République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Larbi Tébessi-Tébessa-
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département : Mathématiques et Informatique

MEMOIRE DE MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Système d'information

Thème

Composition des services web basée QoS en présence des préférences conditionnelles

***Présenté par* :**

- Mr.Saiad riad
- Mr.Adel houche

***Devant le jury*:**

- M.Haouam, MCB, Université Laarbi Tébessi : Président
- S.Bourougaa,MCB, Université Laarbi Tébessi : Examinateur
- Ali-Abdelatif Betouil, MAB, Université Laarbi Tébessi : Encadreur

Date de soutenance : 25/05/2017

Note : 15/20 Mention : bien

Remerciement

Je tiens à remercier l'encadreur Dr **Ali-Abdelatif BETOUIL** pour l'encadrement de nous mémoire et pour ses encouragements et ses aides, ainsi que ses soutiens et ses patiences tout au long du préparation mémoire master. Elle a été toujours disponible pour répondre à mes questions .

Quel' que soit dans le bureau ou par téléphone ou par facebook

Je tiens également à remercier les membres de jury qui m'ont fait l'honneur de bien vouloir évaluer nous mémoire master.

Je tiens à remercier mes parents, ma famille et mes amis surtout **Dr.Adel tolba** Mr.**sekrani yaaqoub** est Dr.**rahab hichem est saiad chahinez** pour nous aident et pour leurs encouragements et soutiens pendant toute la période de la formation master.

ملخص

في الوقت الحاضر أصبحت خدمات الواب كثيرة الاستخدام خاصة من قبل الشركات والمؤسسات وهذا من أجل عرض أعمالهم وخدماتهم على الشبكة العنكبوتية، و تعتبر عملية تركيب خدمات الويب من أكثر المواضيع البحث في الوقت الحالي وهذا لأنها تُمكِن من معالجة المشاكل المعقدة اعتمادا على خدمات واب بسيطة حيث الذي يهدف إلى المقارنة بين عدة بدائل في وجود عدة معايير , متخذ القرار يقوم بتقييم الخيارات و فقا لهذه المعايير (وجهات النظر) و التفضيلات لكل معيار , قد تكون هذه التفضيلات متناقضة الأمر الذي يتطلب إيجاد حلول وسط بين هذه المعايير .

كما أن اكبر تركيز منصب الآن حول توفير حل أمثل لمشكلة القرار متعدد المعايير لذا اعتمدنا في بحثنا حول كيفية فهم اتخاذ القرار فيما يخص خيارات كثيرة جدا ,كل خيار له عدة معايير فتوصلنا إلى ما مدى مساعدتنا لمتخذ القرار بناءا على خياراته و مدخلاته على القيم.

ولفهم هذه النمطية الخاصة باختيارات المستعمل ركزنا على ثلاث خدمات ويب وهي الخطوط الجوية و كراء السيارة و كراء النزل وكل خدمة استعملنا على سبيل الحصر خمسة معايير و هي السعر . المدة . و سمعة الوكالات و الدقة و الاستمرارية.

و كمرحلة أولى ركزنا في هذه المذكرة دراسة خاصية تشيبي شاف و هي خاصية تجميع متعدد المعايير غير انه لاحظنا أنها تعتمد فقط حول القيم المدخلة للمستعمل لتعطينا خيارا أو عدة خيارات متوازنة القيم كنتيجة لتطبيقها ولا تأخذ بعين الاعتبار ترتيب أولوية الخيارات التي بفضلها المستعمل لأن لكل شخص و أرائه و توجهاته الاقتصادية .

لكن المرحلة الثانية اعتمدنا خوارزمية ترتكز حول شيئين اثنين هما مدخلات القيم و ترتيب الاولويات ثم يتم الانتقاء حسب الترتيب الأبجدي للقيم

Résumé

Actuellement, les services web sont devenus très utilisées , essentiellement par les entreprises. Et cela pour exposer leurs services sur le web. La composition de services web devient parmi les thèmes de recherches les plus abordés, et cela est du à sont efficacité dans le traitement des problèmes complexes par des services web élémentaires, alors il vise à comparer différentes alternatives, en présence d'un certain nombre de paramètres. Le décideur, va évaluer les choix selon ses critères (points de vus) et les préférences de chaque critère. Ces choix, peuvent êtres contradictoires, chose qui nécessite des solutions compromis entre ces paramètres là.

