

République Algérienne Démocratique et Populaire  
Université Abou Bakr Belkaid– Tlemcen  
Faculté des Sciences  
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

*Option : modèle intelligent et décision (M.I.D)*

*Thème*

**Sélection de service web basée sur la  
FOA (fruit Fly optimization algorithm)**

**Réalisé par :**

- BERRABAH Samir

*Présenté en novembre 2017 devant le jury composé de MM.*

- *Mme Elyebdri Zeyneb* (Présidente)
- *Mme Halfaoui Amel* (Encadreur)
- *Mr Merzoug Mohamed* (Examineur)

Année universitaire : 2016-2017

## Remerciement

Je tiens vivement, au terme de ce travail, à exprimer mes sentiments de gratitude à toute personne ayant participé de près ou de loin à son élaboration. Je tiens à remercier tout particulièrement Mme. HALFAOUI Amel de m'avoir encadré, pour ses conseils, sa patience et ses encouragements. Tous les membres du jury qui ont accepté d'évaluer mon travail. Je tiens à remercier toutes les enseignants de master II MID de m'avoir fait cette formation et permis d'acquérir des nouvelles connaissances.

## Dédicaces

A mes chers parents, A mes chers frères et sœurs, Aux chers membres de ma famille A tous ceux que j'aime et qui m'aiment. En guise de reconnaissance de leur confiance en moi.

## **Table des matières**

### **Introduction général**

1	Contexte :.....	2
2	Problématique :.....	3
3	Contribution :.....	4
4	Plan de travail :.....	4

### **Chapitre I : Services web et la Sélection basée sur la QoS**

1	Introduction : .....	6
2	Définitions : .....	6
2.1	Définition et description de Service Web .....	6
2.2	Principales technologies de développement de Services Web.....	6
2.2.1	XML – eXtensible Markup Language .....	6
2.2.2	SOAP: Simple Object Access Protocol .....	7
2.2.3	WSDL : Web Services Description Language .....	8
2.2.4	UDDI: Universal Description Discovery and Integration .....	10
2.3	Les thèmes de recherche actuels .....	10
2.3.1	La sélection des services Web .....	11
2.3.2	Composition Automatique des Services Web Sémantiques sans Etats ....	12
2.3.3	Critères de Qos (Quality of service) : .....	13
3	La fonction objective (score ou fitness) .....	15
4	Optimisation combinatoire .....	16
4.1	Définition : .....	16
4.2	Approches d’optimisation combinatoire (approches de sélection) : .....	16
4.2.1	L’optimisation multi-objective : .....	16
4.2.2	L’optimisation mono-objective : .....	16
5	État de l’art des travaux sur le problème QoSSWC .....	18
5.1	Technique de base de données : .....	18
5.2	Approches Exactes : .....	19
5.3	Approche heuristique (Approximative) : .....	19
5.4	Approche Méta-heuristique : .....	19
5.5	Les méta-heuristiques à base de solution unique : .....	19
5.6	Les méta-heuristiques à base de population de solution : .....	20
6	Conclusion : .....	20

## **Chapitre II : Sélection de Services web basée sur l'Algorithme de la FOA**

1	Introduction :	22
2	WS-FOA : Web Service composition based on Fruit Fly Optimization Algorithm (la composition de services web basée sur l'algorithme d'optimisation Fruit Fly) :	22
2.1	Définition :	22
2.2	Comportement de Fruit Fly :	22
2.3	La modélisation de la composition de service :	23
2.3.1	Définition 1 Structure QoS :	23
2.3.2	Définition 2 Structure Service (s) Web :	24
2.3.3	Définition 3 Structure Composition de Service (SC) :	24
2.3.4	Définition 4 QoS prétraitement :	27
2.3.5	Les étapes de l'algorithme de prétraitement QoS :	27
2.4	Modélisation de l'Algorithme :	28
2.4.1	Exemple de changement en position d'un fruit Fly et le Fruit Fly individuels :	28
2.4.2	Définition de la distance :	29
2.4.3	La fonction Fitness :	29
2.4.4	Principe de fruit Fly :	30
2.4.5	Les étapes de l'algorithme WS FOA :	31
3	Conclusion :	34

## **Chapitre III : Implémentation & Expérimentation**

1	Introduction :	36
2	Présentation de l'environnement de développement :	37
3	Outils de système et paramètres :	37
4	Interface (paramètre et résultat) :	38
4.1	Générer et charger la base :	38
4.2	Lancement d'Algorithme et Affichage :	41
5	Expérimentations :	43
6	BILAN :	44
7	Conclusion :	45
	<b>Conclusion général</b> :	47
	Références Bibliographiques :	48
	Liste de Figures :	50
	Liste des Acronyms :	51



**Introduction**

**Général**

## 1 Contexte :

La technologie des services Web est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes. Les services Web sont basés sur le modèle SOA (service oriented architecture).

L'architecture orientée service (SOA) est née pour répondre aux inconvénients des technologies orientées composants, telles que CORBA [OMG, 2008] (Common Object Request Broker Architecture), Java RMI [Downing, 1998] (Remote Method Invocation), DCOM (Distributed Component Object Model). [Horstmann et Kirtland, 1997].

L'architecture SOA [21] est caractérisée par son couplage faible, son indépendance par rapport aux plateformes, et sa grande capacité d'intégration et de réutilisation.

Le couplage faible signifie que la dépendance entre les entités est très réduite, et ceci permet une large capacité de réutilisation et d'intégration de ces composants [1].

L'utilisation des technologies standards du Web telles HTTP (**Hyper Text Transport Protocol**) et XML (**extensible Markup Language**) par les services Web facilite le développement d'applications réparties sur Internet, et permet d'avoir des applications très faiblement couplées. L'intégration est sans doute le facteur essentiel qui favorise l'utilisation des services Web [2].

Les entreprises ont commencé à utiliser la technologie de services web pour l'intégration B2B (i.e. intégration Business to Business). Les applications qui utilisent ces technologies, ont des liaisons fixes aux services web. Ce schéma n'a pas en compte de l'existence d'autres services avec une meilleure qualité de service [3]. Dans ce travail nous proposons une adaptation pour la sélection de services web par une application de composition de service basée sur l'algorithme d'optimisation Fruit Fly.

## 2 Problématique :

La problématique de notre travail concerne la sélection de services web.

Un service Web peut être vu comme un programme, décrit par ses attributs en entrée et ses attributs fournis en sortie, pouvant dynamiquement être découverts et invoqués à travers le Web.

La composition de services permet de regrouper au sein d'un "tout" cohérent des invocations à différents services. Pour une même requête utilisateur (composée d'attributs donnés en entrée et d'attributs souhaités en sortie), il existe plusieurs compositions de services possibles. [4]

Ces différentes compositions se distinguent les unes des autres par leurs propriétés et par leurs propriétés non fonctionnelles, comme les critères de Qualité de Service (réputation, fiabilité, durée d'exécution etc.). Un bon algorithme de composition doit donc retourner à l'utilisateur une composition transactionnelle (permettant quelles services composants forment un "tout" cohérent en vérifiant certaines contraintes au niveau de leur propriété transactionnelle) et optimisant la Qualité de service globale du service composite.

L'ensemble des services web disponibles peut être vu comme un graphe, dont les nœuds représentent les attributs en entrée et en sortie ainsi que les services Web et les arcs entre ces nœuds correspondent soient à des liens entre des attributs entrées et un service Web (i.e. les attributs en entrée dont le service Web a besoin pour pouvoir s'exécuter) ou des liens entre un service Web et ses attributs en sortie (i.e. les attributs que le service produit après s'être exécuté). Une composition de services correspond à un sous-graphe de ce graphe des services Web disponibles.

Le problème de ce travail est de sélectionner une combinaison de service web à base de l'algorithme d'optimisation Fruit Fly.



### 3 Contribution :

Notre travail consiste à chercher et retourner une meilleure sélection des services web composites en fonction de leurs (QoS i.e : **Quality of Service**). Pour cela, nous avons proposé une approche d'optimisation mono objective basée sur les méta-heuristiques en particulier l'algorithme de Fruit Fly qui se base sur le comportement des Fruit Fly et leur mode de vie ainsi que leur productivité.

L'algorithme d'optimisation proposé est nommé l'algorithme WS-FOA (Web Services selection based on Fruit Fly Optimisation Algorithm).

Ce dernier permet d'obtenir la quasi meilleure composition du service web.

En construisant le modèle de service Web, divers attributs de service Web peuvent être clairement définis, Ceci est pratique pour la mise en œuvre de l'algorithme.

### 4 Plan de travail :

Le mémoire est composé de trois chapitres, qui sont organisés comme suit :

#### **Le premier chapitre :**

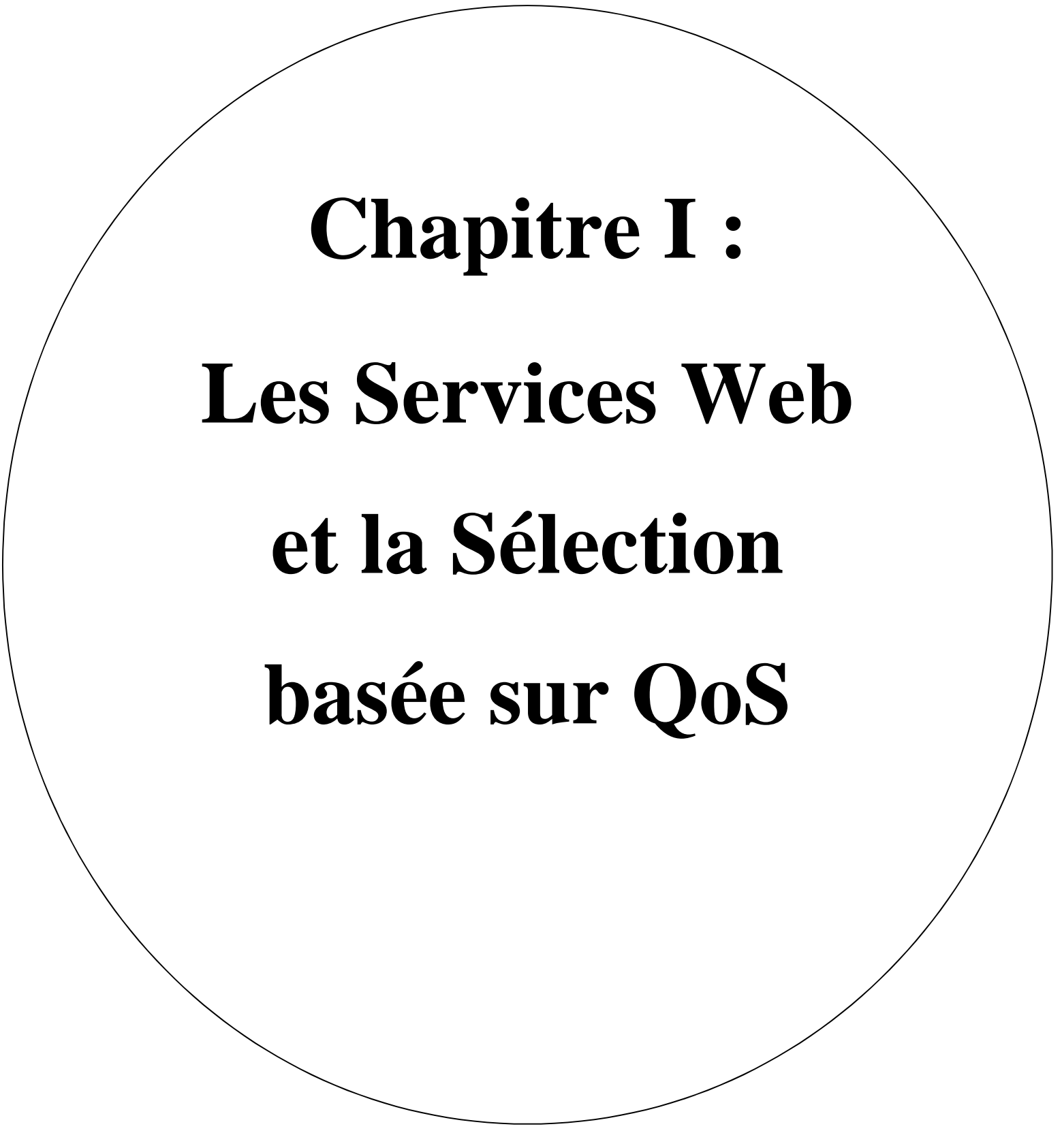
Nous avons présenté quelques définitions, architectures et protocoles qui concernent les services web et la sélection selon la qualité de service QoS.

#### **Le deuxième chapitre :**

Nous avons présenté notre algorithme avec un petit historique, une définition, son fonctionnement, les domaines d'application et enfin nous avons cité quelques avantages, inconvénients. La suite du chapitre décrit la conception de notre application.

#### **Le troisième chapitre :**

Ce dernier chapitre est consacré à l'implémentation et l'expérimentation de notre algorithme. Les interfaces et les résultats expérimentaux y seront présentés.



**Chapitre I :**  
**Les Services Web**  
**et la Sélection**  
**basée sur QoS**

## 1 Introduction :

Avec l'essor du web qui a eu lieu dans les dernières années, le besoin de permettre qu'une application client invoque un service d'une application serveur en utilisant Internet, a surgi. Ce même besoin a été l'origine de ce qu'on appelle communément les services web. En tenant en compte que les services web permettent de connecter des applications différentes, l'utilité de cette technologie devient évidente et importante. Pour cette raison les activités de la recherche et développement autour du sujet *services Web*, ont un dynamisme très haut. Dans ce chapitre on va décrire les technologies qui ont généralement un lien très grand avec ces services Web.

