1], ptc [2], …, ptc[i]}

Vecteur contient les priorités de TC[i]

Matrice [i] [j]: {A0-0, A0-1, …, Ai-j} A= type booléenne

Output:

Matrice2 [i] [j]: {P0-0, P0-1, …, Pi-j}

Matrice contient les priorités affectées à TC pour chaque sous-tache

```
Begin
For (i=0 → m)
    Remplissage de vecteur de priorité TC
End
For (i=0 → n)
    For (j=0 → m)
        Remplissage de matrice d'affectation
    End
End
For (i=0 → n)

    For (j=0 → m)
        If (Matrice [ i] [j] == 1) then vp[j] = v[j]  //  faire un découpage et affectée les  priorités TC
        Else vp[j] = 0
    End
    Double max = 0
    For (j=0 → m)
        If (max<vp[j]) then max = vp[j]    //  recherche sur max de priorités TC
    End
     sum = 0
    For (j=0 → m)
        If (vp[j]! =0) then
        Double aa = max-vp[j];
        Double ax = Math.pow (2, aa)
        Sum= sum + (1/ax);
    End
      Double p = matrice [i][m] / sum //  calcul de P
    For (j=0 → m)
        If (vp[j] ==0) then Vp[j]=0
        Else
        If (vp[j] ==max) then vp[j]=p
        Else vp[j] = (p / Math.pow (2,max-vp[j]))
    End
End


End
For (j=0 → m)
    Matrice [ i ] [ j ] = vp[ j ]
End
```

FIGURE 3.1 – Algorithme de priorité

L'algorithme d'attribution de priorité reçoit en entrée un ensemble de sous-tâches ST et un

ensemble de contraintes temporelles TC. La matrice d'allocation "Matrice" permet d'affecter les contraintes temporelles TC au sous-tâches ST, la dernière case de chaque ligne (i) de la matrice contient la variable cpt qui calcule la somme des contraintes TC attribuées à la sous-tache (i), le vecteur ptc permet de classer contraintes temporelles de la plus haute priorité a la plus basse. Pour calculer P on passe par plusieurs étapes, tout d'abord on observe toutes les TC affectées à ST(i) et on met leurs priorités dans le vecteur vp. puis on cherche le maximum Max des priorité dans le vecteur VP. La variable p est calculé en divisant la variable cpt sur la variable sum. Ensuite on reparcoure toute la ligne (i) et pour chaque TC attribué à ST(i) reçois la valeur p divisée par 2 puissance (Max - prio de cette contrainte).
Enfin, la priorité de chaque sous-tâche est calculée et attribuée en accordant la contrainte temporelle qui lui est attribué et la distance entre chacune d'eux.

La figures 2.3 montre les étapes de l'algorithme de priorité

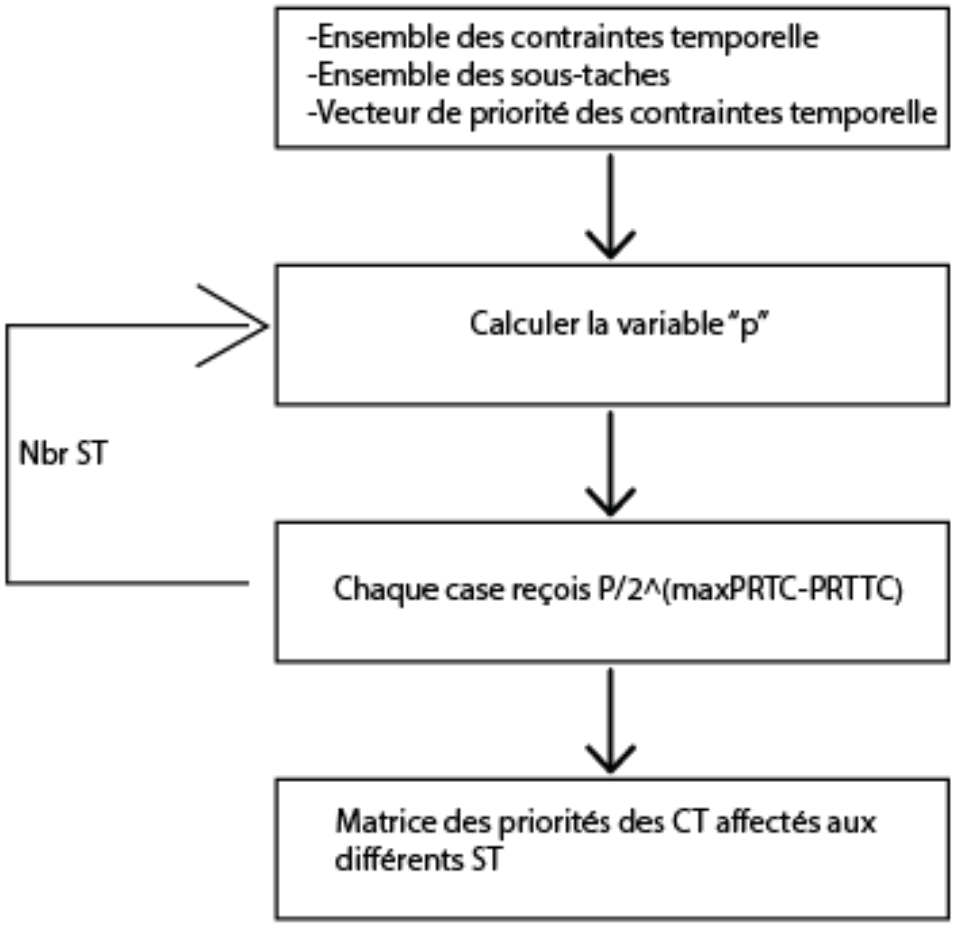


FIGURE 3.2 – étapes de l'algorithme de priorité

3.3 Example

Le tableau 3.1 montre un exemple numérique, qui contient quatre contraintes temporelles avec un vecteur de classement des priorités RPrV[t] = {TC_2,TC_1, TC_4, TC_3}ce vecteur est trié

par ordre décroissant de priorité (i.e. TC_2 doit avoir la plus haute priorité, TC_3 doit avoir la plus faible priorité et TC_1 doit avoir une priorité élevée que TC_4.) et avec des vecteurs d'affectation AV_i (TC_t) de chaque contrainte temporelle comme définit dans le tableau suivant :

	AV(TC_1)	*AV(TC_2)*	*AV(TC_3)*	*AV(TC_4)*	*S[i]*
ST1	1	0	0	1	2
ST2	1	0	1	1	3
ST3	1	1	1	1	4
ST4	0	0	1	1	2

TABLE 3.1 – Vecteurs d'allocation des contraintes temporelle affectées aux différents sous-taches.

L'algorithme de priorité est utilisé pour calculer la priorité de chaque contrainte de temps affectée aux différentes sous-tâches. C'est un excellent moyen de hiérarchiser les contraintes de temps des sous-tâches.
Le vecteur d'allocation peut définir quelles sous-tâches sont incluses dans la contrainte. Par exemple ,dans le tableau 3.1 ,la contrainte de temps TC1 est allouée aux trois sous-tâches ST1 ,ST2 ,ST3 ,car le vecteur d'allocation AVTC1 de la première contrainte de temps est égal à [1.1 ,1.0]. Le vecteur S reflète le nombre de contraintes temporelles affectées à la même sous-tâche STi ,par exemple pour la ST2 on a : ENS [2 ,j] = [TC1 ,TC3 ,TC4 ,null] telle que définie dans le tableau 3.2.

ENS[i,j]	*1*	*2*	*3*	*4*
1	*TC_1*	*TC_4*	*nulle*	*nulle*
2	*TC_1*	*TC_3*	*TC_4*	*nulle*
3	*TC_1*	*TC_2*	*TC_3*	*TC_4*
4	*TC_3*	*TC_4*	*nulle*	*nulle*

TABLE 3.2 – Matrice de Dépendances entre contraintes temporelles

ReturnHighestPriority (TC_1, TC4) = TC_1
ReturnHighestPriority (TC_1, TC3, TC4) = TC_1
ReturnHighestPriority (TC_1, TC_2, TC3, TC4) = TC_2
ReturnHighestPriority (TC_3, TC4) = TC_4

Basé sur la matrice de dépendances du tableau 3.2, nous obtenons la matrice des priorités des contraintes entre les sous tâches pour chaque contrainte dans le tableau 3.3.