La concentration est orientée actuellement sur la façon d'offrir une solution optimale au problème de la décision multi critères, c'est pour ça qu'on est basé dans notre rechercher sur la façon de comprendre la prise de décision en présence d'un très grand nombre de choix, et chaque choix aura plusieurs critères. Alors, on est abouti au point d'aider le décideur selon ses choix et ses valeurs entrés.
Pour comprendre ce modèle (standard) des choix de l'utilisateur, on est concentré sur trois services web, qui sont : les services aériennes, la location automobile, et la réservation dans les hôtels (location dans les hôtels). Et pour chaque service, et à titre d'exhaustivité, on a utilisé cinq critères : Le prix, la durée, la réputation de l'agence,la disponibilité,la fiabilité .

Comme première étape, on a appliquée dans ce mémoire la norme Tchebycheff, qui est un paramètre de rassemblement multi critères. Cependant, on a remarqué qu'elle considère seulement les valeurs saisies par l'utilisateur, pour nous donner, en tant que résultat, un ou plusieurs choix à valeurs équilibrés, sans prendre en considération l'ordre de priorité des choix de l'utilisateur, car chacun a ses propre opinions et choix économiques.

Dans la deuxième étape, on a adopté un algorithme qui se base sur deux principes : les valeurs d'entrée, et l'ordre de préférence. Et le choix, se fera selon l'ordre lexicographique des vecteurs.

Mots-clés : aide à la décision multicritère, optimisation combinatoire, modélisation des préférences, services web, qualité de service (QoS), Tchebycheff

Abstract

In the present time the web's services became verry useful especially by the companies and the foundations and this is to expose their works and servicies on the net , and the operation of the make-up of the web's servicies is considred as the most researched topics in the present time and that because it has the ability of handling the complicated problems relying on simple web servicies where it aims to the comparison between many replacements with the existace of many standards , the adopter of the decision evaluates the options according to these standards (points of view) and preferences to each standard , these preferences may be contradicted which requires finding medial solutions among these standards

Besides that the biggest focus is on providing the best solution to the problem of the multi standards decision so we rely in our research on understanding the way of taking the decision in what it belongs to many choices , and each choise has alot of standards so we came to an end of how extent is our help to the adopter of the decision according to his choices and his accesses on the values .

And to understand this special type with the choices of the user we focused on three web servicies and they are : the atmospherical lines and the rental of cars and the rental of hostels and each service we used on the way of inclosing five standards which are the price .the duration . the reputation of agencies and the accuracy and the continuity .

And as a first step we focused on this memorandum the study of the speciality of Tchebycheff which is the speciality of the assemblage of multi standards however we noticed that it depends only on the values entered to the user in order to give us an option or many options with balanced values as a result to applicate it without taking in considration the arrangement of the priority of options that the user prefers because each person has his marbles and his economical orientation

But the second step we reyied adopted a grounded that focuses on two things which are: the access of the values and the arrangement of priorities then the selectness is done according to the alphabetical arrangement of the values

Keywords: multicriteria decision support, combinatorial optimization, preference modeling,web services, quality of service (QoS) , Tchebycheff .

Table des matières

Liste des tableaux

Liste des figures

Introduction

Au cours des dernières années, l'industrie du logiciel a été dominée par les services Web.

La plupart des entreprises publient leurs applications sur le World Wide Web à l'aide de services Web qui sont des applications accessibles sur internet réalisant chacune une tâche spécifique.

Les services web fournissent une nouvelle manière de développer des applications conformes aux besoins de l'internet en vue de rendre le web plus dynamique, pour fournir une solution à une tache complexe, on peut regrouper des services web pour n'en former qu'un seul.

Dans notre mémoire on va aborder les problèmes clés suivants:

1-qualité de service (QoS).

En présence de plusieurs services Web avec chevauchement ou une fonctionnalité identique, les utilisateurs discriminent de manière indiscutable les offres de services Web en fonction de leur QoS.

La qualité de service (QoS) est un concept large qui englobe un certain nombre de propriétés non fonctionnelles telles que **Prix**, **durée, disponibilité**, **fiabilité** et **réputation** .

Prix : le prix est le tarif qu'un demandeur de service doit payer pour invoquer l'opération. Fournisseurs de services Web soit publient directement le prix d'exécution de leurs opérations, ou fournissent des moyens de potentiel pour les demandeurs à s'en renseigner.

Durée : mesure le délai attendu en secondes entre le moment où une demande est envoyée et le moment où les résultats sont reçus.