## 2 Définitions :

### 2.1 Définition et description de Service Web :

Le groupe de W3C qui travaille sur les services Web a utilisé dans un document appelé

« Web Services Architecture » la définition de service Web suivante [5] :

« Un service web est un système logiciel destiné à supporter l'interaction ordinateur-ordinateur sur le réseau. Il a une interface décrite en un format traitable par l'ordinateur (e.g. WSDL). Autres systèmes réagissent réciproquement avec le service web d'une façon prescrite par sa description en utilisant des messages SOAP, typiquement transmis avec le protocole http et une sérialisation XML, en conjonction avec d'autres standards relatifs au web ».

### 2.2 Principales technologies de développement de Services Web :

Une caractéristique qui a permis un grand succès de la technologie des services web est qu'elle est construite sur des technologies standards de l'industrie. Dans cette section il y a une description de ces technologies [6].

#### 2.2.1 XML – eXtensible Markup Language :

XML [7] est un standard promulgué par le W3C, l'organisme chargé de standardiser les évolutions de web. On retrouve dans XML une généralisation des idées contenues dans HTML et SGML. XML permet de définir des balises extensible et indépendante de

l'aspect graphique que doit revêtir la page dans le navigateur web. Dans XML les balises permettent d'associer toutes sortes d'informations au fil du texte. Il a été conçu pour des documents arbitrairement complexes, tout en s'appuyant sur cinq grands principes simples et clairs :

- Lisibilité à la fois par les machines et par les utilisateurs.
- Définition sans ambiguïté du contenu d'un document.
- Définition sans ambiguïté de la structure d'un document.
- Séparation entre documents et relation entre documents.
- Séparation entre structure des documents et présentation des documents [8].

### **2.2.2 SOAP: Simple Object Access Protocol:**

SOAP [8] est une spécification de communication entre services web, par échange de messages en XML au travers du web. Simple et facile à implémenter dans les serveurs web ou dans les serveurs d'application, SOAP est indépendant des langages de programmation ou des systèmes d'exploitation employé pour l'implémentation des services web. SOAP est un protocole de communication entre applications fondé sur XML, visant à satisfaire un double objectif : servir de protocole de communication sur les intranets, dans une optique d'intégration d'applications d'entreprise, et permettre la communication entre applications et services web, en particulier dans un contexte d'échanges inter-entreprises (le réseau internet) [7].

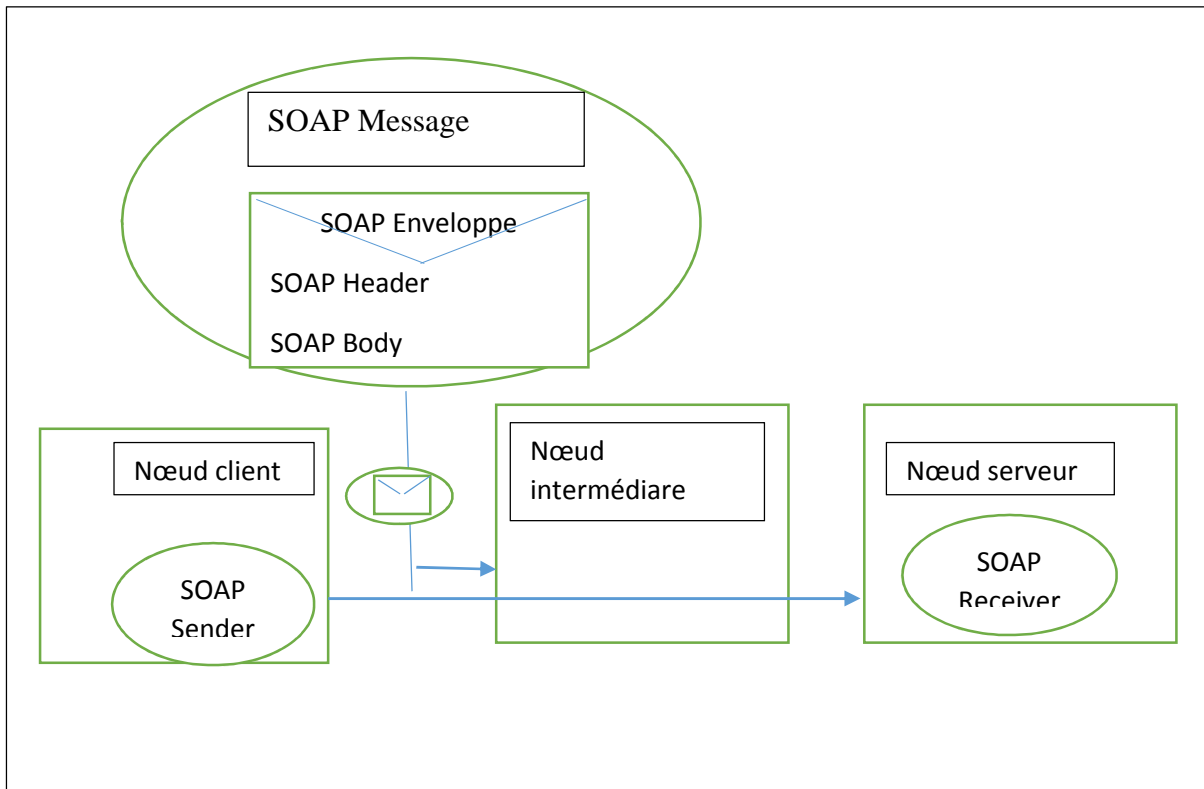
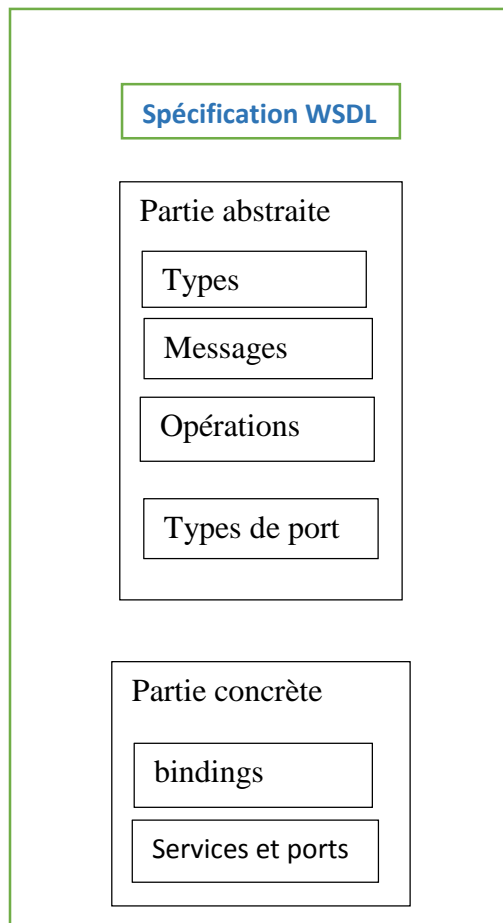


Figure I.1: Structure d'un message SOAP [22]

### 2.2.3 WSDL : Web Services Description Language :

WSDL est un langage qui permet de décrire les services web, et en particulier, les interfaces des services web. Ces descriptions sont les documents XML. WSDL décrit un service web en deux étapes fondamentales : une abstraite et une concrète. Dans chaque étape, la description utilise un nombre de constructions pour favoriser la réutilisation de la description et pour séparer les préoccupations de conception indépendantes (Figure I.2).



**Figure I 2 : La spécification d'un service Web avec WSDL [25]**

Au niveau abstrait, WSDL décrit un service Web en termes des *messages* qu'il envoie et reçoit ; les *messages* sont décrits de façon indépendante d'un format spécifique de fil en utilisant un système de *types*, typiquement un schéma XML. La partie abstraite est composée de définitions de "*port type*" qui sont analogues aux interfaces dans les IDLs du "middleware" traditionnel. Chaque "*port type*" est une collection logique d'opérations. Une "*opération*" (i.e. opération) associe un modèle d'échange de message à un ou plusieurs messages. Un *message* est une unité de communication avec un service web. Il représente les données échangées dans une unique transmission logique. Un modèle d'échange de messages identifie l'ordre et la cardinalité des messages envoyés et/ou reçus. Une interface groupe un ensemble d'opérations.

Au niveau concret, un "*binding*" indique des détails de format de transport pour une ou plusieurs interfaces. Un "*endpoint*" (i.e. point final) associe une adresse de réseau à un "*binding*" (i.e. attache). Finalement, un service groupe un ensemble d'*endpoints* qui implémente une interface commune.

#### 2.2.4 UDDI: Universal Description Discovery and Integration:

L'objectif primaire d'UDDI est la spécification d'un canevas pour décrire et découvrir des services Web. Le noyau d'UDDI travaille avec la notion de "business registry", qu'est un service sophistiqué de noms et répertoires. Plus précisément, UDDI définit des structures de données et APIs pour publier les descriptions des services dans le registre et pour interroger le registre afin de chercher des descriptions publiées. Parce que les APIs de UDDI sont aussi spécifiés en WSDL avec une attache SOAP, le registre peut être invoqué comme un service Web (en conséquence, ses caractéristiques peuvent être décrites dans le même registre, comme un autre service).

Les spécifications du « registry » UDDI ont deux buts principaux en ce qui concerne la découverte d'un service : le premier, soutenir les développeurs dans la découverte d'informations sur les services. Le deuxième, permettre la liaison dynamique et en conséquence habilitier les clients pour interroger le registre et obtenir des références aux services d'intérêt.

De plus pour définir un registre de services Web, UDDI aussi adresse le concept d'Universal *Business Registry* (*UBR*). En fait, le but initial du consortium a été de soutenir des répertoires mondiaux où chacun pourrait publier des descriptions de service et où chacun pourrait interroger ces répertoires pour trouver les services d'intérêt. L'idée était qu'UBRS pourrait contenir chaque service déployé dans le monde.

#### 2.3 Les thèmes de recherche actuels :

Les actuels sujets de recherche dans le domaine des services web sont nombreux. Un nombre considérable d'études tournent autour de sélection et la composition des services web. Dans cette section, nous allons analyser et décrire les travaux les plus pertinents.

### 2.3.1 La sélection des services Web :

Avec la sélection des services web, on cherche à choisir le meilleur fournisseur d'un service web, étant donné un ensemble de fournisseurs de ce service.

Il y a une proposition de base sur la Qualité de Service (i.e. QoS en anglais *Quality of Service – QoS*) [9]. La plate-forme de cette proposition est un modèle réglé qui peut coexister avec les registres UDDI dérégulés. Les registres dérégulés actuels peuvent fournir des services aux gens pour qui la qualité de service n'est pas importante. Les registres réglés basés sur ce modèle peuvent servir aux applications qui ont besoin de qualité de service.

Il y a quatre rôles dans ce modèle (Figure I.3) :

- Fournisseur de services web.
- Consommateur de services web.
- *Certificateur* de la Qualité de Service.
- Et le nouvel registre.

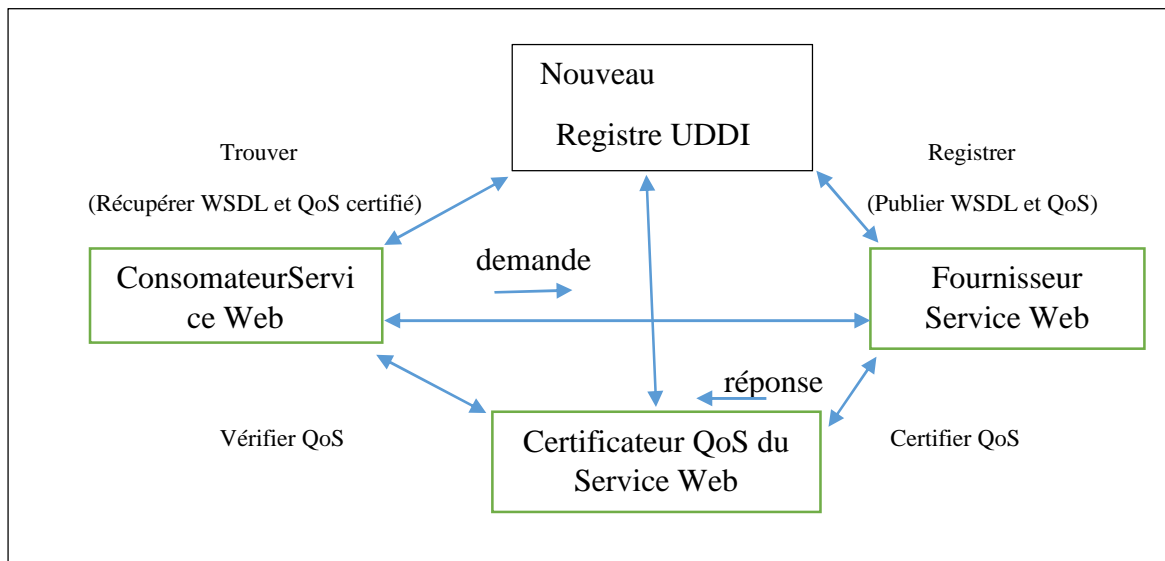
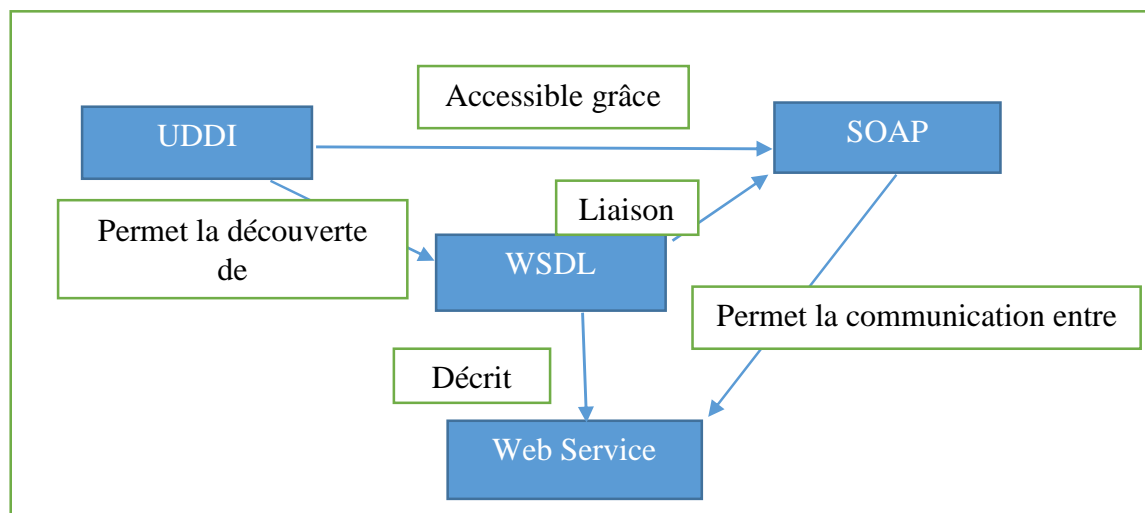