	Pr(TC_1)	Pr (TC_2)	Pr (TC_3)	Pr (TC_4)	$\sum$
ST1	1.33	0	0	0.66	2
ST2	1.71	0	0.42	0.85	3
ST3	1.06	2.13	0.26	0.53	4
ST4	0	0	0.66	1.33	2

TABLE 3.3 – Matrice de priorité des contraintes affecté aux diffèrent sous-tache.

3.4 L'approche de sélection des services augmentés de contraintes temporelles sous différentes priorités

Afin d'associer le service le plus pertinent à chaque sous-tache, nous calculons d'abord les contraintes locales de chaque sous-tache. Ces contraintes locales sont calculées sur la base du temps de réponse d'un service candidat d'un ensemble de contraintes de temps sous différentes priorités.

Plusieurs méthodes de sélection de services supposent une valeur de temps de réponse unique, dont la plupart visent à obtenir une valeur de temps de réponse locale unique pour la sélection de services, mais les prévisions de temps de réponse ne peuvent jamais être précises. Pour cette raison, la sélection des services candidats ne peut pas être la meilleure. Pour résoudre ce problème, nous utilisons une plage de valeurs pour représenter le temps de réponse maximum et minimum.

Le calcul des contraintes locales de chaque sous-tache doit tenir compte de la structure combinée et de la plage de temps de réponse de tous les services condidat en même temps. À cette fin, nous utilisons le vecteur RTi-j = {MinRTi-j, MaxRTi-j} pour représenter le temps de réponse maximum et minimum de tous les les services condidats.

Notre approche de sélection de service candidat augmenté des contraintes de temps sous différentes priorités peut être formellement exprimées comme suit :

Un ensemble n de sous-tache.
Tache T={ ST_1 , ST_2 ,. . . , ST_n }.

Un ensemble AS de n sous-taches dans un service composite.
Asi={As_1 , As_2 ,. . . , As_i,}.

Un ensemble CS de services candidat pour chaque sous-tache :
Csi-j= { $Cs1_1$, $Cs1_2$,. . . ,Cs_1-C_1 , $Cs2_1$,. . . , Cs_2-C_2 ,. . . , Csi_j ,. . . ,Cs_i-C_j }
Où {Csi_1, Csi_2... Cs_i-C_1} sont des services fonctionnellement équivalents de sous-tache index i.

Ensemble de temps de réponse minimum des services candidats :
MinRTi- j= {MinRT1-1,. . . ,MinRT1-c1,. . . ,MinRTi-ci,. . . ,. . . , MinRTn-cn}.

Ensemble de temps de réponse maximum des services candidats :
MaxRT$_{i-j}$*={MaxRT*$_{1-1}$*,. . . ,MaxRT*$_{1-c1}$*,. . . ,MaxRT*$_{i-ci}$*,. . . ,MaxRT*$_{n-cn}$*}.*

Vecteur de priorité qui pondère l'importance des contraintes temporelles comme définies dans la dernière section *Pr[t] = { TC*$_H$ *,. . . , TC*$_t$*, TC*$_f$*,. . . , TC*$_L$ *}*

Où $TC_t = \begin{cases} VA_t[i], \forall i \in \{1, n\} \\ \leq Const_t \end{cases}$

Chaque contrainte temporelle a un vecteur d'allocation *AV*$_t$

$$AV_t\,(i) = \{AS_1,\ \ldots, AS_i, \ldots, AS_n\}\ /\ \begin{cases} AS_i = 1,\ Si\ AS_i\ est\ inclus\ dans\ cette\ contrainte \\ AS_i = 0,\ \ sinon \end{cases} \tag{3.3}$$

Premièrement, chaque temps de réponse des services candidats est transformé en valeur en le comparant à la valeur maximale du temps de réponse pour chaque sous-tache. La valuation du temps de réponse VMaxRT des services candidat sont transformés en quantification uniforme est calculé dans l'équation 3.4 :

$$VMaxRT_{i-j}\ = \frac{MaxRTi-j}{MaxRT}\ \ \forall i \in \{1, n\}\,,\ \forall j \in \{1, c_i\} \tag{3.4}$$

Où

$$\textit{MaxRT = MAX[MaxRT}_{i-j}\textit{]}, \forall i \in \{1, n\}\,,\ \forall j \in \{1, c_i\}. \tag{3.5}$$

MaxRT est le maximum temps de réponse de tous les services candidats.
$AVG\,VRT_i$ Indique la valeur moyenne de $VMaxRT_i - j$ pour chaque sous-tache

$$AVG\,VRT_i = \frac{\sum_{j=1}^{c_i} VMaxRT_{i-j}}{c_i},\ \forall i \in \{1, n\} \tag{3.6}$$

De plus, chaque valeur constante est quantifiee en comparant chaque valeur constante de chaque contrainte temporelle avec la somme du temps de réponse maximum dans chaque sous tache. La quantification de la contrainte temporelle V (TC) est calculée dans l'équation 3.7 :

$$V\left(TC_t\right) = \frac{Const_t}{maxTC}. \tag{3.7}$$

Où $maxTC = \sum_{i=1}^{n} MAX[MaxRT_{i-j}\,],\ \forall j \in \{1, c_i\}$.
MaxTC est la somme des maximums temps de réponse de tous les services candidats de chaque sous-tache.

$$\forall t \in \{1, T\}\,,\ VRT_{i-j}TC_t = \frac{AV_t\left(i\right) * V\left(TC_t\right) * VMaxRTi - j}{\sum_{i=1}^{n} AV_t\left(i\right) * AVG\,VRT_i}.\forall i \in \{1, n\}\,,\ \forall j \in \{1, c_i\}\,. \tag{3.8}$$

Ensuite, pour chaque contrainte temporelle, la valeur $VRT_{i-j}TC_t$ est calculée pour exprimer l'valuation du temps de réponse augmentée des contraintes temporelles avec différents priorités en utilisant Eq. 3.8, qui calcule la valeur $VRT_{i-j}TC_t$ pour chaque contrainte.aprés avoir obtenu une évaluation du temps de réponse pour chaque contrainte temporelle.$VRT_{i-j}TC_t$ et en considérant la valeur de priorité de chaque contrainte, nous pouvons construire une fonction de priorité pour indiquer la valuation du temps de réponse.

Pr_t Représente les priorités pour chaque contrainte temporelle

$$PrVRT_{i-j} = \frac{\sum_{t=1}^{T} VRT_{i-j}TC_t * Pr_t}{S[i]}$$

.$\forall i \in \{1, n\}\,,\ \forall j \in \{1, c_i\}\,.$

$$S\left[i\right] = \sum_{t=1}^{T} AV_t[i]$$

$AVG\ Pr_i$ Indique la valeur moyenne de $PrVRT_{i-j}$pour chaque classe de service.

$$AVG\ Pr_i = \frac{\sum_{j=1}^{c_i} PrVRT_{i-j}}{c_i},\ \forall i \in \{1, n\}\,. \tag{3.9}$$

Enfin, la valeur de contrainte locale de chaque sous tache est calculée en multipliant chaque valeur de $AVG\ Pr_i$ par la valeur de maxTC définie dans eq.3.7.

Eq.3.10 illustre le calcul de ces contraintes locales.

$$LC\left[i\right] \approx AVG\ Pr_i * maxTC \tag{3.10}$$

Après avoir calculé les contraintes locales de chaque sous tache, la dernière étape de notre solution est de trier les services candidats et à choisir la meilleure sélection. Ces contraintes locales sont calculées de manière à satisfaire toutes les contraintes de temps. La phase de sélection est complétée par l'algorithme de sélection défini dans la section suivante.