En pratique la durée d'exécution est la somme de temps de traitement et le temps de transmission, le temps de traitement est connu à l' avance mais le temps de transmission est estimé.

Fiabilité : La fiabilité d'un service est la probabilité qu'une demande soit correcte a répondu dans le délai maximal prévu dans la description du service Web.

La fiabilité est une mesure liée à la configuration matérielle et / ou logicielle des services Web et les connexions réseau entre les demandeurs de service et les fournisseurs.

Disponibilité : La disponibilité d'un service *s* est la probabilité que le service soit accessible, la valeur de la disponibilité d'un service est le temps total (en secondes) dans

lequel le service est disponible pendant les dernières θ secondes (θ est une constante établie par un administrateur de la communauté des services).

Les services envoient des notifications au système concernant leurs états en cours d'exécution (c'est-à-dire disponibles, indisponibles).

Réputation : La réputation d'un service est une mesure de sa fiabilité. Elle dépend principalement de l'expérience de l'utilisateur final de service.

Différents utilisateurs finaux peuvent avoir différents opinions sur le même service. La valeur de la réputation est définie comme le classement moyen donné au service par les utilisateurs finaux.

Ces cinq propriétés s'appliquent à la fois à des services Web autonomes et des services Web composés.

2- La composition de services web :

La composition de services web est un processus qui permet d'assembler plusieurs services afin de créer une fonctionnalité, ayant une valeur ajoutée.

Plusieurs avantages peuvent être tries en adoptant la composition :

- Minimiser le cout et le temps de développement.
- Assurer une bonne adaptation aux changements d'environnement, et des besoins des clients (grâce aux compositions dynamiques).
- Les services composites peuvent être utilisés comme des services de base dans d'autres compositions.

Notre travail dans mémoire master vise à faire avancer l'état de l'art actuel dans les technologies de la composition du service Web.

Dans ce mémoire, un algorithme de classement du service Web basé sur les préférences des utilisateurs est proposé pour classer les services Web en fonction des préférences des utilisateurs et de l'aspect QoS du service Web.

Lorsque la demande de l'utilisateur ne peut pas être remplie par un seul service atomique, plusieurs services existants devraient être composés et livrés en tant que composition. Le cadre proposé permet à l'utilisateur de spécifier les contraintes locales et globales pour les services web composites qui améliorent la flexibilité.

Notre algorithme de recherche identifie les services les mieux adaptés pour chaque tâche dans la demande de l'utilisateur et, en choisissant le nombre de services candidats pour chaque tâche, réduit le temps de génération des plans de composition. Pour aborder le problème de la composition du service Web, l'algorithme de composition automatique des services Web QoS proposé dans notre projet fin d'étude est basé sur des opérations sur les aspects QoS des services Web et des préférences des utilisateurs.

Chapitre 1

Décision Multicritère et

Optimisation combinatoire

Ce premier chapitre est composé de deux parties ; la décision multicritère et l'optimisation combinatoire. Dans la première partie, on va définir les notions de base pour représenter les préférences d'un décideur, puis, on va introduire la notion de critère et les difficultés rencontrées dans le cas de décision multicritère avec prise en compte des différentes métriques (frontière de Pareto, de Lorenz, norme de Tchebycheff). Dans la deuxième partie, on va étudier Qu'est ce qu'un problème d'optimisation ? Nombreuses applications peuvent être vues comme des problèmes d'optimisation. L'optimisation concerne par exemple la minimisation du risque, du temps, du cout ou la maximisation de la qualité, du gain, ou de l'efficacité

1.1-Décision Multicritère

- Décision Multicritère l'action de décider après délibération. Petit Larousse

Dans un problème de décision, on appelle :

-décideur : l'entité (être humain ou artificiel) qui prend la décision.

-espaces *: des alternatives* : toutes les décisions possibles qu'un décideur peut prendre.

-alternative : une décision particulière.

-choix : parmi les différentes alternatives, le décideur doit faire un choix.

-conséquences *:* chaque alternative implique des conséquences. (Dubus, 2010).