Figure I.3 : Un nouveau modèle de registre et découverte de services web [23]



Le fournisseur du service offre le service web en publiant ce dernier dans le nouveau registre ; le consommateur a besoin du service web offert par le fournisseur ; le nouvel registre UDDI est un lieu de stockage de services web enregistrés avec facilités de recherche. Le nouveau rôle *certificateur* sert à vérifier la qualité de service du fournisseur du service. Le nouveau registre est différent de l'actuel modèle UDDI en ayant information sur la description fonctionnelle du service web, et sur la qualité de service enregistré dans le stockage. Les interactions entre les différents rôles sont montrées dans la figure I.3.

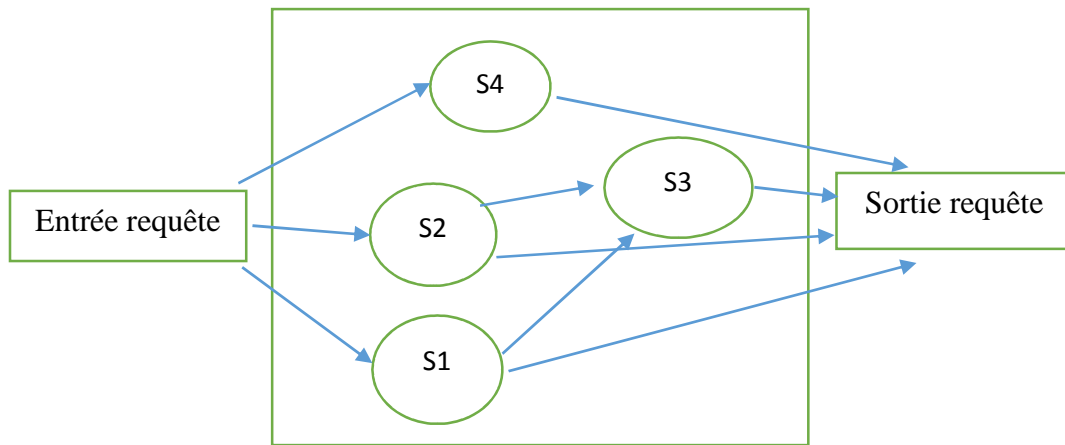
On peut schématiser les différentes relations entre les spécifications des services web [21] par la figure I.4 :



**Figure I 4 : Les différentes relations entre les spécifications des services web [21]**

### 2.3.2 Composition Automatique des Services Web Sémantiques sans Etats :

Dans la plupart des cas, un seul service ne peut répondre aux besoins des clients, et par conséquent, nous sommes obligés de composer plusieurs services afin de satisfaire une requête. Si nous considérons l'exemple de planification d'un voyage, un utilisateur doit consommer plusieurs types de services (qui sont offerts par plusieurs organisations). Par exemple, le premier type de service permet de réserver l'hôtel, le deuxième permet la réservation du billet d'avion, le troisième permet le paiement en ligne, et le quatrième permet la consultation de la météo.



**Figure I.5 : Exemple de composition.**

L'utilisateur fournit un ensemble de concepts en entrée (nom, prénom, date d'arrivée/départ, lieu de départ/d'arrivée) et attend un ensemble de concepts en sorties (des messages de confirmation, des informations de la météo...). Il est clair qu'il n'y a pas de service individuel qui répond à ce besoin, mais l'assemblage de quatre services différents permet de générer une composition ayant une valeur ajoutée.

### 2.3.3 Critères de Qos (Quality of service) :

Cinq critères de qualité génériques pour des services élémentaires sont considérés :

- ) **Latence** : Latence mesure le retard prévu en secondes entre le moment où l'exécution commence et moment où l'exécution finit. La latence est calculé par :

$$\text{Latence (sol)} = \sum_{i=1}^n l_i \quad (C)$$

- ) **Cout** : le prix d'exécution est la somme d'argent qu'un demandeur de service doit payer pour exécuter une opération d'un service. Les allocataires de service annoncent directement le prix d'exécution de leurs opérations.

$$\text{Cout (sol)} = \sum_{i=1}^n c_i \quad (C)$$

- ) **Réputation** : la réputation d'un service est une mesure de sa fidélité. Elle dépend principalement des expériences de l'utilisateur d'employer ce service.

Les différents utilisateurs peuvent avoir le service à peu près identique de différents avis. La valeur de la réputation est définie comme rang moyen indiqué au service par des utilisateurs.

$$\text{Rep (sol)} = \frac{1}{n} \sum_{i=1}^n r_i \quad (C)$$

J) **Disponibilité** : La disponibilité d'un service est la probabilité que le service est accessible

$$\text{Disp (sol)} = \log_2 \prod_{i=1}^n d_i \quad (C)$$

J) **Fiabilité** : La fiabilité d'un service est la probabilité qu'une demande est correctement répondue dans le délai de temps prévu maximum (ce critère calcul le taux d'erreur). La fiabilité est une mesure liée à la configuration de matériel et/ou de logiciel des services et aux connexions réseau entre les demandeurs de service et les fournisseurs.

$$\text{Fiab (sol)} = \log_2 \prod_{i=1}^n f_i \quad (C)$$

On suppose que les domaines de valeurs de critères de Qos sont définit comme suit :

Critères	Domaine de valeur
Cout	1...30euros
Temps d'exécution	1...300ms
Fiabilité	0 ...5
Disponibilité	0...1
Réputation	0...1

**Tableau I.1: Les domaines de valeurs des critères**

### 3 La fonction objective (score ou fitness) :

Ces critères doivent optimiser une fonction globale qui maximise les critères positifs (Réputation, Disponibilité et Fiabilité) et minimise les critères négatifs (cout et temps d'exécution), et nous devons également satisfaire des contraintes globales par exemple :

- Cout (sol) < 200euros : Le cout d'une solution doit être moins de 200euros.
- Temps (sol) < 1000s : Le temps d'exécution doit être moins de 1000ms.
- Rep (sol) > 3 : La réputation doit être supérieure à 3.
- Disp (sol) > 0.6 : La disponibilité doit être supérieure à 60%.
- Fiab (sol) > 0.6 : La fiabilité doit être supérieure à 60%.

Nous définissons une fonction score qui nous facilite la comparaison entre les qualités de chaque composition de web services :  $Q_i(\text{sol}) = (\text{Cout}(\text{sol}), \text{Rep}(\text{sol}), \text{Disp}(\text{sol}), \text{Fiab}(\text{sol}), \text{Temps}(\text{sol}))$

$$U(C) = \sum_{Q \in N \in Y} W \frac{Q_m - Q_s}{Q_m - Q_m} + \sum_{Q \in P} W \frac{Q_s - Q_m}{Q_m - Q_m}$$

Où  $W_i \in R_c^+$  et  $\sum_{i=1}^n W = 1$  est la somme des poids assignée pour chaque critère de qualité pour représenter les priorités de l'utilisateur.

Les contraintes globales doivent être vérifiées par la condition suivante :  $Q(k(C)) \geq \text{Conti}$

Et pour intégrer les contraintes globales exigées par l'utilisateur on va calculer un nouveau score de pénalité donné par la formule suivante :

$$P(C) = \sum_{i=1}^t (C_i - Q_i)^2$$

Donc le score finale de la fonction objective sera le suivant :

$$F(C) = U(C) - P(C)$$

## 4 Optimisation combinatoire :

### 4.1 Définition :

L'optimisation est une branche des mathématiques qui permet de résoudre des problèmes en déterminant le meilleur élément d'un ensemble selon certains critères prédéfinis. De ce fait, l'optimisation est omniprésente dans tous les domaines et évolue sans cesse depuis Euclide. Un problème d'optimisation en général est défini par un espace de recherche  $S$  et une fonction objective  $f$ . le but est de trouver la solution  $s^* \in S$  de meilleure qualité  $f(s^*)$ . Suivant le problème posé, on cherche soit le minimum, soit le maximum de la fonction  $f$  [10].

### 4.2 Approches d'optimisation combinatoire (approches de sélection) :

Plusieurs efforts de recherches ont été rapportés dans la littérature sur la sélection automatique de service [10]. Les travaux existants sont scindés en deux types [11] :

#### 4.2.1 L'optimisation multi-objective :

Appelée aussi l'optimisation à base de skyline (the skyline based optimization) peut être manipulé en employant les méta-heuristiques ou les techniques de base de données [12]. Plus spécifiquement nous pouvons employer l'algorithme de clivage et conquérir, l'algorithme bitmap, l'algorithme basé par index (B tree, Hash table), et l'algorithme de plus proche voisin (R tree).

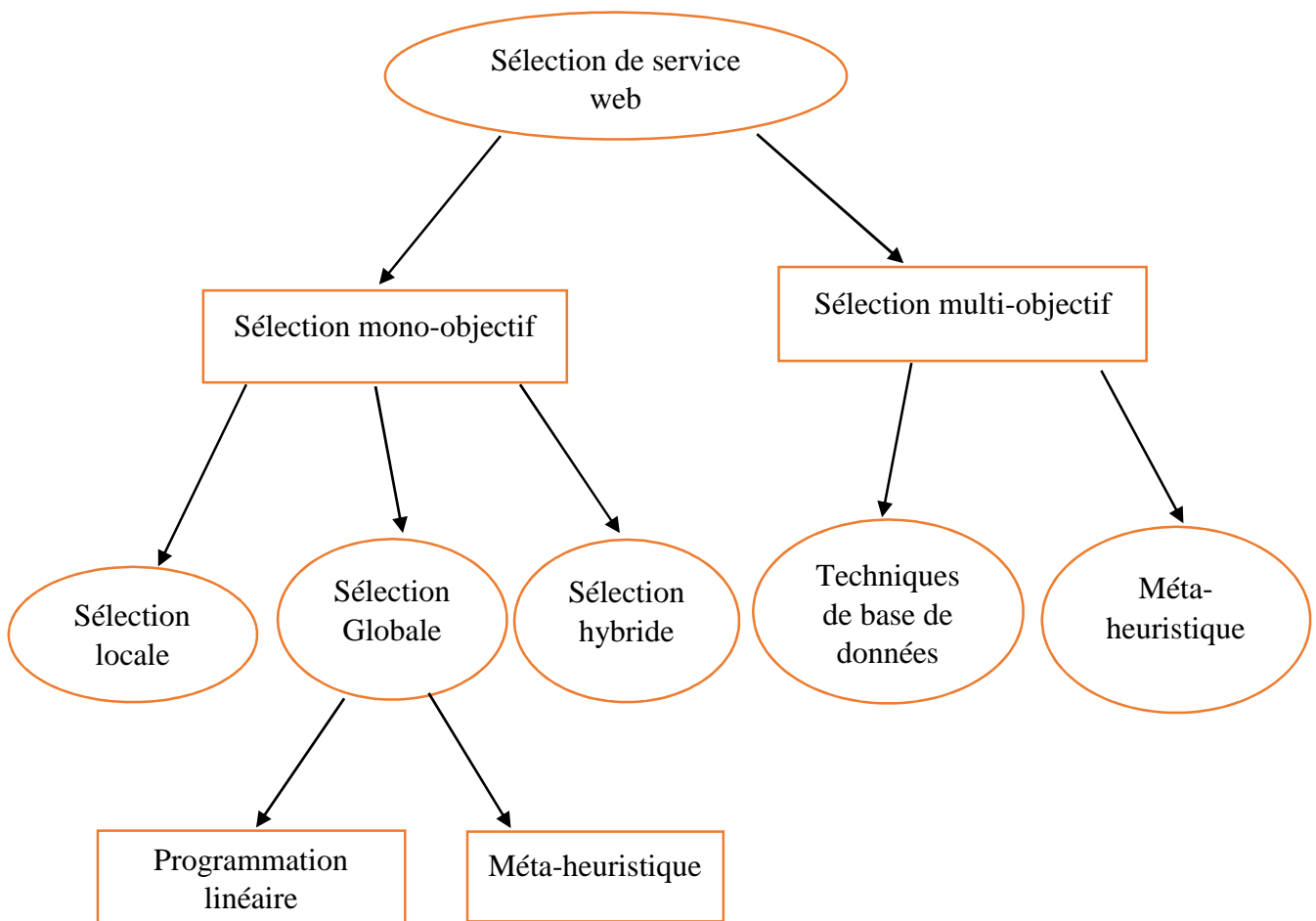
D'ailleurs, il y a plusieurs travaux qui tient compte des préférences d'utilisateur pour choisir les  $K$  top horizons dominants [11 ; 12] certains d'entre eux utilisent la théorie des ensembles flous pour modeler les préférences et le rapport de dominance, les autres emploie le concept de Pareto-dominance pour ranger les services de web.