3.5 Algorithme de sélection de service avec contrainte de temps prioritaire (PTCCSSA) (PRIORITISED TIME CONSTRAINT CMFG SERVICE SELECTION ALGORITHM) :

Lors de la sélection, il est important d'estimer le temps locale de chaque sous tache pour s'assurer que la combinaison des services sélectionnés peut répondre aux exigences de temps de l'utilisateur final. La contrainte de temps n'étant pas modifiable à l'exécution, il est nécessaire de choisir la meilleure combinaison lors de la phase de sélection.
Après avoir expliqué comment calculer les contraintes locales pour chaque sous-tache, nous présentons l'algorithme de classification et de sélection PTCCSSA dans cette section. La figure 3.2 résume l'algorithme PTCCSSA de notre méthode de sélection de services, qui sélectionne le meilleur service après avoir classé des services candidats. Les services consistants sont ceux qui satisfont à toutes leurs contraintes respectives.
La vérification de la cohérence temporelle dans la composition des services cloud ne doit pas seulement être effectuée au moment de la selection, mais également au moment de l'exécution pour garantir que la combinaison peut être exécutée efficacement.

3.5.1 Cohérence lors de la selection :

Lors de la selection, les contraintes de temps auront un impact partiel ou total sur la collaboration des services cloud. Selon l'impact possible de la cohérence temporelle, nous considérons trois types de cohérence : la cohérence temporelle totale, la cohérence temporelle partielle et l'incohérence temporelle.

Après avoir calculer les contraintes locales de tout les sous-taches, si chaque contrainte temporelle peut être satisfaite par rapport à la contrainte locale en toutes circonstances, alors cette contrainte temporelle est totalement cohérente. De plus, si dans certains cas chaque contrainte temporelle peut satisfaire la contrainte locale, alors cette contrainte temporelle est partiellement cohérente. Enfin, si chaque contrainte temporelle ne peut pas satisfaire la contrainte locale dans tous les cas, alors cette contrainte temporelle est incohérente.

3.5.2 Cohérence lors de l'exécution

Au moment de l'exécution, nous devons vérifier la cohérence des contraintes de temps pour s'assurer que la combinaison peut être exécutée efficacement. Nous considérons deux types de cohérence : la cohérence temporelle et l'incohérence temporelle. Dans la phase d'exécution du service, si chaque contrainte temporelle est satisfaite par rapport au temps d'exécution réel, alors cette contrainte temporelle est cohérente, et dans les autres cas elle est incohérente.

Input : paramètres de l'algorithme

Ensemble des sous-taches STi = {ST1, ST2,..., STi}

Ensemble des services candidats SCi-j = {SC0-0, SC0-1,..., SCi-j}

Valeur min du temps de réponse des services candidats

MinRTi-j= {MinRT1-1, MinRT1-2,..., MinRTij}

Valeur max du temps de réponse des services candidats

MaxRTi-j= {MaxRT1-1, MaxRT1-2,..., MaxRTij}

TC = {TC1, TC2,..., TC i} Constant

MAt (i,j) = {b0-0, b0-1,..., bi-j}, b = type booléenne

priorité de chaque contrainte de temps affectée à différents sous-taches

Pri (i, t) = {Pri1, Pri2,.... Pri i} $\forall i \in \{1, n\}$, $\forall t \in \{1, T\}$

Output : la classification de tous les services et la sélection de meilleur service [i] pour chaque sous-tache

Begin

For *(i=1 → n)*

Compute LC[i] using eq 3.4 to eq 3.10

End

For *(i=1 → n)*

For *(j=1 → Cj)*

Diff [j] = MaxRTij – LC[i]

If MinRTij ≤ LC[i] & MaxRTij ≤ **then Return** total temporal consistency [i, j]

If MinRTij ≤ LC[i] & MaxRTij ≥ **then Return** total temporal consistency [i, j]

If MinRTij > LC[i] **then Return** temporal inconsistency [i, j]

End

Order and rink Diff [j]

Best [i] = min [Diff[j]]

Return Best [i]

End

End

FIGURE 3.3 – PTCCSSA algorithme

En d'autres termes, l'algorithme prend en entrée un ensemble de temps de réponse pour chaque service candidat et un ensemble de contraintes temporelles de priorité croissante. L'algorithme PTCCSSA vérifie si les contrainte globale "TC" peut être respectée sur la base de la limite de contrainte locale de chaque sous-tache calculée à l'aide des éq.3.4 à éq.3.10.

L'algorithme calcule la valeur Diff, qui représente la différence entre le temps de réponse maximum de chaque service candidat la valeur LC (contrainte locale) associée à chaque classe. Le choix du meilleur choix est associé à la valeur minimale de la différence Diff. Si la valeur de la contrainte locale est comprise entre le temps de réponse minimum et le temps de réponse

maximum du service candidat MinRT<LC<MaxRT, la contrainte LC peut être cohérente, le résultat est donc une cohérence temporelle partielle. Si la valeur de la contrainte locale est supérieure au temps de réponse maximal LC>MaxRT du service candidat, la contrainte LC est cohérente, le résultat est donc une cohérence temporelle totale. Enfin, si la valeur de la contrainte locale est inférieure au temps de réponse minimum du service concretLC <MinRT, alors la contrainte LC est incohérente, donc le résultat est une incohérence temporelle.

Dans chaque classe, chaque service candidat est classé selon sa distance par rapport aux contraintes locales de chaque sous-tache. Le service le moins éloigné de ces contraintes locales aura un meilleur score de classement. L'algorithme PTCCSSA renvoie un score de classement pour chaque service candidat et sélectionne une combinaison de services proche de la meilleure, en tenant compte de toutes les contraintes de temps imposées. L'algorithme PTCCSSA proposé renvoie le score de classement de chaque service candidat et sélectionne une meilleure combinaison de services qui ne viole aucune contrainte de temps imposée.

3.6 Évaluation de la complexité temporelle

Afin d'analyser notre algorithme de sélection, nous utilisons les trois paramètres suivants pour évaluer la complexité des différentes étapes de sélection. En supposant qu'il existe n sous-tache, chaque sous-tache a m services candidat et T contraintes temporelles.

3.6.1 Assignation des priorités :

Cette étape gère également un ensemble de contraintes de temps avec des priorités différentes. Premièrement, pour chaque service abstrait, le nombre de niveau calculé est T. Par conséquent, la complexité de la première étape est O(n*T).

3.6.2 Sélection de service :

Cette étape permet de classer et de sélectionner la meilleure combinaison de services satisfaisant un ensemble de contraintes T. Dans l'algorithme PTCCSSA, la première boucle utilise un ensemble d'équations pour calculer les contraintes locales de chaque sous-tache. Eq3, Eq4, Eq5, Eq7 et Eq11 peuvent être exécutés au plus (n * m) fois, et Eq6 peut être exécuté au plus T fois et Eq8, Eq9 peuvent être exécutés au plus (n * m * T) fois. Eq10 peut être exécuté au plus (n * T) fois. Eq12 peut être exécuté jusqu'à (n) fois. Par conséquent, la première boucle peut être exécutée jusqu'à n * [(5 + 2T) m + (T + 1)] fois.

La dernière boucle de l'algorithme PTCCSSA pour le calcul de la valeur (Diff) peut être exécutée au plus (n * m) fois. Les fonctions de tri peuvent être exécutées au plus (m) fois, et la meilleure sélection peut être exécutée au plus (n * m) fois. Par conséquent, la dernière boucle peut être exécutée jusqu'à 2m * (n + 1) fois. Par conséquent, la complexité de la deuxième étape est O(n * m * T).

D'après la discussion précédente, la complexité de la méthode proposée est O (n * m * T).

3.7 Example

Pour illustrer nos idées, nous considérons un exemple de composition de service, qui a trois sous-teches consécutifs, et chaque sous-teche ASi a trois services candidats. Chaque service condidat CSj a deux nombres représentant la plage de valeurs de temps de réponse, comme indiqué dans le tableau 3.5 (par exemple, la valeur de plage de CS11 est comprise entre 10 et 50 unités de temps).