Relation de préférence et fonction d'utilité

Pour qu'un décideur puisse prendre une bonne décision ou un bon choix, il doit être capable de comparer les alternatives pour choisie une (des) alternative (s) préférée (s).

a/ Relation de préférence

Une relation de préférence est définie par Queiroz (Queiroz, 2008) comme suit :

Soit A l'ensemble des alternatives disponibles. Les préférences du décideur sur les éléments de peuvent être décrites par une relation binaire sur A, c'est-à-dire un sous ensemble du produit cartésien $A \times A$. Cette relation est nommée une relation de préférences, et notée par $\succsim$. Ainsi pour un ensemble d'alternatives A et des éléments a^1 et $a^2 \in$, $a^1 \succsim a^2$ signifie que le décideur préfère a^1 à a^2 ou bien qu'il est indifférent entre a^1 et a^2.

b/ Fonction d'utilité

Dans un problème décisionnel, pour comparer les alternatives on attribue à chaque alternative une valeur réelle qui exprime le point de vue du décideur, on appelle cette valeur **« utilité »**. D'après Dubus (Dubus, 2010) une fonction d'utilité est définie comme suit:

Définition 1.1 (fonction d'utilité): soit $\succsim$ une relation de préférences sur un ensemble d'alternatives A fini, telle que $\succsim$ un péordre complet alors il existe une fonction u: $A \mapsto$R telle que:

$$\forall a^{1,2} \in A, u(a^1) \geq u(a^2) \Leftrightarrow a^1 \succsim a^2$$

On dit que u est une fonction d'utilité de la relation de préférence $\succsim$.

La décision multicritère est le domaine scientifique qui étudie des problèmes décisionnels en présence de plusieurs points de vue, chaque point de vue possède sa propre relation de préférences. On appelle *critère* une relation de préférence qui correspond à un point de vue. Dans ce genre des problèmes décisionnels, au lieu d'associer à un décideur une seule relation de préférence $\succsim$ on lui associe m relations $\succsim^1, \ldots, \succsim^m$ où $\succsim^i$ correspond à la relation de préférence associée au critère i.

Plusieurs problèmes existent en présence de plusieurs critères : Comment comparer deux alternatives ? Comment chercher une solution de compromis entre plusieurs critères qui peuvent être contradictoires ? (Dubus, 2010).

Exemple 1.1

Voici un exemple de service web est leurs critères, chaque service web est caractérisée par 5 critères qui sont : le prix, la durée, la réputation, la fiabilité et la disponibilité.

ID	Prix	Durée	Reput	Fiabilité	Disponibilité
Service web 1	**418**	**236**	**576,25**	**241**	**29,8**
Service web 2	**421**	**246**	**595,5**	**257**	**30,5**
Service web 3	**413**	**243**	**723**	**285**	**27,3**
Service web 4	**433**	**261**	**1160**	**279**	**33,2**
Service web 5	**402**	**256**	**553,93**	**278**	**34**
Service web 6	**422**	**259**	**684**	**285**	**28,6**
Service web 7	**450**	**239**	**708,4**	**278**	**19**

Tab 1.1 exemple de service web

1.2-Démarche générale en décision multicritère

- Définir l'ensemble des alternatives possibles (les objets sur lesquels portent la décision : candidats, ordonnancements, plans), et leurs attributs
- définir les différents points de vue sur lesquels on jugera les alternatives, établir des préférences sur ces points de vue, les traduire en critères (de manière ordinale ou cardinale)
- Synthétiser une structure de préférence globale sur les alternatives.
- Exploiter cette structure de préférence globale pour décider (algorithmes, calculs).

1.3. Comparaison d'alternatives en décision multicritère

Dans un problème de décision multicritère, au lieu d'associer à un décideur une seule relation de préférence $\succsim$ on lui associe *m* relations $\succsim^1$, …, $\succsim^m$ où $\succsim^i$ correspond à la relation de préférence Associée au critère *i* et on attribue pour chaque relation $\succsim$ une utilité span style On peut donc définir une fonction d'utilité multicritère comme suit :

1.3.1 Fonction d'utilité multicritère:

Soit *A* l'espace d'alternatives, (u^1,…,u^m) un ensemble de fonction d'utilité sur A . On appelle fonction d'utilité multicritère la fonction

U : A $\longrightarrow$ $\mathbf{R^m}$ telle que $\forall a \in A$ **u(a) = (u^1(a) ,…,u^m (a)).** (Dubus, 2010).

On associe à chaque alternative un vecteur d'utilité, chaque valeur dans ce vecteur correspond a l'utilité d'un critère. Alors, la comparaison d'alternatives en décision multicritère consiste à comparer les vecteurs.