#### 4.2.2 L'optimisation mono-objective :

Comporte plusieurs approches, elle peut employer un modèle de sélection global [13 ; 14 ; 15] ou un modèle de sélection local [15] ou un modèle de sélection hybride [10].

- Le modèle de sélection global se concentre sur toute la composition (i.e. les n composants de la solution), il peut obtenir la solution optimale, mais il a une complexité exponentielle.
- Néanmoins, le modèle local a seulement une complexité linéaire mais ne peut pas manipuler les contraintes globales (il manipule seulement des contraintes locales).
- Le troisième modèle est un compromis des deux approches, il commence la recherche par une optimisation globale, puis il continue le travail avec une optimisation local, sa complexité temporelle est inférieure par rapport à l'optimisation globale, et il peut également manipuler les contraintes globales.

La figure suivante montre l'état de l'art des différents travaux effectués sur le problème de sélection de service web.



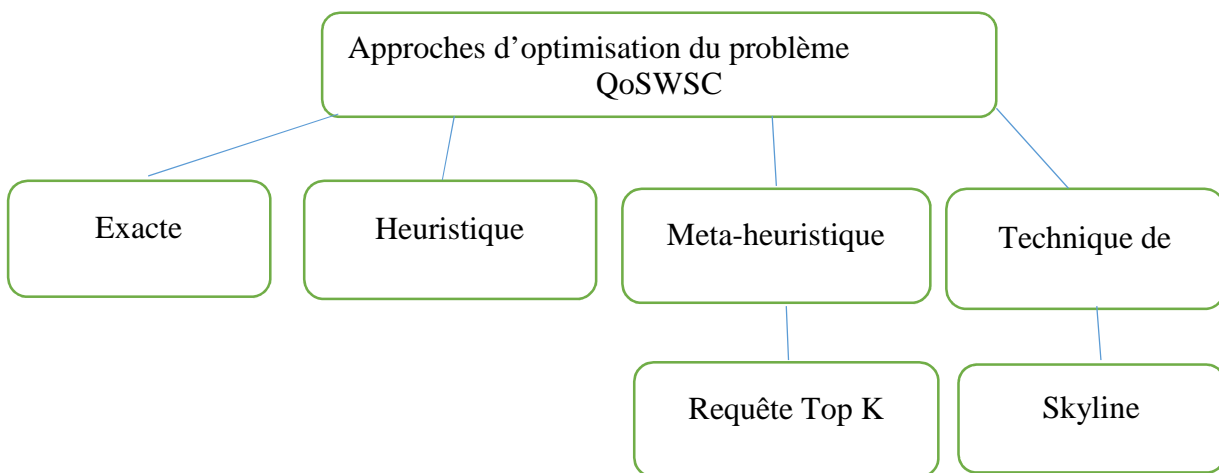
**Figure I.6: Les approches de Sélection**

## 5 État de l'art des travaux sur le problème QoSSWC :

Le but de la sélection des services web est de trouver le service ou les services qui satisfassent les besoins de l'utilisateur et pour cela plusieurs travaux ont vu le jour pour résoudre ce problème. Ces travaux sont classés en 3 grandes classes distinctes selon leur stratégie, paramètre utilisé et résultat obtenue.

Dans cet état de l'art, nous nous concentrons sur les travaux d'optimisation de ce problème (QoSWSC) de premier rang, 4 catégories peuvent être considérées dans la résolution du problème de genre : [24]

1. Technique de base de données.
2. Exacte.
3. Heuristique (approximative).
4. Meta-heuristique (approximative).



*Figure I.7: Approches de résolution de problème de sélection de SW [24]*

### 5.1 Technique de base de données :

La sélection des services web est traité comme un problème multi-objectif avec cette stratégie, qu'il soit traité en local ou en global (sans le réduire à une fonction mono-objectif).

Elle repose surtout sur la notion de Pareto dominance et sur le principe des requêtes Skyline ou Top-k. ont utilisant des algorithmes pareil a Divide and conquer ou autres.

## 5.2 Approches Exactes :

Toujours dans l'obtention de la solution optimale d'un problème traité, les méthodes exactes assure cet objectif avec un parcours complet sur l'ensemble de l'espace de recherche afin de retirer toutes les solutions qui peuvent être bien meilleur que la solution optimale courante.

## 5.3 Approche heuristique (Approximative) :

Une méthode heuristique est une méthode de résolution de problème qui ne s'appuie pas sur une analyse détaillée ou exhaustive du problème. Elle consiste à fonctionner par approches successives en s'appuyant, par exemple, sur des similitudes avec des problèmes déjà traités afin d'éliminer progressivement les alternatives et ne conserver qu'une série limitée de solutions pour tendre vers celle qui est optimale.

Les heuristiques prennent beaucoup plus moins de temps pour trouver la solution optimale par rapport au Méthode exacte.

## 5.4 Approche Méta-heuristique :

Les méta-heuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction, par échantillonnage d'une fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème afin d'en trouver une approximation de la meilleure solution (d'une manière proche des algorithmes d'approximation).

## 5.5 Les méta-heuristiques à base de solution unique :

Cette catégorie initialise la recherche avec une solution unique puis elle l'améliore au cours d'une série d'itération en se basent sur la notion du voisinage. La solution initiale subit des modifications selon son voisinage afin d'améliorer progressivement sa qualité.

Nous trouvons dans ces classe deux familles d'algorithmes de recherche locale :

- Pour une optimisation locale : recherche locale simple (la descente).
- Pour une optimisation globale : Recuit simulé.



### 5.6 Les méta-heuristiques à base de population de solution :

Cette catégorie initialise la recherche avec ensemble de solutions afin d'obtenir la meilleure (solution optimal) qui est la solution du problème traité. L'utilisation d'un ensemble de solutions augmente la possibilité d'arriver à une solution de bonne qualité.

Catégories de cette classe :

- Les algorithmes évolutionnaires.
- Les algorithmes génétiques.
- Algorithmes à base d'intelligence par essaims : Colonies de fourmis, Fruit Fly.

## 6 Conclusion :

Nous avons présenté dans ce chapitre les services web et les différentes compositions en se basant sur la qualité de service.

Nous avons défini les services web et les critères de qualité de service QoS, la fonction objective et l'optimisation combinatoire avec les différents types et les différentes catégories qui nous permettent de classer notre problématique.

Nous présenterons dans le chapitre suivant notre travail qui concerne la sélection de service web basée sur l'algorithme de fruit Fly.

**Chapitre II :**  
**Sélection de**  
**service web basée**  
**sur l'Algorithme**  
**de la FOA**

## 1 Introduction :

L'objectif de ce chapitre est de décrire le modèle de notre approche pour sélectionner le meilleur fournisseur d'un service web.

Dans ce chapitre, nous proposons notre algorithme de sélection et sa méthode de fonctionnement.

FOA (Fruit Fly Optimization Algorithm) est une nouvelle méthode pour trouver l'optimisation globale sur la base de fruit Fly

## 2 WS-FOA : Web Service composition based on Fruit Fly Optimization Algorithm (la composition de services web basée sur l'algorithme d'optimisation Fruit Fly) :

### 2.1 Définition :

FOA (Fruit Fly Optimization Algorithm) est une nouvelle méthode pour trouver l'optimisation globale d'une composition de service. [16]

La Recherche fruit Fly est basée sur un mouvement par essaims, l'équipe de chercheurs, Yiwen Zhang, Guangming Cui, Yan Wang, Xing Guo et Shu Zhao a traduit le phénomène à un processus en 2015. [17]

C'est une très récente Meta-heuristique à base d'une population de solution.

### 2.2 Comportement de Fruit Fly :

La Recherche de nourriture de fruit Fly va prendre le scénario suivant :

Le chemin d'alimentation d'un animal est effectivement une marche aléatoire.

Dans notre exemple, on va suivre le protocole naturel de fruit Fly qui est dans un emplacement aléatoire et qui va voler vers la nourriture dans plusieurs directions sur la base de la vision et la meilleur distance, on va suivre la même chose dans notre algorithme et on doit choisir le meilleur déplacement.

## 2.3 La modélisation de la composition de service :

### 2.3.1 Définition 1 Structure QoS :

Soit le vecteur de qualité QoS = {A, T, C, R} (1), où A représenté la disponibilité, T représente le temps de réponse, C représente coût, et R représente la réputation. [18]

#### (1) Disponibilité (A) :

La disponibilité fait référence à la probabilité qu'une bande de service web est accessible, à savoir la proportion de visites de service Web avec succès dans la période de temps t

Soit  $A = t_s / t$  (2), où le  $t_s$  représente le temps d'accès avec succès du service dans le délai période t. A est un nombre dans l'intervalle [0, 1].

#### (2) Temps de réponse (T) :

Le temps de réponse correspond au délai prévu entre le moment où un demandeur de services envoie une demande de service et le moment où le résultat est obtenu.

Il comprend principalement le temps d'exécution de service Web  $T_e$ , le temps de transmission du service  $T_t$ , et d'autre consommation de temps  $T_o n$ . Par conséquent, le temps de réponse est évalué comme suite :

$$T = T_e + T_t + T_o n. \quad (3)$$

#### (3) Coût (C) :

Coût fait référence à la taxe qu'un demandeur de service doit payer au fournisseur de service pour l'appel de service.

Il comprend principalement le frais de requis de base  $C_s$ , tels en tant que logiciel (SaaS), le matériel (IaaS), et la plate-forme ( $P$ ) et les frais de gestion des services  $C_m$ . Il peut être décrit comme  $C = C_s + C_m$  (4)

#### (4) Réputation (R) :

Réputation fait référence à la mesure fiable du service web, qui est principalement basé sur l'expérience d'utilisateurs après avoir utilisé le service Web. Différents utilisateurs peuvent avoir des évaluations différentes sur le même service Web.

$$R = \sum_{i=1}^n R_i / n \quad (5)$$

Où  $R_i$  représente l'évaluation de services web de l' $i$ ème utilisateur final.

$R$  est un nombre dans l'intervalle  $[0, 1]$  et  $n$  est le nombre d'utilisateurs.

Selon la fonction objective citée au **chapitre I** nous avons identifié ces différents paramètres QoS dans notre méthode car ces paramètres ont un grand impact sur l'assistance aux demandeurs pour des sélections raisonnables. En outre, ces paramètres sont les qualités fondamentales pour remplir les objectifs du service Web. Ceux-ci peuvent être divisés en deux groupes :

□ **Le premier groupe** des valeurs d'attribut QoS négatifs ont besoin d'être minimisées, c'est-à-dire plus la valeur du temps de réponse ou le coût est inférieur meilleur est la qualité.

□ **Le deuxième groupe** des attributs QoS positifs qui ont besoin être maximisé tel que disponibilité, fiabilité et réputation.

Pour résoudre un problème d'objectif de maximisation, il faut multiplier les valeurs négatives par (-1) pour faire face à l'hétérogénéité des critères de QoS.

### 2.3.2 Définition 2 Structure Service (s) Web :

Service Web est un quatre-tuple. Soit  $s = \{ID, Source, Fonction ; QoS\}$  (6), où ID est l'identification unique d'un service Web, Source est l'information décrivant du nom du service et de l'éditeur, etc., une fonction est une description fonctionnelle du service Web et QoS est la description de la qualité du service Web.

### 2.3.3 Définition 3 Structure Composition de Service (SC)

Soit  $SC = (S_1, S_2, S_3, \dots, S_{n-1}, S_n)$  où  $S_1 \dots S_n$  représenter chaque service Web respectivement,

En même temps,  $S_1$  à  $S_2$  sont appelés ensembles de services de composition de service, services en  $S_i$  ont la même fonction, mais ont des qualités de service différentes. Ainsi, il est conclu que  $SC \in S_1 \times S_2 \times S_3 \times \dots \times S_{n-1} \times S_n$

Les chemins d'exécution pour la composition de service peut être représenté par quatre modèles, telles que séquentielle, sélection, parallèle, et la boucle (voir Fig. 2) [20].