ST	CS	MInRT	MaxRT	VA1	VA2	VA3	pri1	pri2	pri3	Si
1	CS11	0.228	5.982	1	1	1	0.85	0.42	1.7	3
	CS12	0.221	0.237							
	CS13	0.222	0.527							
2	CS21	0.453	0.566	0	1	1	0	0.4	1.6	2
	CS22	0.386	0.649							
	CS23	5.394	5.55							
3	CS31	6.415	7.067	1	0	1	0.66	0	1.33	2
	CS32	5.828	5.875							
	CS33	5.776	5.782							
		MaxRT=7.067								

TABLE 3.4 – Matrice de Dépendances entre contraintes temporelles

Supposons que la tache a trois contraintes temporelles avec un vecteur de priorité TC = TC3, TC1, TC2 (c'est-à-dire TC3 doit avoir la priorité la plus élevée que TC1 et TC2 doit avoir la priorité la plus basse). Les contraintes temporelles sont définies comme suit :

$$TC_1 = \begin{cases} VA_1 = \{1,\ 1, 1\} \\ \leq 5 \end{cases} .TC_2 = \begin{cases} VA_2 = \{0,\ 1, 1\} \\ \leq 7 \end{cases} .TC_3 = \begin{cases} VA_3 = \{1,\ 0, 1\} \\ \leq 8 \end{cases}$$

Afin de calculer les contraintes locales de chaque sous-tache, nous déterminons d'abord la valeur maximale du temps de réponse de toutes les classes. Dans cet exemple : MaxRT = 50. Sur la base de ces valeurs, nous pouvons calculer une quantification du temps de réponse maximal défini dans l'équation 3.4.

$$VMaxRT = \left\{ \frac{5.982}{7.067}, \frac{0.237}{7.067}, \frac{0.527}{7.067}, \frac{0.566}{7.067}, \frac{0.649}{7.067}, \frac{5.55}{7.067}, \frac{7.067}{7.067}, \frac{5.875}{7.067}, \frac{5.782}{7.067} \right\}$$
$$= \{0.84, 0.033, 0.07, 0.08, 0.091, 0.78, 1, 0.83, 0.81\}$$

Ensuite, nous calculons la moyenne des précédentes valeurs de valuation pour chaque sous-tache comme définit dans eq.3.6.

$$AVGVRT_1 = \frac{\sum_{j=1}^{3} VMaxRT_{1-j}}{3} = \frac{0.84 + 0.033 + 0.07}{3} = 0.31$$

Par la suite, une valuation des contraintes temporelles en utilisant la valeur constante qui sera comparée à la somme des valeurs maximales du temps de réponse dans chaque sous-tache.comme définit dans eq.3.7

$$V\left(TC_t\right) = \frac{Const_t}{maxTC} \;/\; maxTc = \sum_{i=1}^{n} \max\left[MaxRT_{i-j}\right] = 5.98 + 5.55 + 7.06 = 18.59$$

Alors $V\left(TC_t\right) = \{\frac{5}{18.59}, \frac{7}{18.59}, \frac{8}{18.59}\}$= {0.26, 0.37, 0.43}.

Ensuite, pour chaque contrainte temporelle, la valeur $VRT_{i-j}TC_t$ est calculée pour exprimer la valuation du temps de réponse augmentée de contraintes temporelles prioritaires en utilisant Eq.3.8, qui calcule les valeurs de $VRT_{i-j}TC_t$ pour chaque contrainte. Où $VA_t(i)$ est le i-ème vecteur d'affectation de la t ème contrainte et $VA_t(i)$ se réfèrent à la valuation de la t-ème contrainte et valeur de $VMaxRT_{i-j}$ comme défini dans eq.3.4.

Le numérateur de l'équation 3.8 est
$AV_t\left(i\right) * V\left(TC_t\right) * VMaxRT_{i-j} = AV_1\left(i\right) * V\left(TC_1\right) * VMaxRT_{i-j} = 0.84 * 0.26 * 1 = 0.21$
Le dominateur de l'équation 3.8 est $\sum_{i=1}^{n} AV_t\left(i\right) * AVGRT_i = \sum_{i=1}^{3} AV_1\left(i\right) * AVGRT_1 = 1 * 0.31 + 0 * 0.31 + 1 * 0.88 = 1.20$ Est la somme du produit entre la valuation de la contrainte et les valeurs $AVGVRT_i$ de tous les services candidat tels que définis dans l'équation 3.6. Le résultat de eq.3.6 est 0.18

Après avoir obtenu une valuation du temps de réponse pour chaque contrainte temporelle $VRT_{i-j}TC_t$, et considérer la valeur de priorité des contraintes, nous pouvons construire une fonction de priorité pour représenter la valeur de temps de réponse..

En utilisant Eq. 3.8, pour calculer les valeurs de $PrVRT_{i-j}$ entre toutes les contraintes $AVGPr_i$ Indique la valeur moyenne de $PrVRT_{i-j}$ pour chaque classe

Enfin, la valeur de contrainte locale de chaque sous tache est calculée en multipliant chaque valeur de $AVGPr_i$ par la valeur maxTC en prenant sa valeur entière comme défini dans eq.12. Le calcul des contraintes locales de l'exemple pour chaque classe de service est $LC_1 = E\left(0.10 * 18.59\right) = 1.85$, $LC_2 = E\left(0.11 * 18.59\right) = 2.04$, $LC_3 = E\left(0.23 * 18.59\right) = 4.27$

L'intérêt du calcul des contraintes locales de chaque sous-tache est de calculer la différence entre les contraintes locales et la valeur maximale du temps de réponse de chaque service candidat. Avec cette différence, nous pouvons classer chaque service candidat et choisir le service avec la valeur de différence la plus petite comme le meilleur choix. Le tableau 3.6 montre les étapes de calcul de la valeur de contrainte locale pour chaque sous-tache.

CS	Max RT	VMax RT	AVG VRT	AV1* AVG VRT	AV2* AVG VRT	AV3* AVG VRT	VRT TC1	VRT TC2	VRT TC3	Pr VRT	AVG Pri	LCi
CS11	5.98	0.84					0.19	0.50	0.24	0.26		
CS12	0.24	0.033	0.31	0.31	0.31	0.31	0.01	0.02	0.01	0.01	0.1	1.85
CS13	0.53	0.07					0.02	0.04	0.02	0.02		
CS21	0.57	0.08					0	0.05	0.02	0.03		
CS22	065	0.09	0.31	0	0.31	0.31	0	0.05	0.03	0.03	0.11	2.04
CS23	5.39	0.78					0	0.46	0.22	0.27		
CS31	7.07	1					0.22	0	0.28	0.26		
CS32	5.82	0.83	0.88	0.88	0	0.88	0.19	0	0.24	0.22	0.23	4.30
CS33	5.78	0.81					0.18	0	0.23	0.21		
Max	7.07		$\sum$	1.20	0.63	1.51						

TABLE 3.5 – Calcul des contraintes locales

ST	CS	MinRT	MaxRT	Lci	Diff(j)	Result	Classement	BestSelection
1	CS11	0.228	5.98	0.85	-5.13	PTC	3	CS12
	CS12	0.221	0.24	0.85	0.61	TCC	1	
	CS13	0.222	0.53	0.85	0.32	TCC	2	
2	CS21	0.45	0.57	2.04	1.47	TCC	1	
	CS22	0.38	0.65	2.04	1.39	TCC	2	
	CS23	5.39	5.55	2.04	-3.51	TI	3	CS21
3	CS31	6.41	7.07	4.30	-2.77	TI	3	
	CS32	5.82	5.88	4.30	-1.58	TI	2	
	CS33	5.77	5.78	4.30	-1.48	TI	1	CS33

TABLE 3.6 – Meilleure sélection de service

Dans le Tableau 3.6, la valeur Diff (j) compare la distance entre les contraintes locales de chaque sous-tache et la valeur maximale du temps de réponse. Le score de classement est calculé en triant les dernières différences calculées. En calculant la valeur de classement pour tous les services concrets, le service candidat CS12 CS21 CS33 a obtenu le score le plus élevé dans les classes 1, 2 et 3 respectivement. Par conséquent, le service composite CS12 CS21 CS33 est déterminé comme le meilleur service de sélection qui satisfait TC1 TC2 TC3 en tant que priorité de contrainte temporelle ordonnée.