Pour comparer les vecteurs entre eux, une première approche ; la dominance de Pareto ; issue de l'économie consiste à comparer les vecteurs deux à deux, critère par critère et éliminer les vecteurs dominés. Une alternative dominée n'est pas une alternative Pareto optimale, autrement elle est Pareto optimale. On appelle **«frontière de Pareto »** l'ensemble des alternatives non dominées au sens de Pareto.

Plusieurs chercheurs ont proposé des approches pour comparer les alternatives en décision multicritère ; parmi eux ; (Kostreva et Ogryczak, 1999) qui ont utilisé une relation de

dominance compatible avec la dominance de Pareto et consiste à appliquer la dominance de Pareto à une transformation de vecteurs, cette relation est : la dominance de Lorenz généralisée est définie dans (Dubus, 2010) comme suit :

1.3.2 Transformée de Lorenz:

Soit v ∈ R^m un vecteur , La transformée de Lorenz du vecteur *v* notée par L(v) est le vecteur des sommes cumulées du tri en ordre croissant de v .

on aura donc : $L(v) = (v^{(1)}, v^{(1)} + v^{(2)}, \dots \sum_{j=1}^{m} v^{(j)})$

Puisque la dominance de Lorenz est basée sur la dominance de Pareto, il existe donc des vecteurs incomparables entre eux. Dans ce cas, il faut obtenir un ensemble de solutions pouvant intéresser le décideur (optimales au sens de Lorenz).

1.3.2.1 Frontière de Lorenz:

On appelle « «frontière de Lorenz » l'ensemble des alternatives non dominées au sens de Lorenz.

1.4- Agrégateur et relations de dominances complètes

On est besoin de choisir une seule alternative, dans un ensemble de vecteurs non dominés qui sont incomparables entre eux, il faut utiliser des relations complètes, appelées « agrégateur ».

1.4.1 Agrégateur : Un agrégateur est une fonction qui associe à chaque vecteur de *m* valeurs réelle une seule valeur réelle est définie par (Dubus, 2010) comme suit :

Soit A l'espace d'alternatives, U :A → R^m une fonction d'utilité multicritère et U l'espace des critéres engendré par u. On appelle agrégateur de U toute fonction F :U →R . Alors la valeur *F(u(a))* est une solution agrégé de l'alternative a ∈ A

Dans la littérature, il existe beaucoup d'agrégateurs (Grabish et Perny, 2003), (Fodor et Roubens, 1994), parmi eux : la somme ordonnée pondérée (OWA), la norme de Tchebycheff et l'intégrale de Choquet. Dans notre travail, on a utilisé la norme de Tchebycheff.

1.4.2 - **La norme de Tchebycheff**

La norme de Tchebycheff est un agrégateur qui consiste d'abord à identifier deux vecteurs (le point idéal et le point Nadir), puis à comparer les vecteurs selon leurs distances par rapport à ces points.

Soit A l'espace d'alternatives , u :A →Rm une fonction d'utilité multicritére .

$Tcheb(v^1,\ldots,v^m)= \max_j \in \{1,\ldots,m\} (I^j - v^i)\backslash(I^j - N^j)$

Tcheb : est appelé norme de Tchebycheff

I : le pont idéal de la norme Tchebycheff

N : le pont nadir de la norme Tchebycheff

$(I^j - v^i)$ représente la distance à l'idéal, pour la mise à l'échelle pour cette distance en divise par $(I^j - N^j)$ pour chaque vecteur *v*, on associe une distance *Dis(v)* et on s'intéresse au critère maximisant cette distance ; La solution agrégée optimale est la solution qui minimise cette distance (Dubus, 2010). Le point idéal est un vecteur composé des meilleures valeurs qu'il est possible d'obtenir dans les différents critères, c'est-à-dire il domine au sens de Pareto toutes les solutions possibles et fourni une borne supérieure. Le point idéal est un vecteur qui n'existe pas nécessairement dans *A*. Le point Nadir est la borne inférieure, ce vecteur est composé des mauvaises (pires) valeurs des solutions spécialistes et ne pas les pires valeurs qu'il est possible d'obtenir dans chaque critère.