Les formules de tous les motifs sont les suivants :

(1) modèle séquentiel (Figure II.2 (a)). QoS est trouvée par Eqs (7).

$$\left\{ \begin{array}{l} A = \prod_{i=1}^n A_i ; \\ R = \sum_{i=1}^n R_i / n ; \\ C = \sum_{i=1}^n C_i ; \\ T = \sum_{i=1}^n T_i \end{array} \right. \quad (7)$$

(2) le modèle de sélection (Figure II.2 (b)). Supposons que la probabilité sélectionnée de chaque service  $S_i$  est  $\lambda_i$ , puis  $\sum_{i=1}^n \lambda_i$  Ensuite, la qualité de service QoS est trouvé par Eqs (8).

$$\left\{ \begin{array}{l} A = \prod_{i=1}^n (A_i \times \lambda_i) ; \\ R = \sum_{i=1}^n (R_i \times \lambda_i) ; \\ C = \sum_{i=1}^n (C_i \times \lambda_i) ; \\ T = \sum_{i=1}^n (T_i \times \lambda_i) \end{array} \right. \quad (8)$$

(3) le modèle parallèle (Figure II.2 (c)). Supposons n services sont exécutée en parallèle. Ensuite, la qualité de service est trouvée par Eqs. (9).

$$\left\{ \begin{array}{l} A = \prod_{i=1}^n A_i ; \\ R = \sum_{i=1}^n R_i / n ; \\ C = \sum_{i=1}^n C_i ; \\ T = \max (T_i), i \in [1, n] \end{array} \right. \quad (9)$$

(4) modèle de boucle (Figure II.2 (d)). Supposons que le modèle de la boucle est effectuée  $\Theta$  fois. Ensuite, la qualité de service QoS est trouvée par Eqs. (10).

$$\left\{ \begin{array}{l} A = \prod_{i=1}^n A_i ; \\ R = \sum_{i=1}^n R_i / n ; \\ C = \theta \times \sum_{i=1}^n C_i \\ T = \theta \times \sum_{i=1}^n T_i \end{array} \right. \quad (10)$$

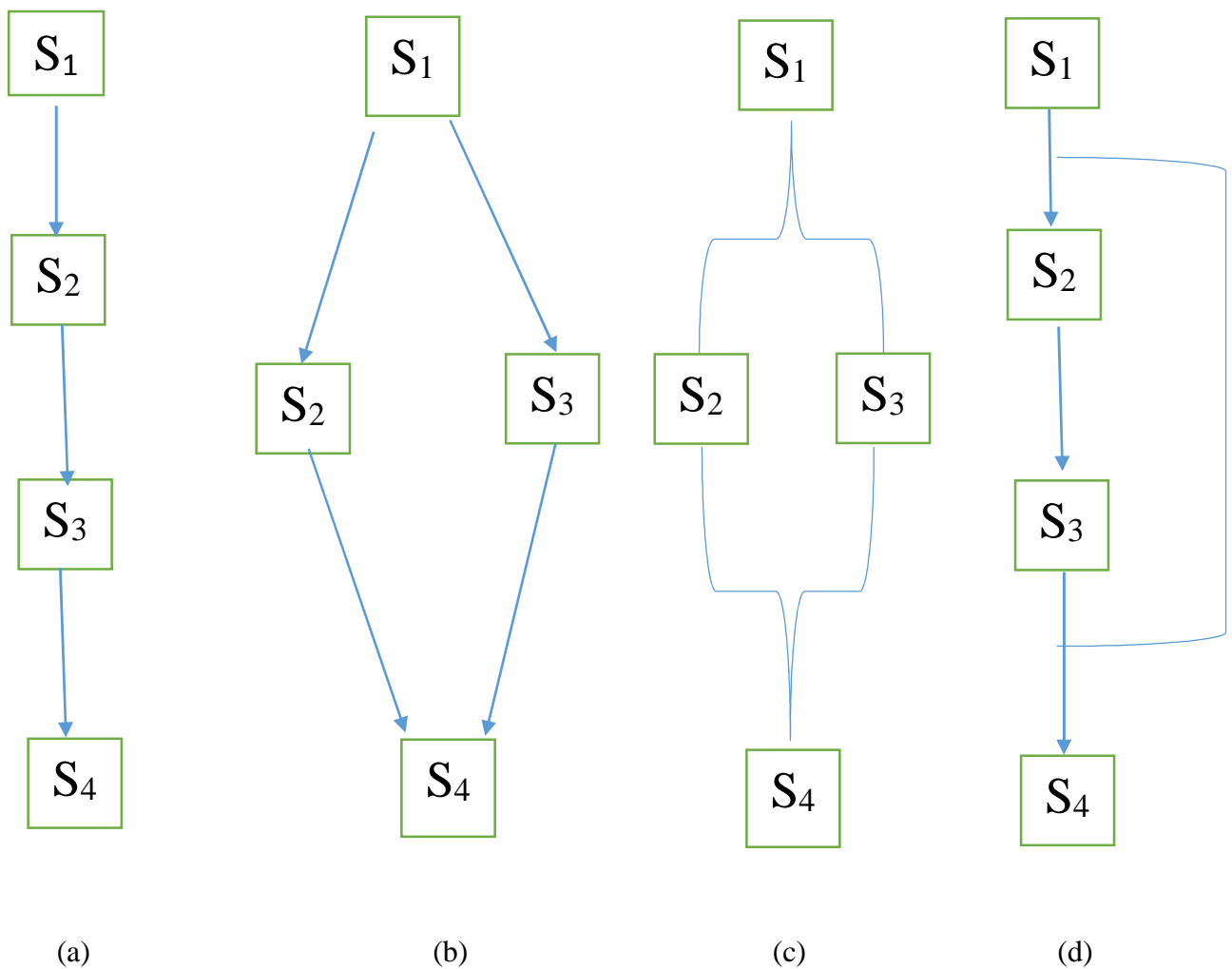


Figure II.1: Quatre modèles pour la composition de service.

Notre exemple utilise le modèle séquentiel.

### 2.3.4 Définition 4 QoS prétraitement

Selon une fonction d'utilité certaine (habituellement il est normalisé à [0,1]).

Les indices discrets et normalise en fonction de la formule suivante.

$$q_i = \begin{cases} (q_j^m - q_i) / (q_j^m - q_j^m), & \text{Pour les Critères à maximiser ;} \\ (q_i - q_j^m) / (q_j^m - q_j^m), & \text{Pour les Critères à minimiser ;} \end{cases} \quad (11)$$

Où  $q_j^m$  e  $q_j^m$  représentent le maximum actuel et la valeur minimale de l'indice d'évaluation respectivement. Leur valeur va changer dynamiquement avec l'ajout ou retrait d'un service Web.

### 2.3.5 Les étapes de l'algorithme de prétraitement QoS :

Entrée : Les informations d'index d'origine du jeu de services

Sortie : Les informations d'index normalisé de jeu de services

Étape 1 : Obtenir les informations d'index de chaque ensemble de services.

Étape 2 : Obtenir la valeur maximale et minimale, la valeur de chaque indice dans l'ensemble des services

Étape 3 : Pour chaque enregistrement de service, A et R indices sont calculés à l'aide Eqs. (11) selon l'équation de la prestation, et C et T indices sont calculés en utilisant les équations. (11) selon l'équation du coût.

Étape 4 : Répétez les étapes 2-4, jusqu'à ce que chaque ensemble de services soit totalement éliminés.

Étape 5 : sortie les informations d'index de chaque ensemble de services après la normalisation.

On peut formaliser la fonction Objectif Fct (QoS) comme suit :

Fct (QoS) (S : Service, Q : Qualité)

{

SC1=SC2=SC3=SC4=SC5=0 ; Q max : la qualité maximal de chaque type de qualité de service, Q min : la qualité minimal de chaque type de qualité de service.



**Pour i=1 jusqu'à n faire**

Si = (Q1Si, Q2Si, Q3Si, Q4Si, Q5Si) // on applique les Cinq qualité de service sur chaque service.

Q1=  $Q1 - Q_{\min} / Q_{\max} - Q_{\min}$  ; // normaliser le temps d'exécution.

Q2=  $Q2 - Q_{\min} / Q_{\max} - Q_{\min}$  ; // normaliser le coût.

Q3=  $Q_{\max} - Q3 / Q_{\max} - Q_{\min}$  ; // normaliser la disponibilité.

Q4=  $Q_{\max} - Q4 / Q_{\max} - Q_{\min}$  ; // normaliser la fiabilité.

Q5=  $Q_{\max} - Q5 / Q_{\max} - Q_{\min}$  ; // normaliser la réputation.

SC1=  $SC1 + Q1Si$  ; // minimiser le temps d'exécution.

SC2=  $SC2 + Q2Si$  ; // minimiser le coût.

SC3=  $SC3 * Q3Si$  ; // maximiser la disponibilité.

SC4=  $SC4 * Q4Si$  ; // maximiser la fiabilité.

SC5=  $SC5 * Q5Si$  ; // maximiser la réputation.

**Fin Pour ;**

Si  $SC1 < 0$  alors  $SC1 = SC1 * (-1)$  ; // multiplier des valeurs négatifs par (-1)

Si  $SC2 < 0$  alors  $SC2 = SC2 * (-1)$  ; // multiplier des valeurs négatifs par (-1)

$SC = SC1 + SC2 + SC3 + SC4 + SC5$  ;

j=5 ;

**Répéter**

Si  $SC_j < SC_{j-1}$  alors  $SC = SC_{j-1}$  ;

Sinon  $SC = SC_j$  ; // on prend la valeur maximal

**Jusqu'à j=2 ;**

**// On sélectionne le service qui a la plus grande valeur de qualité SC maximal**

}

**2.4 Modélisation de l'Algorithme :****2.4.1 Exemple de changement en position d'un fruit Fly et le Fruit Fly individuels :**

Dans la nature, les animaux recherchent de la nourriture de manière aléatoire. Généralement, le chemin d'alimentation d'un animal est effectivement une marche aléatoire parce que le prochain déplacement est basé à la fois sur l'emplacement / l'état actuel et sur le phénomène étudié et représenté par l'équation de mouvement (F) que l'on peut définir par la direction et la taille selon laquelle l'essaim de Fruit Fly peut

déplacer de façon optimale. Autrement dit,  $F = V \times t$ , ( $t \in [-1,1]$ ).  $V$  désigne l'étape mobile, ce qui signifie la plage où Fruit Fly pourrait se déplacer une fois. Ainsi, l'équation de Fruit Fly individuel est la suivante. [19]

$$P_k^{i+1,j} = P_k^{i,j} + F \quad (12)$$

$P_k^{i+1,j}$  Représente l'emplacement de k-ème Fruit Fly de la j-ème essaim voler dans (i+1) – ème processus itératif.

$P_k^{i,j}$  Représente l'emplacement optimal de la j-ème essaim dans le processus itératif i-ème.

#### 2.4.2 Définition de la distance :

Dans chaque itération de FOA pour chaque fruit Fly, utilisez la réciproque du décalage qui correspond à l'emplacement du fruit Fly voler dans son ensemble de services. Après cela, la valeur est calculée en utilisant la relation entre distance et valeur Small.

La distance (Dist) et la valeur (Small) sont définies comme suit :

$$\left\{ \begin{array}{l} D_k^{i+1,j} = 1/P_k^{i+1,j} ; \\ S_k^{i+1,j} = 1/D_k^{i+1,j} \end{array} \right. \quad (13)$$

$P_k^{i+1,j}$  Est l'endroit où le fruit Fly individuel a déplacé dans son ensemble de services

$D_k^{i+1,j}$  Désigne la distance du k-ième fruit Fly, d'essai j-ème dans le processus itératif i+1-ième

$S_k^{i+1,j}$  Représente la valeur de la k-ème fruit Fly de la j-ème essaim volé dans le (i+1)-ième processus itératif.

#### 2.4.3 La fonction Fitness :

La fonction Fitness est utilisée pour calculer l'aptitude de chaque SC. La sélection de chaque SC dépend de cette valeur avec les résultats de la qualité de service obtenue par la fonction objectif en se basant sur les équations (11), la fonction de Fitness peut être définie comme suite :

$$\text{Fitness} = \sum_{j=1}^n \sum_{k=1}^m C_j W_k Q_j = \sum_{j=1}^n \sum_{k=1}^m C_j W_k S_k^{i,j} \quad (14)$$

Où  $S_{k,j}$  représente k-ième composant indice de Fruit Fly d'essaim j-ème dans le i-ème itérative processus

.  $C_j$  Représente le poids du service j-ème dans le SC

.  $Q_j$  Représente le k-ième composant indice de j-ième service

$W_k$  Représente le poids de la k-ème indice de composant.

n désigne le nombre de service

m signifie que chaque service contient m Indice de qualité de service.

#### 2.4.4 Principe de Fruit Fly :

Un service composé généré est constitué d'un ensemble de service, [6] et chaque'un appartient à un essaim et notre traitement pour trouver le fruit Fly optimal est réalisé sur chaque essaim.

Initialisation au hasard de l'emplacement d'essaim de fruit Fly (Xaxis, Yaxis) où

Xaxis : Indice de la classe X, Yaxis : Indice de service Y dans sa classe X.

On prend en considération la fonction objective de qualité de service pour choisir le meilleur service Fct (QoS).

La position de nourriture est calculée par la distance, la fitness et l'essaim va utiliser la vision pour voler a une nouvelle position.

La meilleure position est de fitness maximal.

### 2.4.5 Les étapes de l'algorithme WS FOA :

Les principales étapes et leur implémentation sont décrites ci-dessous [16] :

Entrée : le nombre de classe, l'ensemble normalisé de services par classe, le nombre de compositions, le nombre d'itérations.

Sortie : l'emplacement du fruit Fly individuel optimal

Étape 1 : Initialisation de l'emplacement ( $P^{E,1}$  à  $P^{E;n_b}$ ) de fruit Fly dans chaque ensemble de services.

Étape 2 : Calculer l'emplacement du fruit Fly essaim selon l'équation. (12).

Étape 3 : Calculez la distance et la valeur de Smell Fruit Fly individuels dans chaque ensemble de services selon Eq. (13).

Étape 4 : calculer Fitness dans chaque enregistrement de SC.

Étape 5 : Calculer chaque enregistrement de SC selon l'Éq. (14). enregistrez le SC optimale dans le processus itératif, et traiter chaque service dans l'enregistrement combiné fruit Fly voler  $P_i$  ;  $j_{best}$  dans son essaim correspondant.

Comparer la valeur optimale dans cette itération avec la valeur optimale globale et réserve le meilleur emplacement.