3.8 Conclusion

Notre contribution propose un nouvel algorithme de sélection de services sous différentes CT, qui est contraintes de temps. Tout d'abord, un modèle de sélection formel est proposé. Ensuite, l'algorithme de sélection permet de trouver la combinaison de services candidat sous

multiples contraintes temporelles avec des priorités différentes. Le classement et la combinaison quasi-optimale des services candidats sont obtenus par la comparison avec les contraintes locales.

Chapitre 4

REALISATION ET EXPERIMENTATIONS :

4.1 Introduction

Après avoir présenté la méthode de sélection des services CMfg proposée, nous allons maintenant mettre en œuvre la solution proposée travers une implémentation. Nous aborderons quelques détails techniques correspondant à la mise en œuvre. Enfin,pour l'évaluation de notre approche on a utilisé des ensembles des données des services candidats web collectés dans [47] pour simuler les enregistrements de temps de réponse historiques de nombreux services fonctionnellement équivalents.

4.2 Realisation :

La figure 4.1 montre l'interface principale de l'application, elle contient deux boutons le premier bouton (PTCCS ALGORITHM) mène vers l'implémentation de notre solution de sélection, et le deuxième bouton (PRIORITY ALGORITHM) mène vers l'algorithme d'affectation automatique de priorité.

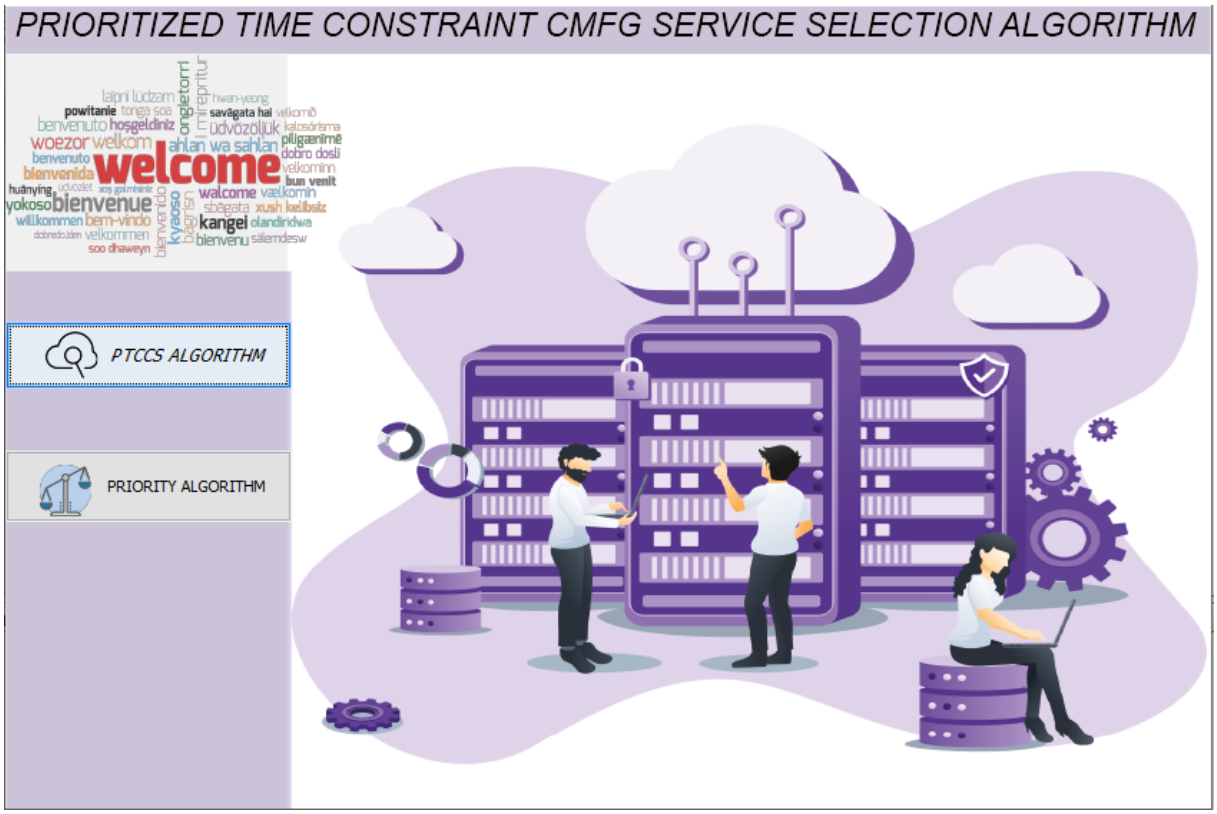


FIGURE 4.1 – Interface Principale

La figure 4.2 montre l'interface qui permet de saisir les différents paramètres de notre approche de sélection et composition des services CMfg, elle contient trois champs de texte et un bouton (NEXT), le premier champ de texte (nombre de sous-tache) permet saisir le nombre de sous-tache, le deuxième champ de texte (nombre constTC) permet de saisir le nombre de contrainte temporelle qui serons affecter au sous-taches et le troisième champ de texte (nombre Service candidat) permet de saisir le nombre de services candidats affectés à chaque sous-tache. Le bouton (NEXT) mène vers l'étape suivante d'exécution.

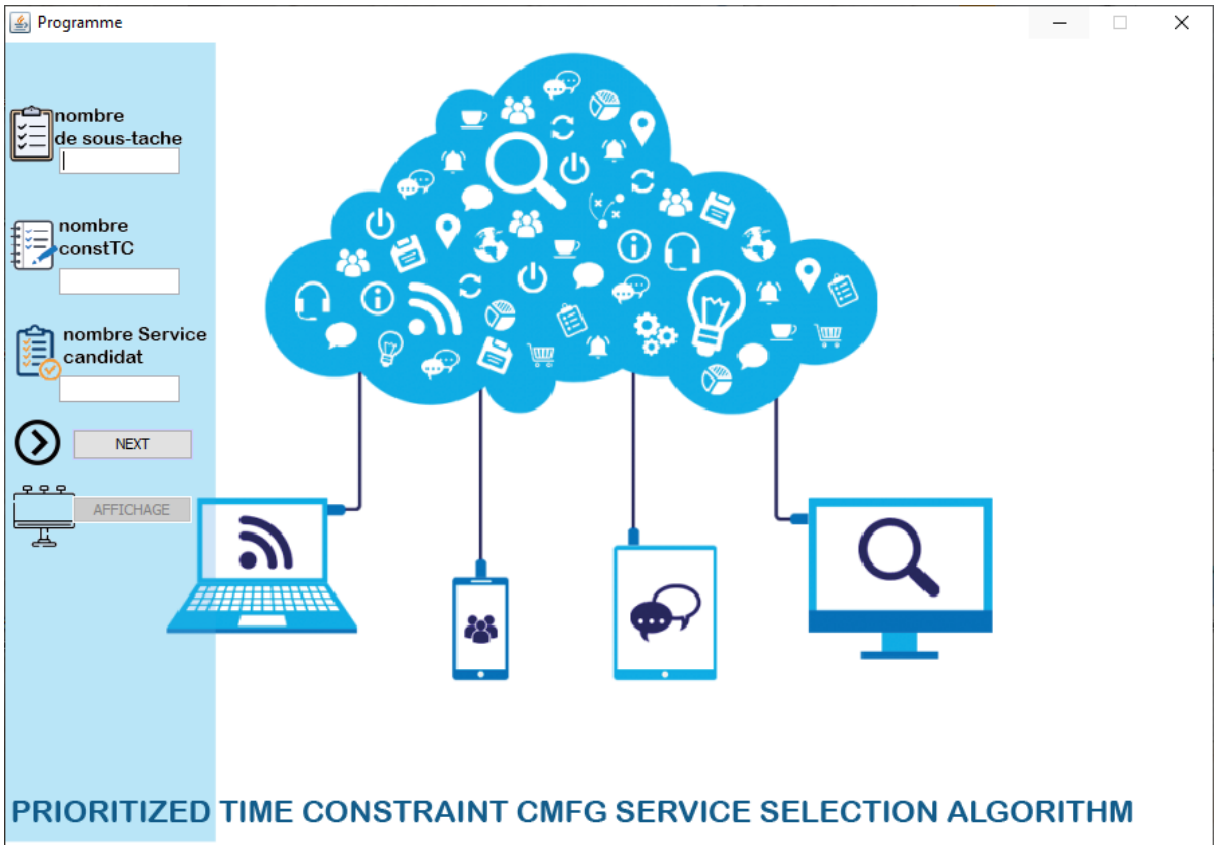


FIGURE 4.2 – Interface de saisi les différents paramètres de l'approche PTCCSSA

Les résultats de notre solution se présente dans la fig 4.3 en forme de rapport qui comporte l'affichage des contraintes locales calculées pour chaque sous-tache (lci[i]), l'état de cohérence temporelle de chaque service candidat, la meilleur sélection pour chaque sous-tache ainsi que les classement de la sélection

FIGURE 4.3 – Interface représente le rapport des résultats

La figure 4.4 montre l'interface qui permet de saisir les deux paramètres de notre algorithme de priorité, elle contient deux champs de texte et un bouton (ENTER), le premier champ de texte permet de saisir le nombre de sous-tache, le deuxième champ de texte permet de saisir le nombre de contrainte temporelle. Le bouton (ENTER) mène vers l'étape suivante d'exécution.