Exemple 1.2

ID	Prix	Durée	Reput	fiabilité	Disponibilité	Transformation	Distance max
Service web 1	**418**	**236**	**576,25**	**241**	**29,8**	0.66 ; 1 ; 0.96 ; 1 ; 0.28	1
Service web 2	**421**	**246**	**595,5**	**257**	**30,5**	0.60 ; 0.6 ; 0.93 ; 0.63 ; 0.23	0.93
Service web 3	**413**	**243**	**723**	**285**	**27,3**	0.77 ; 0.72;0.72;0;0.44	0.77
Service web 4	**433**	**261**	**1160**	**279**	**33,2**	0.35 ; 0;0;0.13;0.05	0.35
Service web 5	**402**	**256**	**553,93**	**278**	**34**	1 ; 0.2 ; 1 ; 0.15 ; 0	1
Service web 6	**422**	**259**	**684**	**285**	**28,6**	0.58 ; 0.08 ; 0.78 ;0 ;0.36	0.78
Service web 7	**450**	**239**	**708,4**	**278**	**19**	0 ; 0.88 ; 0.74 ; 0.15 ; 1	1

Tab 1.2 : Les vecteurs et la norme de Tchebycheff.

La solution agrégée optimale au sens de la norme de Tchebycheff est la solution qui minimise les distances maximales. Dans notre exemple, le Service web qui minimise les distances maximales est le service web 4

1.4.3 - **Algorithme de recherche (Thése-Betouil 2015)**

Une fois les plans d'exécution d'alternatifs sont générés, avec leurs vecteurs d'utilité respectifs, l'algorithme de recherche entre en action.

Les vecteurs d'utilité des services composites sont d'abord triés en fonction de préférences de l'utilisateur. Ainsi, le rang *i* d'un attribut de qualité dans un vecteur d'utilité doit correspondre à son rang dans les préférences utilisateur. On note $U_{\succsim}$ le vecteur ordonné du vecteur d'utilité *u*. Par exemple, si les préférences de l'utilisateur sont représentées comme suit :

Prix>fiabilité>durée

Et le vecteur d'utilité original est U=(prix = 0.88 , durée = 0.75 , fiabilité = 0.77) alors le vecteur ordonné de U est $U_{\succsim}$ =(0.88,0.77,0.75).

Une fois les vecteurs d'utilité sont obtenus , l'algorithme de recherche basé sur un péordre lexicographique fonctionne comme suit :

Dans la premiere itération ,les vecteurs d'utilité sont tries par ordre décroissant de l'utilité de l'attribut de qualité le plus préféré en fonction des préférences de l'utilisateur (ligne 3 , ou le résultat est dans la variable ordre – vect).

On a identifié deux situations : la premiere est celle ou on est confronté a plusieurs vecteurs avec la meme valeur pour l'utilité de lattribut le plus préféré de la qualité (L>=2). La deuxiéme situation est celle ou on a seulement un vecteur d'utilité avec la plus grande utilité pour l'attribut de qualité le plus préféré .

Dans le premier cas , on utilise un preordre lexicographique pour trier les meilleurs vecteurs d'utilité les mieux classés (ligne 5 et 6).

Le principe est comme suit :

On prnd la version ordonnée des L vecteurs les mieux classés , qui ont la meme utilité pour l'attribut le plus préféré , $\mathbf{(u^1_{\succsim},U^2_{\succsim},...,U^L_{\succsim})}$ .Pour simplifier la notation , on va omettre le

symbole $\gtrsim$ dans la version ordonnée des vecteurs d'utilité ($U^i_{\gtrsim} = U^i$) . puis , on applique la recherche lexicographique pour ces vecteurs d'utilité ordonnés $\mathbf{u^j=(U^j_1,U^j_2,\ldots,U^j_r)}$ et $\mathbf{U^k=(U^k1,u^k_2,\ldots,u^k_r)}$ ou r est le nombre d'attributs de qualité .

$\mathbf{(U_1^j, U_2^j, \ldots, U_r^j) > (U_1^k, U_2^k, \ldots, U_r^k)}$ $\mathbf{\exists m > 0, \forall j < m, (U_i^j = U_i^k) \wedge (U_m^j > U_m^k)}$.

Autrement dit , si une des utilités des m$^{\text{iéme}}$ attribut de qualité $U_m^j > U_m^k$ et tous les utilités précédents son égales .Plus simplement , on compare les premiéres utilités . Si elles sont égales, on compare les deuxiémes utilités et ainsi de suite . La relation entre les premiéres utilités correspondantes qui ne sont pas égales détermine la relation entre les vecteurs d'utilité entiers .Le vecteur de la plus haute valeur de l'utilité correspondante est celui qui est mieux classé .

Quand un préordre lexicographique est obtenu , on vérifie l'acceptabilité du vecteur d'utilité mieux classée dans ordre-vect (ligne 7). En effet , on dit que l'utilité d'un attribut de qualité est acceptable (ligne 8). Si elle est égale ou supérieure a une valeur de référence connue a l'avance .