Étape 6 : Répétez les étapes 2-5 jusqu'à ce que l'itération soit terminée ou la demande de précision est satisfaite.

Étape 7 : sortie la composition optimale.

Dans chaque itération, on a pris des déplacements de l'essaim, et Chaque déplacement pour sélectionner une composition de service parmi plusieurs, le but est de choisir la meilleure composition, l'algorithme génère à chaque fois une composition et on doit calculer la meilleure solution.

Nous proposons ici une version simplifiée de l'algorithme :

**Entrées :**

$C_g$  : contrainte global,  $P(i)$  : pénalité,

$L$ : le nombre de classes de services,  $m$  : le nombre de services par classe.

$N$  : le nombre de compositions par itération,  $n$  : nombre de qualité de service QoS.

$Max\ gen$ : le nombre d'itérations

$It = 1$  ;

**Début****Tant que ( $It \leq Max\ gen$ ) faire**

Initialisation au hasard de l'emplacement d'essaim de fruit Fly

Générer meilleur service composite SC par la fonction objective Ftc (QoS) (meilleur service (i)) : Ftc (QoS) ;

$F = V \times t$ , ( $t \in [-1,1]$ ) ; // calculer le mouvement de déplacement.

$P_K^{i+1,J} = P_K^{i,J} + F$  ; // l'emplacement de Fruit Fly

$D_K^{i+1,J} = 1 / P_K^{i+1,J}$  ; // la distance de Fruit Fly

$S_K^{i+1,J} = 1 / D_K^{i+1,J}$  ; // la valeur de Fruit Fly

$Fitness = \sum_{j=1}^n \sum_{k=1}^m C_j W_k S_K^{i,J}$  ; // Calculer la fitness

Best Fitness best indexe = max (Fitness i); // Remplacer  $F_i$  par  $F_{i+1}$  est la fitness maximal  $F_i$  si  $F_{i+1} > F_i$

$It = It + 1$ ;

**Jusqu'à nombre itération > max gen ou critère d'arrêt**

$Fitness\ best = best\ Fitness$  ; // Trouver la meilleure composition SC( $S_i$ ). Si la Fitness meilleure que celle de l'itération précédente la remplacer

$P(C) = 0$  ;

Pour  $j = 1$  jusqu'à 5 faire

$P(C) = P(C) + ((C_g(j) - Q(j)) * (C_g(j) - Q(j)))$  ; // pénalité par rapport à contrainte global donnée

Fin pour

**Fin.**

Si on prend les étapes de notre travail, avec l'application de la fonction objective  $f(QoS)$  et l'algorithme de fruit Fly, on suit la démarche suivante :

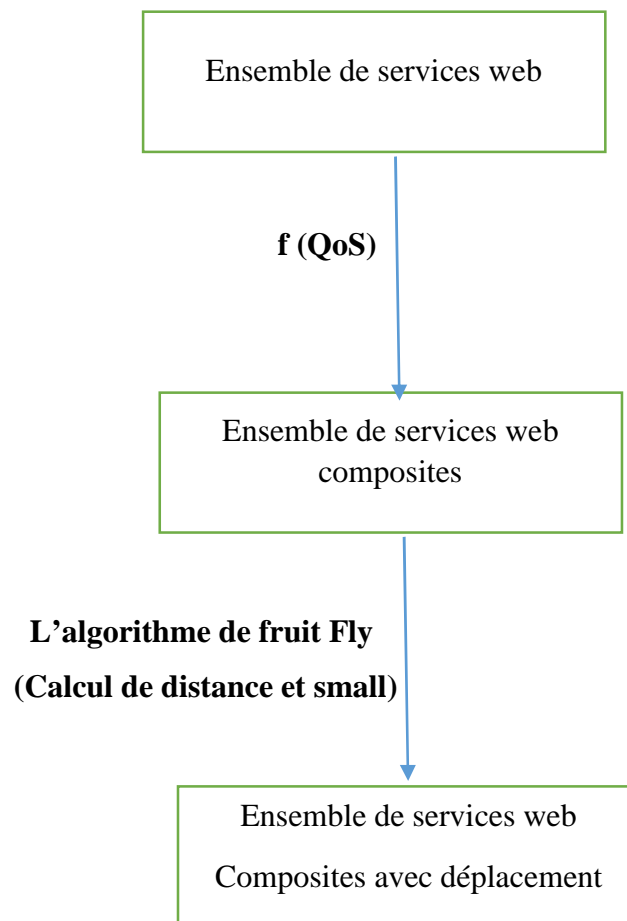


Figure II 2 : les étapes de l'ensemble de services web par fruit Fly

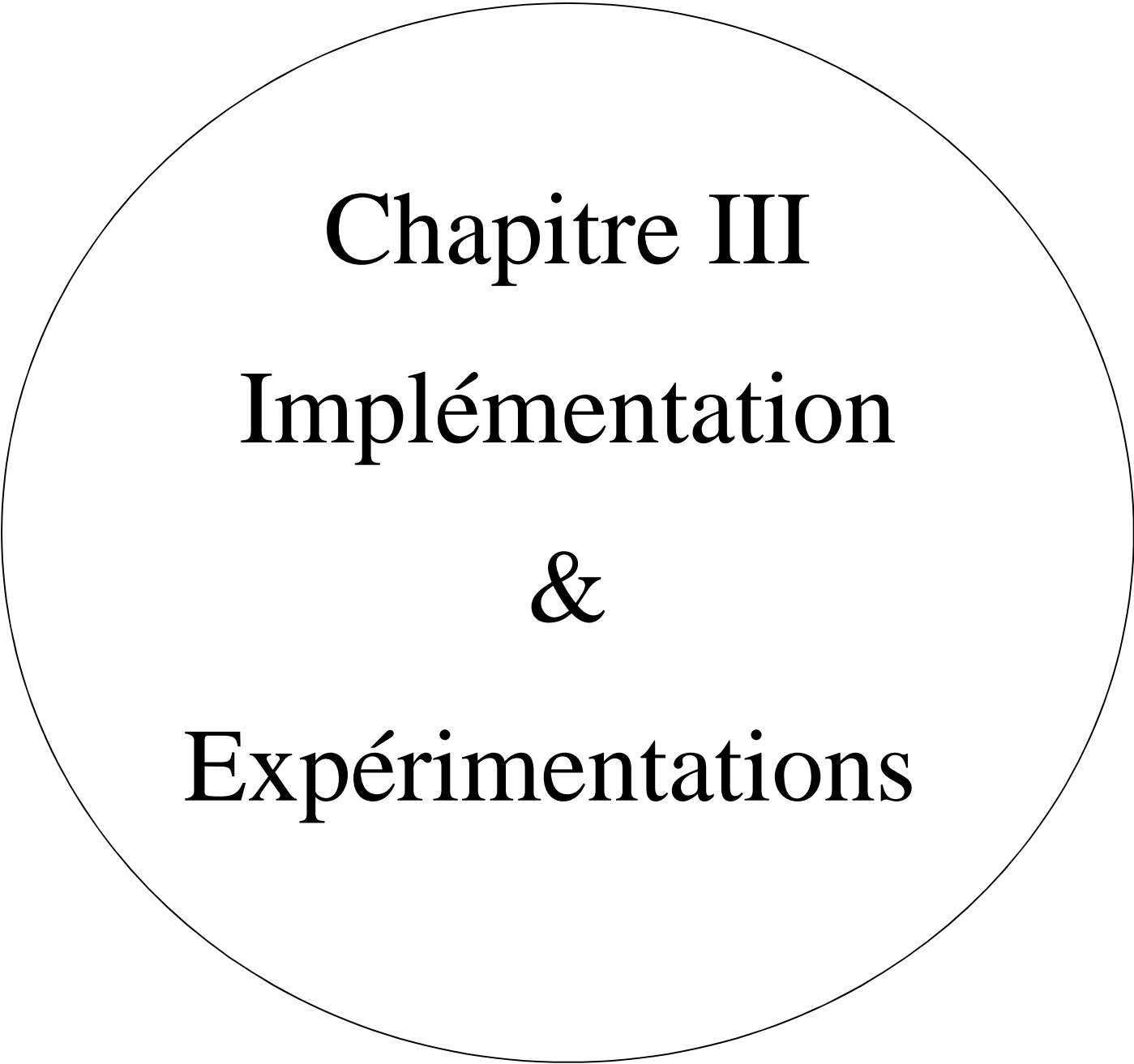
### 3 Conclusion :

Ce chapitre a été consacré à la définition et le comportement de Fruit Fly, à modéliser la composition de service et notre algorithme WS-FOA.

Après la modélisation de composition de service, on a montré la fonction de QoS (la fonction objective  $f(QoS)$ ) par un simple algorithme.

Ensuite on a défini le déplacement, la distance, la valeur de fruit Fly Small et la fonction fitness afin de présenter l'algorithme WS-FOA.

Dans le chapitre suivant on va présenter l'implémentation de notre application et donner une expérimentation.



Chapitre III  
Implémentation  
&  
Expérimentations



## 1 Introduction :

Dans ce qui suit nous présentons et décrivons notre prototype implémentant l'environnement de sélection d'une composition de services web, commençant par la description des différents composants de notre système avec leurs rôles, passant aux corpus utilisés pour évaluer l'environnement, et terminant avec les expérimentations appliquées sur le système en discutant chaque résultat obtenu.

## 2 Présentation de l'environnement de développement :

Nous avons implémenté et développé notre prototype sous Netbeans IDE 8.2 avec java (JDK 1.8.0) dans une machine HP de 8 GO de RAM qui a Windows 7 comme système d'exploitation.

## 3 Outils de système et paramètres :

Les paramètres d'entrée sont présentés dans L'interface de notre prototype montré dans la figure suivante :

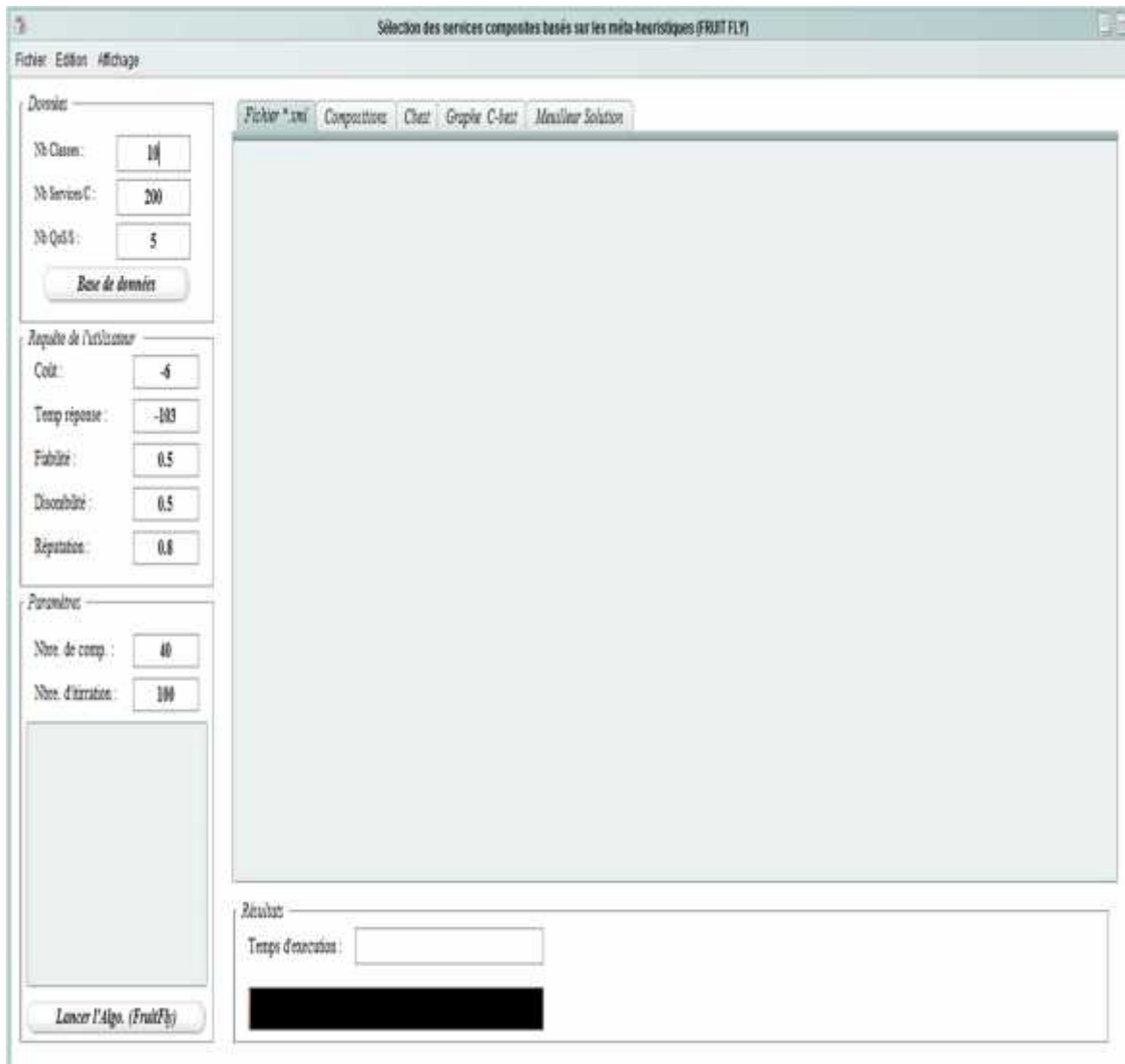


Figure III.I : Interface principale

## 4 Interface (paramètre et résultat) :

Nous avons construit notre interface d'une manière simple afin de faciliter la manipulation des paramètres, et que les résultats soient compréhensibles.