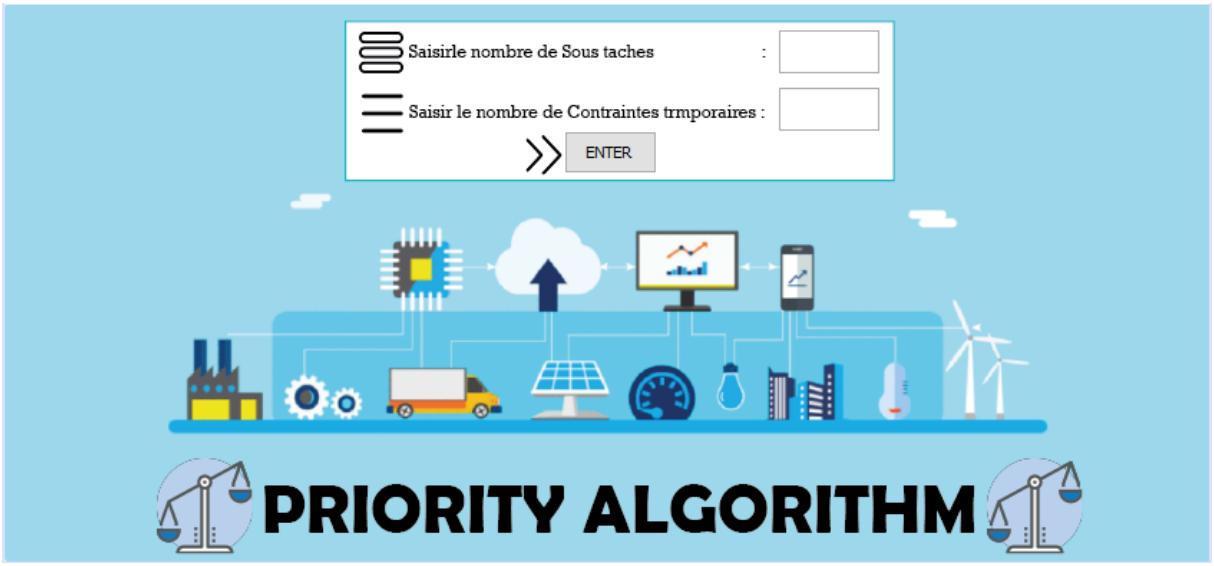


FIGURE 4.4 – Interface de saisi de l'algorithme de priorité

4.3 Evaluation de Performance :

Tous les algorithmes de notre approche sont implémentés en Java et exécutés à l'aide d'un ordinateur de bureau équipé d'un processeur Ryzen 5 3600xt, 4 GHz, 16 Go de RAM et exécutant Windows 10 (64 bits). Afin de vérifier l'efficacité de la méthode proposée, nous avons mené des expériences répétées 50 fois et donné le temps d'exécution moyen.

4.4 Performance de l'algorithme (PTCCSSA) en termes de temps de calcul

Dans cette section, nous présentons les résultats expérimentaux en fonction du temps de calcul de notre approche de sélection de services sous différentes priorités.
Afin de vérifier l'efficacité de l'algorithme (PTCCSSA), nous avons mené plusieurs expériences en utilisant la structure de combinaison séquentielle. Nous avons mené trois séries d'expériences pour évaluer les performances de notre méthode proposée en termes de temps de calcul. Dans ces expériences, nous avons mesuré :
1.Le temps de calcul de notre approche sous plusieurs sous-taches comparés à quatre autres méthodes de sélection de service.
2.Le temps de calcul de notre approche sous plusieurs services candidats comparés à cinq autres méthodes de sélection de service.
3.Le temps de calcul de notre approche sous plusieurs contraintes

Nous avons comparé l'efficacité de l'approche proposée avec les approches de sélection suivantes :

• Les approches globales [32].
• Artificial Bee Colony (ABC) [18].
• Particle Swarm Optimization (PSO) [31].
• Hybrid Artificial Bee Colony (HABC) [46].

4.5 Temps de calcul sous multiple sous-taches :

Dans un premier temps, nous montrons les résultats expérimentaux de notre méthode en fonction de ses performances en temps de réponse et du nombre de sous-tache dans une structure de composition séquentielle.

Les sous-taches varient de 10 à 100, le nombre de contraintes est fixé à 50, et chaque sous-tache comporte 100 services candidats.

Ce test compare la méthode proposée avec les quatre autres méthodes de sélection de services sous contraintes : HABC [46], ABC [18], GA [32], PSO [31]. Les résultats de cette expérience sont présentés dans la figure suivante.

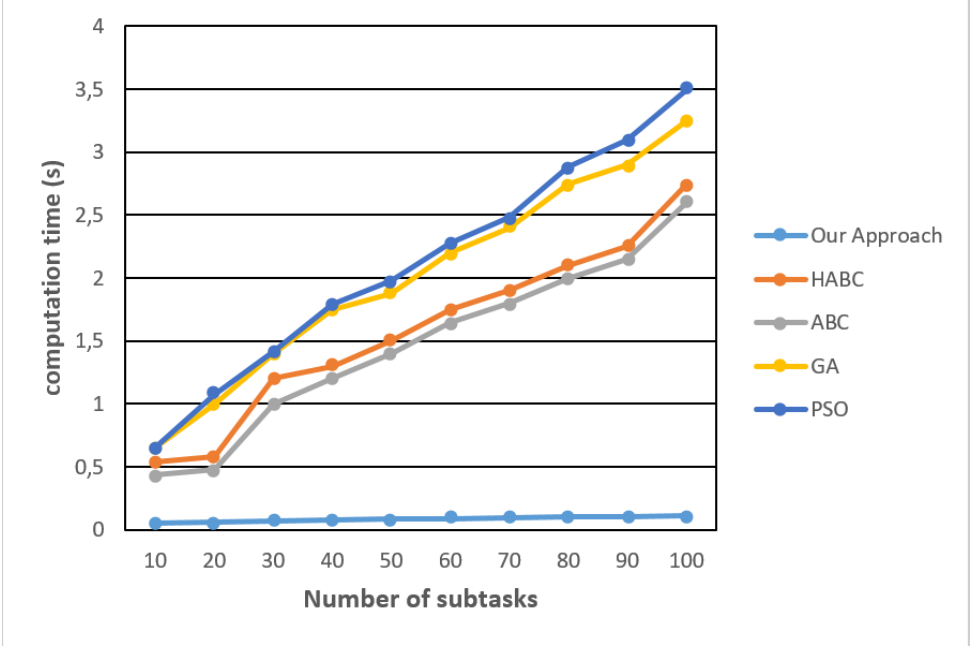


FIGURE 4.5 – Temps de calcul sous multiple sous-taches

Temps de calcul est quasiment stable ce qui est un aventage majeur paraport au autre solution qui augement en fonction,cela montre que la solution proposée dans cette étude est plus adaptée à ce genre de problèmes dans la selection de services CMfg.

4.6 Temps de calcul sous multiple services candidats :

Nous montrons les résultats expérimentaux de notre méthode en terme de ses performances en fonction de temps de calcule et de nombre de service candidat. Les services candidats varient de 50 à 400, le nombre de contraintes est fixé à 100 et il existe 5 sous-taches.