Quand une utilité d'un attribut de qualité est inférieure a une valeur de référence , il représete la détérmination de la qualité de l'attribut , dans ce cas , tous les vecteurs qui ont été classés en respectant un préordre lexicographique son rejetés (ligne 10)et une autre recherche est appliquée sur les vecteurs restants (lignes 10 et 11).

Procedure pareto (U : set of ordered utility vectors , var v : pareto vector)

Begin

/*U= $(U_{\gtrsim}^1, U_{\gtrsim}^2, \ldots, U_{\gtrsim}^n)$

1- **J← 1 ; /* the current most preferred quality attribute**
2- **Max(j) ← $0<i\leq n U_i^j$;**
3- **Ordre – vect ← constrt(U,max(j)); /*ordr-vect corresponds to vectors with the value max(j) of the j-th quality attribute**
4- **L←cardinality (order-vector);**
5- **If L>=2 then {**

6- Order-vect ←order-lex(order-vect) }/*update the order vector with its lexicographic preorder version

7- v←Best-ranked(order-vect);

8- Check (v,Acceptable);

9- If acceptable=true then return v

10- Else U←Reject(U,order-vect)

11- Pareto (U,v) ; //recursive call

1.5-Optimisation Combinatoire

1.5.1-définition :

Langue française

Optimiser (v) : permettre d'obtenir le meilleur résultat possible par une action adaptée

Synonymes : améliorer, maximiser, mettre au point, optimaliser

Mathématiques :

La branche optimisation

Soit f : $\Omega \rightarrow R$, trouver $x^* \in \Omega$ tel que $f(x^*)=\min_{x \in \Omega} f(x)$

Ω : espace de recherche, espace des états, configurations, solution, est dans notre cas alternatifs

F : fonction objectif a minimiser /maximiser

Un problème d'optimisation combinatoire consiste à trouver la *meilleure* solution dans un ensemble discret dit ensemble des solutions réalisables. En général, cet ensemble est fini mais compte un très grand nombre d'éléments, et il est décrit de manière implicite, c'est-à-dire par une liste, relativement courte, de contraintes que doivent satisfaire les solutions réalisables.

Pour définir la notion de *meilleure solution*, une fonction, dite *fonction objectif*, est introduite. Pour chaque solution, elle renvoie un réel et la meilleure solution (ou *solution optimale*) est celle qui minimise ou maximise la fonction objectif. Clairement, un problème d'optimisation combinatoire peut avoir plusieurs solutions optimales.

Exemple 1.3

- Le problème du plus court chemin entre deux sommets *A* et *B* d'un graphe est un exemple classique de problème d'optimisation combinatoire. L'ensemble des solutions réalisables est l'ensemble des chemins entre *A* et *B* tandis que la fonction objectif est la longueur du chemin.
- Un autre problème d'optimisation combinatoire consiste à trouver le meilleur coup dans une position donnée au jeu d'échecs ou au jeu de dame