### 4.1 Générer et charger la base :

Nos tests sont effectués sur une base de données synthétique dans un format XML par chargement de fichier XML

On obtient les classes avec leurs services et les critères de QoS

id classe	N° service	Coût	TempsRep	Fiabilité	Disponibilité	Réputation
Classe1	Service1	-13.887937726039375	-212.2596670770839	0.6572125193881031	0.11486488194743248	0.39344276669497506
Classe1	Service2	-24.54101117090762	-153.227312700959386	0.16247895340919488	0.11387670835934694	0.3481141020078069
Classe1	Service3	-26.4396828789918	-89.51847649802339	0.6681055546620104	0.1158467590468572	0.8691904270803448
Classe1	Service4	-19.364892983541925	-207.86716321917204	0.265495553392148	0.2099800803699787	0.7066973946033446
Classe1	Service5	-22.89755221756205	-271.7461759051097	0.36309874256185914	0.12733678926390743	0.04050502954631383
Classe1	Service6	-9.016354906056709	-280.3815669338367	7.350377565086666E-4	0.3222062017222314	0.27519153448216306
Classe1	Service7	-10.146918485634345	-214.05941218366277	0.23718784067024815	0.806627076799531	0.9159437676219527
Classe1	Service8	-39.126180026623824	-21.95627896147284	0.03295091113683308	0.06638886645875847	0.159164485205851927
Classe1	Service9	-7.29638407128625	-257.19163702066453	0.20794130583428982	0.2557998000125079	0.4407403820263719
Classe1	Service10	-19.7628532505795	-154.64306830074008	0.15885519829923608	0.05242315979981519	0.08918532908965252
Classe1	Service11	-23.942167670987553	-60.393753789500146	0.12044911870599418	0.15749427290602344	0.50987799686695994
Classe1	Service12	-27.380610520918612	-90.95520881771203	0.6494372103532601	0.04441439440234418	0.06770202223804123
Classe1	Service13	-9.272457804540995	-50.07974388705493	0.663198712532167	0.26840124996079825	0.4248914257666873
Classe1	Service14	-22.24269084201867	-261.98097157790085	0.28152901973190114	0.3434400082141379	0.7240258083461463
Classe1	Service15	-8.358399005945083	-14.488816524835535	0.23183098018361287	0.3220043353178713	0.2515177371391439
Classe1	Service16	-27.840883861120922	-294.234946497288	0.3738658922822336	0.356168342270053	0.18943100434059943
Classe1	Service17	-0.029603271087394354	-131.18624577564937	0.4091028819117865	0.09455189564122306	0.40893052573485
Classe1	Service18	-10.10682841952741	-117.97378266004849	0.24614062209452675	0.0674597188496586	0.429796886377859
Classe1	Service19	-29.998825072818367	-279.8574386776034	0.4534274521368584	0.2884283184325037	0.4261210637637655
Classe1	Service20	-16.018615962257734	-178.88695948928165	0.39636000557212453	0.30183940679014487	0.09993818414117828
Classe1	Service21	-0.1815207056679912	-242.60936125150047	0.012583779796953461	0.04082069443537949	0.0849933378043802
Classe1	Service22	-1.4689174619577801	-122.16235418187021	0.23394382790013166	0.14570136987193	0.7831068617200017
Classe1	Service23	-4.402478853007308	-25.935778978845094	0.057295280272820444	0.2866543933680384	0.8268003747729148
Classe1	Service24	-29.413053280110347	-24.740848545065653	0.3686369389833474	0.29527777527777543	0.948375477469446

Figure III.1: Chargement d'un fichier XML

On génère la composition comme suit :

The screenshot shows the 'Sélection des services composés basés sur les méta-heuristiques (FRUITFLY)' application. The interface is divided into several sections:

- Données:**
  - No Classes: 7
  - No Services C: 120
  - No QoS: 5
  - Base de données
- Requis de l'utilisateur:**
  - Coût: -4
  - Temps réponse: -100
  - Fiabilité: 0.5
  - Disponibilité: 0.5
  - Réputation: 0.8
- Paramètres:**
  - Nbre. de comp.: 40
  - Nbre. d'itérations: 100
- Tableau des services:**

N°	X <sup>1</sup>	Y <sup>1</sup>
1	0.78445830988	0.61099792873
2	0.5808099828	0.2484787948
3	0.58184054888	0.81815629887
4	0.8427958786	0.37712786022
5	0.88372847281	0.72824888872
6	0.8894848888	0.38282261594
7	0.72961841387	0.88887174889
8	0.48188914288	0.42287888883
9	0.88788484888	0.82482788884
- Tableau des services composés:**

Id class	N° service	Coût	TempsRap	Fiabilité	Disponibilité	Réputation
Classel	Service7	-25.833681832018485	-227.1435569458486	0.0386581171407088	0.0839772289328202	0.0854853153553884
Classel	Service4	-25.814375128014734	278.31370660699054	0.25131748908121654	0.2917926668130774	0.51580018119882
Classel	Service1	-7.0886339333007638	-31.805695288488195	0.345049620632091	0.18139432988974553	0.7078467761430006
Classel	Service15	-14.728193088888787	-280.0712703408829	0.8064324576887113	0.27975078688480416	0.056794681177791238
Classel	Service8	-4.07418103167265	224.34528570209558	0.5380518823994722	0.0014031618214742308	0.0807983467977906
Classel	Service9	-25.63852848839538	-28.539432189230003	0.0350735389417735	0.0883973385489027	0.8864771282113779
Classel	Service7	-4.782189628921195	-186.50051422415	0.5603187381134638	0.2515533185748886	0.35239637489572865
- Statistiques:**
  - Fiabilité: 0.0
  - Fitness: 1.5378106832688986
  - Fitness finale: 1.5378106832688986
  - Temps d'exécution: [Redacted]

Figure III.3 : Lancement des compositions de service avec leur QoS

Afin de comparer la solution quasi optimale atteinte par l'exécution de la méthode Fruit Fly avec la meilleure solution existante dans la base générée, Nous calculons la composition Optimale de note base.

The screenshot shows a window titled "Composition optimale" with a central box labeled "Composition Optimale". Below this is a table with 7 columns: "id class", "N° service", "Cout", "TempsRep", "Fiabilité", "Disponibilité", and "Reputation". The table contains 7 rows of data. At the bottom of the window, there is a label "Fitness Opt. :" followed by a red-bordered box containing the value "0.78786705132069".

id class	N° service	Cout	TempsRep	Fiabilité	Disponibilité	Reputation
Classe1	Service38	-7.074671403974474	-18.531287619438565	0.5382970868777429	0.1530599329859603	0.9308892952756019
Classe2	Service21	-9.261125668964194	-40.09789951515389	0.5460549943945725	0.14129096563394422	0.9183063400139015
Classe3	Service21	-6.114906540225373	-8.712861702897657	0.5595490775352708	0.24087335089420994	0.6126897662945721
Classe4	Service105	-0.18735874292121202	-188.93642451221513	0.689472719904999	0.3356138448063763	0.6307588057910801
Classe5	Service69	-7.923436376205174	-11.566899242228912	0.6686566451608602	0.23145850040715704	0.8441247247510817
Classe6	Service95	-4.046561463283853	-45.925407719442376	0.6330604504836014	0.3533646348726958	0.5938052576305646
Classe7	Service49	-1.93869119365643	-40.35521977356233	0.5586664667647653	0.15898760334842027	0.6742741302010521

Fitness Opt. : 0.78786705132069

Figure III.4 : Composition optimale générée à partir de la base.

## 4.2 Lancement d'Algorithme et Affichage :

On va lancer notre algorithme Fruit Fly.

La meilleure composition de chaque itération est montrée dans l'interface suivant :

**Sélection des services composés basés sur les méta-heuristiques (FRUIT FLY)**

Fichier Edition Affichage

Données

N° Classes : 7  
N° Services C : 120  
N° QoS : 5

Base de données

Requête de l'utilisateur

Coût : -6  
Temp réponse : -103  
Fiabilité : 0.5  
Disponibilité : 0.5  
Réputation : 0.8

Paramètres

Nbre. de comp. : 40  
Nbre. d'itération : 100

N°	X <sup>1</sup>	Y <sup>1</sup>
1	14.58424106026	-1.66142952
2	0.134218199140	-5.44925772
3	0.91432003598	0.228117454
4	0.019830211894	0.703374079
5	-2.13948757615	-7.165708868
6	7.80916707689505	12.47525254
7	10.8787592187	-1.58801124
8	1.869934029073	9.991402366
9	-7.96688742109	-1.84674739

Lancer l'Algo. (FruitFly)

Fichier \*.xml Compositions Clust Graphe C-best Meilleur Solution

N° itération : **Itération N°17**

id class	N° service	Coût	TempRep	Fiabilité	Disponibilité	Réputation
Classe1	Service121	-21.83949558263946	-4.710108137825175	0.6149408056598897	0.310847661730452	0.5260338681633757
Classe2	Service106	-8.769227915990555	-196.07058890767277	0.012573273374250826	0.2659952320413624	0.19987140540162762
Classe3	Service88	-20.155525751784747	-53.61677554370669	0.07818588868378125	0.11382502503266622	0.7367491413318904
Classe4	Service104	-8.423597227022606	-176.52516782270047	0.11658298744543576	0.11482876351144683	0.548810824967465
Classe5	Service47	-1.7611333389679595	-63.877638109287275	0.03274069308494795	0.15307293901025093	0.7840938751123274
Classe6	Service18	-0.040818366401400574	-229.87757579256038	0.3394993681325454	0.15946227700386323	0.36112351320558367
Classe7	Service74	-26.814375128014714	-278.31370060996054	0.25131748908121654	0.2917926668130774	0.515590110119482

Penalité : 0.0

Fitness : 2.224508709866287

Fitness finale : 2.224508709866287

Best Comp. : 39

Résultat

Temps d'exécution : 9.9 sec

Figure III.2: meilleur composition pour chaque itération

La solution optimale est montrée dans l'interface suivant :

Sélection des services composés basés sur les critères heuristiques (FRUIT FLY)

Fichier Edition Affichage

Fichier \*cmd Composition Cher Graphe C-cher Meilleur Solution

Données

N° Classes : 7  
 N° Services C : 120  
 N° QoS : 5

Base de données

Requis de l'utilisateur

Coût : -6  
 Temps réponse : -100  
 Fiabilité : 0.5  
 Disponibilité : 0.5  
 Réputation : 0.8

Paramètres

Nbr. de comp. : 40  
 Nbr. d'itération : 100

N°	N°	Y1
1	14.5642459626	-5.86142332
2	0.134818189346	-5.44809772
3	0.01412000398	0.23817454
4	0.01888211834	0.709374678
5	0.23348777615	7.18578888
6	7.80956707689302	12.47505254
7	10.87879928287	-1.58805124
8	1.869914029671	0.991448366
9	-7.86688741100	-1.96074770

Lancer l'Algo. (FruitFly)

ID class	N° service	Coût	TempsRep	Fiabilité	Disponibilité	Reputation
Classe1	Service1	-22.175827027653204	-183.893188514827	0.48286017070057813	0.30565259454213706	0.11941318529241318
Classe2	Service86	-24.66886543218049	-224.6594972084432	0.22626851689127447	0.2794450642386237	0.16885044675428435
Classe3	Service33	-13.456589309062415	-264.6078682800041	0.35292734726511527	0.010610874966552416	0.9934342350804329
Classe4	Service104	-23.05115497437269	-10.839607473538704	0.034723137795976314	0.3170891772353182	0.17894226459455786
Classe5	Service42	-29.077196647878203	-16.242982259012716	0.09648622599151492	0.04767673616482645	0.650429030510975
Classe6	Service55	-14.213517629566065	-294.2229324801944	0.28382690901438543	0.030067617894873784	0.1064482026256445
Classe7	Service114	-24.661007611484392	-138.51209504304765	0.3998796332584085	0.26576207105804458	0.05074491227275091

Coût : -151.30455883238100

Temps de réponse : -1152.3483015861235

Fiabilité : 0.48286017070057813

Disponibilité : 1.0948526782247712E-7

Réputation : 0.9934342350804329

Finex Opt. : 1.5432738930526746

Résultat

Temps d'exécution : 0.0 sec

Figure III.3: La solution finale des compositions de services

## 5 Expérimentations :

Nos expérimentations sont menées sur une machine dotée de :

<b>Critère QOS</b>	<b>Classe1</b>	<b>....</b>	<b>Classe n</b>
<b>Temps d'exécution</b>	<b>0-300(m.s)</b>	<b>....</b>	<b>0-300(m.s)</b>
<b>Réputation</b>	<b>0-5</b>	<b>....</b>	<b>0-5</b>
<b>Coût</b>	<b>0-30(\$)</b>	<b>....</b>	<b>0-30(\$)</b>
<b>Fiabilité</b>	<b>0.5-1.0</b>	<b>....</b>	<b>0.5-1.0</b>
<b>Disponibilité</b>	<b>0.7-1.0</b>	<b>....</b>	<b>0.7-1.0</b>

**Tableau III.1: Les intervalles des paramètres de QoS de la base de sélection**

Nous présentons les résultats obtenus après plusieurs simulations, nous concentrons sur l'optimalité et le temps d'exécution.

Les paramètres changeables selon les expérimentations sont :

Le Nombre d'itérations va de 50 à 100.

Le Nombre de compositions va de 10 à 40.

La taille de la base : classes (3 à 10), services (50 à 200).