Afin d'obtenir une sélection de services cloud enrichie d'un ensemble de contraintes, nous utilisons notre algorithme PTCCSSA. De plus, on va comparer les performances de notre solution sous plusieurs services candidats avec les cinq autres méthodes de sélection de services augmentées de contraintes, il s'agit de HABC [46], ABC [18], PSO [31], GA [32]. La figure 4.6 montre le temps de calcul des différentes méthodes.

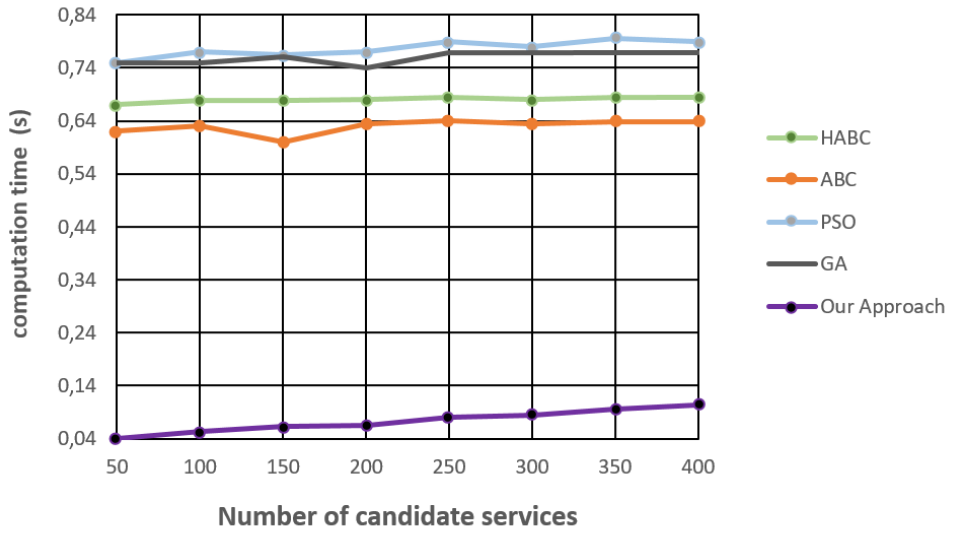


FIGURE 4.6 – Temps de calcul sous multiple services condidats

Comme montre les expériences, le temps de calcul augmente linéairement avec le nombre de service candidat. On peut observer que la méthode globale montre presque toujours le pire temps de calcul, car toutes les combinaisons de service candidat possibles sont vérifiées, et le meilleur temps de calcul est obtenu à partir de notre approche de selection, car lorsque le nombre de service candidat augmente, le temps de calcul il est presque stable.

4.7 Temps de calcul sous multiple contraintes :

Le but de la troisième expérience est d'évaluer l'efficacité de l'approche proposée sous plusieurs contraintes.

Nous présentons les résultats expérimentaux de notre approche en fonction de ses performances en termes de temps de réponse. Les contraintes temporelles ont été choisies au hasard et leurs

nombres varient de 5 à 10, le nombre de sous- tâches est fixé à 4 et chaque sous- tâches comporte 3 services candidats.

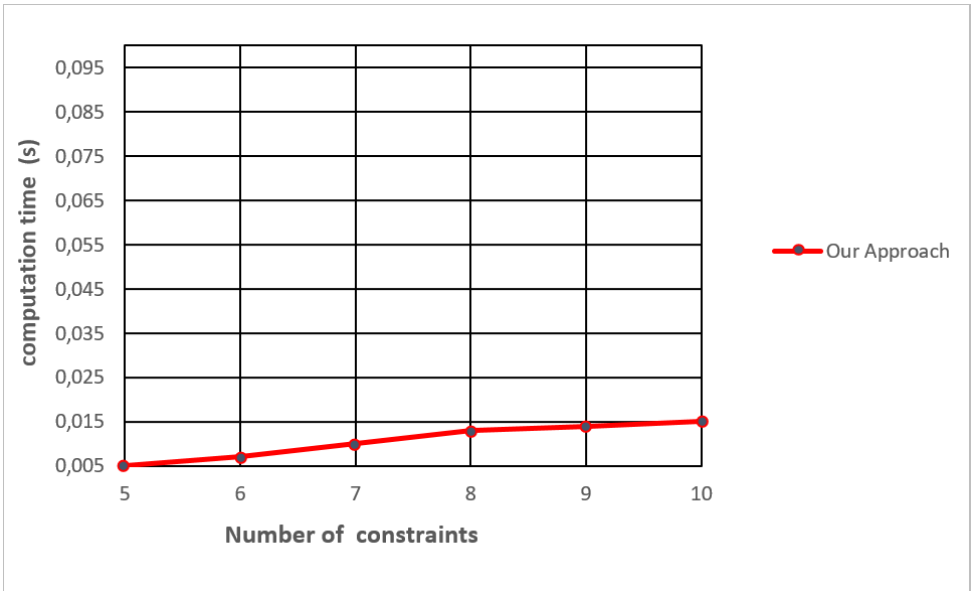


FIGURE 4.7 – Temps de calcul sous multiple contraintes

Sur la base des expériences présentées dans la figure 4.7, nous notons que le temps de calcul de notre approche varie entre 0.005 (s) et 0.015 (s) et augmente très lentement.

4.8 Conclusion

Dans de cette dernière étape de notre travail, nous avons mis en pratique l'approche proposée dans les chapitres précédents, nous avons expliqué les principaux modules pour la mettre en œuvre, et présenté les outils utilisés pour la mettre en œuvre. Cela montre la faisabilité et preuve de l'efficacité de notre solution. Nous avons présenté les résultats de mise en œuvre de notre approche proposée de sélection , et résumé les résultats obtenus.

Sur la base des expériences ci-dessus, nous avons remarqué que notre méthode peut atteindre des meilleures résultats avec un temps de calcul très court, ce qui améliore la sélection dans la composition de service. Les résultats expérimentaux montrent que notre approche améliore considérablement les performances temporelles du processus de sélection de services cloud dans le système de composition de services.

Chapitre 5

CONCLUSION GENERALE :

5.1 Bilan

Le travail de cette thèse apporte une solution pour la selection de service CMfg. En fait, il existe de nombreuses méthodes de formalisation et de modélisation de la composition de services cloud dans la littérature, mais peu considèrent les aspects temporels des différentes opérations que les services cloud peuvent effectuer.

Durant la phase de sélection, Notre contribution a proposé un nouvel algorithme de sélection augmenté de contraintes temporelles prioritaires tout en considérant les dépendances entre ces contraintes. La méthode de sélection classe tous les services concrets et trouvent la combinaison de service quasi-optimale en évitant une recherche exhaustive dans tout le plan d'exécution.

Afin de construire une composition cohérente qui obéit un ensemble de contraintes temporelles, Dans la deuxième contribution de notre thèse nous avons proposé un algorithme de vérification de cohérence temporelle lors de la selection.

Nos évaluations montrent une amélioration significative en termes de temps de calcul, ceci est particulièrement utile pour les applications avec des changements dynamiques et des changements en temps réel.

Bibliographie

[1] Aghamohammadzadeh, E. and Fatahi Valilai, O. (2020). A novel cloud manufacturing service composition platform enabled by blockchain technology. *International Journal of Production Research*, 58(17) :5280–5298.

[2] Aghamohammadzadeh, E., Malek, M., and Valilai, O. F. (2020). A novel model for optimisation of logistics and manufacturing operation service composition in cloud manufacturing system focusing on cloud-entropy. *International Journal of Production Research*, 58(7) :1987–2015.

[3] Ahn, G., Park, Y.-J., and Hur, S. (2019). Performance computation methods for composition of tasks with multiple patterns in cloud manufacturing. *International Journal of Production Research*, 57(2) :517–530.

[4] Akbaripour, H., Houshmand, M., Van Woensel, T., and Mutlu, N. (2018). Cloud manufacturing service selection optimization and scheduling with transportation considerations : mixed-integer programming models. *The International Journal of Advanced Manufacturing Technology*, 95(1) :43–70.

[5] Akoglu, L., Tong, H., and Koutra, D. (2015). Graph based anomaly detection and description : a survey. *Data mining and knowledge discovery*, 29(3) :626–688.

[6] Bouzary, H. and Chen, F. F. (2019). A hybrid grey wolf optimizer algorithm with evolutionary operators for optimal qos-aware service composition and optimal selection in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 101(9) :2771–2784.

[7] Buyya, R., Broberg, J., and Goscinski, A. M. (2010). *Cloud computing : Principles and paradigms*, volume 87. John Wiley & Sons.

[8] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms : Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6) :599–616.

[9] Cao, Y., Wang, S., Kang, L., and 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.

[10] Crouch, R. and Haines*, C. (2004). Mathematical modelling : Transitions between the real world and the mathematical model. *International Journal of Mathematical Education in Science and Technology*, 35(2) :197–206.

[11] Eisa, M., Younas, M., Basu, K., and Zhu, H. (2016). Trends and directions in cloud service selection. In *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 423–432. IEEE.

[12] Foster, I. (2003). The grid : Computing without bounds. *Scientific American*, 288(4) :78–85.

[13] Gamaleldin, A. M. (2013). An introduction to cloud computing concepts. *Software Engineering Competence Center*, page 2.

[14] Garg, S. K., Versteeg, S., and Buyya, R. (2013). A framework for ranking of cloud computing services. *Future Generation Computer Systems*, 29(4) :1012–1023.

[15] He, W., Jia, G., Zong, H., and Huang, T. (2019). Multi-objective cloud manufacturing service selection and scheduling with different objective priorities. *Sustainability*, 11(17) :4767.

[16] Huang, B., Li, C., and Tao, F. (2014). A chaos control optimal algorithm for qos-based service composition selection in cloud manufacturing system. *Enterprise Information Systems*, 8(4) :445–463.

[17] Kamble, S. S., Gunasekaran, A., Ghadge, A., and Raut, R. (2020). A performance measurement system for industry 4.0 enabled smart manufacturing system in smmes-a review and empirical investigation. *International journal of production economics*, 229 :107853.

[18] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer ....

[19] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.

[20] Li, Y., Yao, X., and Liu, M. (2019). Cloud manufacturing service composition optimization with improved genetic algorithm. *Mathematical Problems in Engineering*, 2019.

[21] Liu, Y., Wang, L., Wang, X. V., Xu, X., and Jiang, P. (2019). Cloud manufacturing : key issues and future perspectives. *International Journal of Computer Integrated Manufacturing*, 32(9) :858–874.

[22] Long, W. and Xu, S. (2016). A novel grey wolf optimizer for global optimization problems. In *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 1266–1270. IEEE.

[23] Marinelli, E. (2009). *Cloud Computing on Mobile Devices using MapReduce*. PhD thesis, Master Thesis Draft, Computer Science Dept., Carnegie Mellon University (CMU).

[24] Misra, S., Singh, A., Chatterjee, S., and Mandal, A. K. (2015). Qos-aware sensor allocation for target tracking in sensor-cloud. *Ad Hoc Networks*, 33 :140–153.

[25] Modi, K. J. and Garg, S. (2019). A qos-based approach for cloud-service matchmaking, selection and composition using the semantic web. *Journal of Systems and Information Technology*.

[26] Ning, F., Zhou, W., Zhang, F., Yin, Q., and Ni, X. (2011). The architecture of cloud manufacturing and its key technologies research. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 259–263. IEEE.

[27] Que, Y., Zhong, W., Chen, H., Chen, X., and Xu, J. (2018). Improved adaptive immune genetic algorithm for optimal qos-aware service composition selection in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 96(9-12) :4455–4465.

[28] Rajeswari, M., Sambasivam, G., Balaji, N., Basha, M. S., Vengattaraman, T., and Dhavachelvan, P. (2014). Appraisal and analysis on various web service composition approaches based on qos factors. *Journal of King Saud University-Computer and Information Sciences*, 26(1) :143–152.

[29] Rashid, A. and Chaturvedi, A. (2019). Cloud computing characteristics and services : a brief review. *International Journal of Computer Sciences and Engineering*, 7(2) :421–426.

[30] Tao, F., Zhang, L., Liu, Y., Cheng, Y., Wang, L., and Xu, X. (2015). Manufacturing service management in cloud manufacturing : overview and future research directions. *Journal of Manufacturing Science and Engineering*, 137(4).

[31] Tao, F., Zhao, D., Yefa, H., and Zhou, Z. (2010). Correlation-aware resource service composition and optimal-selection in manufacturing grid. *European Journal of Operational Research*, 201(1) :129–143.

[32] Wang, D., Yang, Y., and Mi, Z. (2015). A genetic-based approach to web service composition in geo-distributed cloud environment. *Computers & Electrical Engineering*, 43 :129–141.

[33] Wang, F., Laili, Y., and Zhang, L. (2020). A many-objective memetic algorithm for correlation-aware service composition in cloud manufacturing. *International Journal of Production Research*, pages 1–19.

[34] Wang, L., Guo, C., Li, Y., Du, B., and Guo, S. (2019). An outsourcing service selection method using ann and sfla algorithms for cement equipment manufacturing enterprises in cloud manufacturing. *Journal of Ambient Intelligence and Humanized Computing*, 10(3) :1065–1079.

[35] Wang, L., Guo, S., Li, X., Du, B., and Xu, W. (2018a). Distributed manufacturing resource selection strategy in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 94(9) :3375–3388.

[36] Wang, Y., Dai, Z., Zhang, W., Zhang, S., Xu, Y., and Chen, Q. (2018b). Urgent task-aware cloud manufacturing service composition using two-stage biogeography-based optimisation. *International Journal of Computer Integrated Manufacturing*, 31(10) :1034–1047.

[37] Wu, Y., Jia, G., and Cheng, Y. (2020). Cloud manufacturing service composition and optimal selection with sustainability considerations : a multi-objective integer bi-level multi-follower programming approach. *International Journal of Production Research*, 58(19) :6024–6042.

[38] Xiang, F., Hu, Y., Yu, Y., and Wu, H. (2014). Qos and energy consumption aware service composition and optimal-selection based on pareto group leader algorithm in cloud manufacturing system. *Central European Journal of Operations Research*, 22(4) :663–685.

[39] Yang, Y., Yang, B., Wang, S., Jin, T., and Li, S. (2020). An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing. *Applied Soft Computing*, 87 :106003.

[40] Yang, Y., Yang, B., Wang, S., Liu, W., and Jin, T. (2019). An improved grey wolf optimizer algorithm for energy-aware service composition in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 105(7) :3079–3091.

[41] Yuan, M., Zhou, Z., Cai, X., Sun, C., and Gu, W. (2020). Service composition model and method in cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 61 :101840.

[42] Zhang, L., Guo, H., Tao, F., Luo, Y., and Si, N. (2010a). Flexible management of resource service composition in cloud manufacturing. In *2010 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 2278–2282. IEEE.

[43] Zhang, Q., Cheng, L., and Boutaba, R. (2010b). Cloud computing : state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1) :7–18.

[44] Zhao, J., Li, M., Zhou, Y., and Wang, P. (2020). Building innovative service composition based on two-way selection in cloud manufacturing environment. *Mathematical Problems in Engineering*, 2020.

[45] Zhou, J. and Yao, X. (2017a). De-caabc : differential evolution enhanced context-aware artificial bee colony algorithm for service composition and optimal selection in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 90(1-4) :1085–1103.

[46] Zhou, J. and Yao, X. (2017b). A hybrid artificial bee colony algorithm for optimal selection of qos-based cloud manufacturing service composition. *The International Journal of Advanced Manufacturing Technology*, 88(9-12) :3371–3387.

[47] Zhu, J. (9 Dec 2018). wsdream dataset. `https://github.com/wsdream/wsdream-dataset`.

[48] Zhu, L.-N., Li, P.-H., and Zhou, X.-L. (2019). Ihdetbo : A novel optimization method of multi-batch subtasks parallel-hybrid execution cloud service composition for cloud manufacturing. *Complexity*, 2019.