Dans notre teste nous calculons le taux d'optimalité par rapport au nombre d'itérations, et le temps d'exécution en parallèle sur la base totale (10 classes, 200 services) et avec un nombre de composition de 40 et nombre d'itération de 100, les résultats sont présenté dans le graphe suivant :





Figure III.4: Courbe d'optimalité / itération des compositions de service

## 6 BILAN

Dans ce travail nous avons proposé une adaptation WS-FOA pour la sélection de services web par l'application de fruit Fly.

Cet algorithme permet la sélection dynamique de services web en utilisant notre modèle cité, dans le chapitre précédent, pour un ensemble de paramètres de qualité de service (QoS). Ces paramètres sont classés : (1) Les paramètres dynamiques qui peuvent changer dans le temps. Par exemple, le temps de réponse et le nombre actuel d'invocations concurrentes que le service web est en train d'exécuter ; (2) Les paramètres "statiques" qui ne changent pas ou qui changent avec peu de fréquence, par exemple, le coût par invocation.

Les paramètres de nombre de composition et nombre d'itération ont été donnée pour choisir la meilleure composition avec des classes, services et les résultats calculés pour les critères de qualité de service.

## 7 Conclusion :

Dans ce chapitre, nous avons présenté l'implémentation de l'algorithme Fruit Fly et la réalisation de notre application, on a montré comment obtenir le meilleur résultat à l'aide de notre méthode ensuite, on a présenté des tests pour expérimenter des courbes d'optimalité avec le nombre d'itérations et le nombre de compositions de service.



**Conclusion**

**Général**

## Conclusion général

Nous avons présenté dans ce mémoire les technologies liées aux services Web. Nous avons proposé aussi un algorithme de sélection qui se base sur l'algorithme Fruit Fly pour l'optimisation de la sélection des services web dans une composition.

Pour cela, nous avons commencé par présenter notre problématique qui a été détaillée d'un chapitre à l'autre jusqu'à l'obtention d'une solution, dans le premier chapitre nous avons présenté les services web et la sélection des qualités de service, et en deuxième chapitre, nous avons présenté un état de l'art sur les méta-heuristiques qui sont utilisées dans la résolution de notre problème avec un algorithme, et nous avons détaillé la méta-heuristique « Fruit Fly » pour résoudre notre problème afin d'atteindre les meilleurs résultats possibles.

Nous avons terminé notre travail avec l'implémentation de notre prototype et quelques expérimentations.

Nous avons discuté les résultats afin de montrer l'efficacité de notre méthode dans la résolution de ce genre de problème, en matière d'optimalité et de temps.

Les résultats obtenus montrent que l'algorithme proposé arrive à trouver des solutions de bonne qualité.

Plusieurs améliorations peuvent être ajoutées à notre algorithme, nous prévoyons, dans un future proche, de prendre en considération plusieurs autres paramètres de qualité. D'adapter l'algorithme proposé dans un environnement Cloud Computing et enfin de comparer notre algorithme avec d'autres méta heuristiques comme PSO ou encore Fire Fly.

## Références Bibliographiques:

- [1] M. Papazoglou and D. Georgakopoulos, Service-oriented computing, Communications of the ACM, vol. 46, no. 10, pp. 25–28, 2003.
- [2] J. F. Chen, H. H. Wang, D. Towey, C. Y. Mao, R. B. Huang, and Y. Z. Zhan, Worst-input mutation approach to web services vulnerability testing based on SOAP messages, Tsinghua Science and Technology, vol. 19, no. 5, pp. 429–441, 2014.
- [3] T. Wen, G. J. Sheng, Q. Guo, and Y. Q. Li, Web service composition based on modified particle swarm optimization, (in Chinese), Chinese Journal of Computers, vol. 36, no. 5, pp. 1031–1045, 2013.
- [4] Li, Q., Liu, A., Liu, H., Lin, B., Huang, L., and Gu, N. Web services provision: solutions, challenges and opportunities (invited paper). In Proceedings of the 3rd international Conference on Ubiquitous information Management and Communication, Suwon, Korea, January 15 - 16, 2009.
- [5] W3C World Wide Web Consortium; “Web Services Architecture”; W3C Working Group Note 11; February 2004; <http://www.w3.org/TR/ws-arch>
- [6] World Wide Web Consortium; “Extensible Markup Language (XML)”; <http://www.w3.org/XML/>
- [7] E. Cerami, Web Services Essentials, édition O’Reilly, Février 2002 , Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler. 6 October 2000 (See)
- [8] J-M. Chauvet, Services Web avec SOAP, WSDL, UDDI, ebXML..., Edition EYROLLES; SOAP Version 1.2 Part 2: Adjuncts, W3C Recommendation, M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Nielsen, 24 June 2003 (See)
- [9] Alonso G., Casati F., Kuno H. et Machiraju V.; “Web Services Concepts, Architectures and Applications”; Springer-Verlag Berlin; Mars 2003.
- [10] Shuping Ran; “A model for web services discovery with QdS”; ACM SIGecom Exchanges; Volume 4 , Issue 1 Spring, 2003; ACM Press;
- [11] E. Alrifai, T. Risse Selecting Skyline Services for QoS-based Web Service Composition In Proceedings of the WWW 2010, April 26-30, 2010, Raleigh, North Carolina, USA.

- [12] Q. Yu, A. Bouguettaya, Foundations for Efficient Web Service Selection Springer Science+Business Media, 2010.
- [13] M. M. Akbar, E. G. Manning, G. C. Shoja, and S. Khan. Heuristic solutions for the multiple-choice multi-dimension knapsack problem. In Proceedings of the International Conference on Computational Science-Part II, pages 659-668, London, UK, 2001. Springer-Verlag.
- [14] D. Ardagna and B. Pernici. Global and local QoS constraints guarantee in web service selection. In Proceedings of the IEEE International Conference on Web Services. Pages 805-806, Washington, DC, USA, 2005. IEEE Computer Society.
- [15] D. Ardagna and B. Pernici Adaptive service composition in flexible processes. IEEE Transactions on Software Engineering, 33(6): 369-384, 2007 Dustdar, S and Schreiner, W. "A survey on web services composition", Int. J. Web and Grid Services, Vol. 1, No. 1, pp.1-30. (2005). [11a] F. Li, F. Yang, K. Shuang and S. Su. Q-peer: A decentralized QoS registry architecture for web services. In Proceedings of the International Conference on Services Computing, pages 145-156, 2007.
- [16] W. T. Pan, A new fruit fly optimization algorithm: Taking the financial distress model as an example, Knowledge Based Systems, vol. 26, no. 2, pp. 69–74, 2012.
- [17] S. Xiang, B. Zhao, A. Yang, and T. Wei, Dynamic measurement protocol in infrastructure as a service, Tsinghua Science and Technology, vol. 19, no. 5, pp. 470–477, 2014.
- [18] [X. Q. Fan, C. J. Jiang, J. L. Wang, and S. C. Pang, Random QoS aware reliable web service composition, (in Chinese), Journal of Software, vol. 20, no. 3, pp. 546–556, 2009.
- [19] E. Alrifai, T. Risse Combining Global Optimization with local selection for Efficient QoS-aware Service Composition in WWW09, April 20-24, 2009, Madrid, Spain.
- [20] H. Al-Helal and R. Gamble, Introducing replaceability into web service composition, IEEE Transaction on Services Computing, vol. 7, no. 2, pp. 198–209, 2014.
- [21]: Service oriented architecture: A field guide to integrating XML and web services, Upper Saddle River, NJ: Prentice Hall.
- [22]: M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Frystyk Nielsen, Editors. World Wide Web Consortium, 24 June 2003, revised 27 April 2007. This version of the SOAP Version 1.2 Part 1.
- [23]: "A Model for Web Services Discovery With QoS", ACM SIGecom Exchanges, 4(1), March 2003

[24] : La sélection des services web dans une composition à base de critères non fonctionnels. Thèse de doctorat Tlemcen.

[25]: Web Services: Concepts, Architectures and Applications, Springer.

## Liste de Figures

Figure I.1: Structure d'un message SOAP [22] .....	8
Figure I 2 : La spécification d'un service Web avec WSDL [Alonso G, Casati F2003 .....	9
Figure I.3 : Un nouveau modèle de registre et découverte de services web [23].....	11
Figure I 4 : Les différentes relations entre les spécifications des services web [Erl, 2004]	12
Figure I.5 : Exemple de composition.....	13
Figure I.6: Les approches de Sélection.....	17
Figure I.7: Approches de résolution de problème de sélection de SW .....	18
Figure II.1: Quatre modèles pour la composition de service.....	26
Figure II 2: les étapes de l'ensemble de services web par fruit Fly .....	33
Figure III.1:Interface principale .....	37
Figure III.2: Chargement d'un fichier XML .....	38
Figure III.5:meilleur composition pour chaque itération .....	41
Figure III.6:La solution finale des compositions de services .....	42
Figure III.7:Courbe d'optimalité / itération des compositions de service .....	44
Tableau I.1: Les domaines de valeurs des critères	14
Tableau III.1:Les intervalles des paramètres de QOS de la base de sélection	43

## Liste des Acronyms

### A

**API: Application Programming Interface**

### B

**B2B: Business To Business**

### C

**CORBA: Common Object Request Broker Architecture.**

### D

**DCOM: Distributed Component Object Model**

### E

**EAI : Entreprise Application Intégration.**

### F

**FOA: Fruit Fly Optimisation Algorithm**

### H

**HTML: Hyper Text Markup Language.**

**HTTP : Hyper Text Transport Protocol.**

### J

**JDK : Java Developpement Kit.**

### Q

**QoS: Quality of Service.**

**QoSWSC: Quality of Service Web Service Composite**

### S

**SOA: service oriented architecture**

**SOAP: Simple Object Access Protocol.**

### U

**UDDI: Universal Description Discovery and Integration.**

### W

**WS-FOA: Web Service composition based on Fruit Fly Optimization Algorithm.**

**WWW: World Wide Web.**

**WSDL: Web Service Description language.**

**W3C: World Wide Web Consortium.**

### X

**XML : extensible Markup Language.**



## Résumé

Entre un énorme nombre de services web offerts sur Internet qui ont la même fonctionnalité, et le fait qu'un seul service web ne peut jamais satisfaire les exigences de l'utilisateur, il est donc nécessaire de choisir une composition de services qui répond au mieux aux besoins exigés. Afin d'atteindre cet objectif des méthodes de sélection sont développées et améliorées pour faciliter le travail de l'utilisateur et minimiser le temps de sa recherche.

Ces méthodes avec leurs différentes stratégies souvent utilisent le critère de qualité qui distingue chaque service pour la sélection, cette qualité de service (QoS) nous aide à déterminer les meilleurs services parmi d'autres.

Nous nous intéressons, dans notre travail à la sélection des services web basée sur la qualité de service d'un côté, et d'un autre côté de décrire le modèle de notre approche pour sélectionner le meilleur fournisseur d'un service web qui est le cas de notre problème de sélection connu comme un problème de Fruit Fly.

La méthode de résolution proposée est une Méta-heuristique récente nommée WS-FOA, nous adapterons cette méthode sur notre problème avec les améliorations nécessaires pour atteindre la solution optimal.

**Mots clés :** Sélection des services web, qualité de service, méta-heuristique, Fruit Fly, WS-FOA : Web Service composition based on Fruit Fly Optimization Algorithm.

## Abstract:

a huge number of web services offered on the Internet that have the same functionality, and the fact that a single web service can never satisfy the user's requirements, it is therefore necessary to choose a service composition that best meets the user's needs in order to achieve this objective selection methods are developed and improved to facilitate the user's work and minimize the time of his research.

These methods with their different strategies often use the quality criterion that distinguishes each service for the selection, this quality of service (QoS) helps us to determine the best services among others.

In our work we are interested in the selection of web services based on quality of service on the one hand and, on the other hand to describe the model of our approach to selecting the best provider of a web service which is the case of our selection problem known as a Fruit Fly problem.

The proposed resolution method is a recent Meta-heuristic named WS-FOA; we will adapt this method to our problem with the necessary improvements to reach the optimal solution.

**Keywords:** Selection of web services, quality of service, meta-heuristics, Fruit Fly, WS-FOA: Web Service composition based on Fruit Fly Optimization Algorithm,

إن عدد كبير من خدمات الويب المعروضة على شبكة الإنترنت التي لها نفس الوظيفة، ونظراً أن خدمة ويب واحدة لا يمكنها تلبية متطلبات بعض الأحيان، ولذلك فمن الضروري اختيار خدمة الويب المركبة التي تلبى الاحتياجات على أتم وجه، من أجل تحقيق هذا الاختيار تم تطوير الطرق وتحسينها لتسهيل عمل المستخدم وتقليل الوقت من أبحاثه.

هذه الأساليب مع استراتيجياتها المختلفة وغالباً ما تستخدم معيار الجودة التي تميز كل خدمة مما يساعدنا على تحديد أفضل الخدمات وغيرها.

في عملنا هذا أعرنا الاهتمام على اختيار خدمات الويب استناداً على النوعية من الجانب الخدماتي وعلى طرق حل المشاكل بطرق مثلى من ناحية أخرى، هذا هو الحال بالنسبة للعمل الذي قمنا به في إطار معالجة

Fruit Fly

Méta-heuristique الطريقة المقترحة التي تم اكتشافها مؤخراً البحث عن

WS-FOA إعطاء حل أمثل بالتحسينات اللازمة.

كلمات البحث (خدمات الويب، خدمة الويب المركبة، معيار الجودة، النوعية من الجانب الخدماتي)