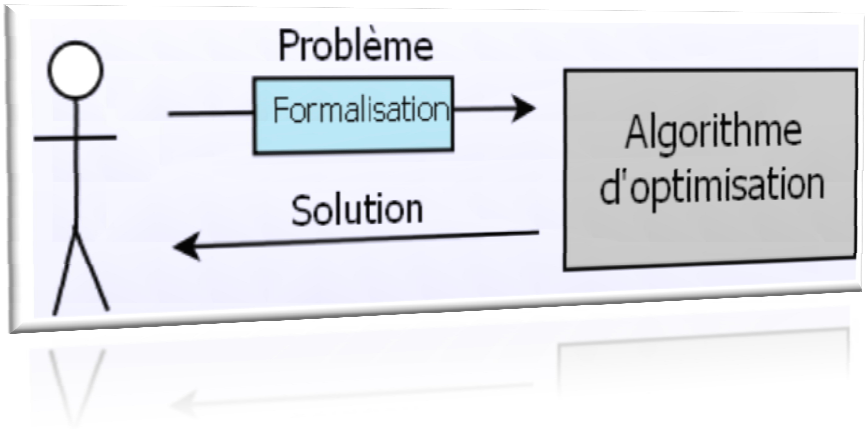


FIG1.1 Méthodes d'optimisation

1.5.2-Complexité, problèmes P et problèmes NP

Pour résoudre un problème d'optimisation combinatoire on utilise un *algorithme*. Un exemple d'algorithme qui marche pour n'importe quel problème d'optimisation combinatoire est le suivant : on essaye toutes les configurations possibles et on choisit celle qui minimise la fonction de coût. Néanmoins bien que cet algorithme soit très simple en théorie, il ne peut pas être utilisé dans la majorité des cas car le nombre de configurations, bien que fini, est très souvent gigantesque. Par exemple, pour le voyageur de commerce, il y a (N-1)!/2 configurations, ce qui fait 181440 pour N = 10 et 4.67 10^{155} pour N = 100. Cet exemple nous incite à définir la *complexité* d'un algorithme comme étant le nombre d'opérations à effectuer pour résoudre le problème. En fait on ne va s'intéresser qu'à la façon dont ce nombre augmente avec la taille des données (pour un graphe ce sera le nombre N de points). Ainsi pour le voyageur de commerce, les meilleurs algorithmes que l'on connaisse

voient leur complexité augmenter comme une exponentielle de N. En revanche pour l'appariement minimal le meilleur algorithme a une complexité en N^3. Et pour l'arbre couvrant minimal le meilleur algorithme est en $N^2\log(N)$.

La distinction fondamentale que l'on fait pour distinguer les différents problèmes d'optimisation combinatoire est le suivant : connaît-on un algorithme pour résoudre le problème dont la complexité soit au plus polynomiale en N ? Si c'est le cas le problème sera un *problème P* (P comme polynomial), sinon ce sera un *problème NP*. (Note pour les puristes : ce n'est pas vraiment la définition de **P** et de **NP** mais en gros ça revient à ça). Ainsi le voyageur de commerce est NP alors que l'appariement minimal et l'arbre couvrant minimal sont P.

1.6.- Quelques problèmes d'optimisation Combinatoire :

1.6.1- Le problème du voyageur de commerce

Le problème du voyageur de commerce consiste a un voyageur de commerce doit passer une fois et une seule dans un certain nombre de villes, il désire faire le moins de chemin possible. L'ensemble des configurations est l'ensemble des chemins fermés passant par tous les points et la fonction de coût est la longueur de ces chemins. Une configuration est donc caractérisée par la donnée de N liens.

1.6.2- Le problème d'appariement minimal

Le problème d'appariement minimal (Minimun Matching Problem (MMP)) il s'agit d'apparier les points deux à deux (il en faut donc un nombre pair) de façon que la longueur totale de l'appariement soit minimale.

L'ensemble des configurations est constitué d'appariements c'est-à-dire de configurations où chaque point est relié à un et un seul autre point (une configuration acceptable est donc composée de N/2 liens). La fonction de coût est la somme des longueurs des liens de l'appariement.

1.6.3- L'arbre couvrant minimal

Le problème de l'arbre couvrant minimal (Minimum Spanning Tree (MST)) consiste à trouver l'arbre passant par tous les points (d'où le nom) dont la somme des longueurs des liens est minimale.
L'ensemble des configurations est l'ensemble des arbres couvrants, c'est-à-dire l'ensemble des configurations constituées de N-1 liens et telles qu'il existe toujours un chemin entre deux points quelconques.

1.6.4- le problème d'affectation

Le problème d'affectation est un problème classique de recherche opérationnelle et d'optimisation combinatoire. Informellement ce problème consiste à attribuer au mieux des tâches à des agents. Chaque agent peut réaliser une unique tâche pour un coût donné et chaque tâche doit être réalisée par un unique agent. Les affectations (c'est-à-dire les couples agent-tâche) ont toutes un coût défini. Le but est de minimiser le coût total des affectations afin de réaliser toutes les tâches.

1.7-Graphe

1.7.1-Notion de graphe

Un graphe est composé d'un ensemble de sommets et d'un ensemble d'arcs; on note X l'ensemble des sommets, un arc est un couple de sommets, donc, un élément du produit cartésien XxX; on note U la famille des arcs.

Ainsi le graphe (noté, par exemple : G) sera-t-il égal au couple (X,U)
Le nombre de sommets est appelé l'ordre du graphe et sera, en général, noté : n
En général, les sommets d'un graphe seront représentés par des points de l'espace et les arcs par des flèches allant d'un point à un autre, c'est ce qu'on appelle la **représentation sagittale** du graphe : Dans un arc (x,y) : x est l'extrémité initiale et y l'extrémité finale. Un arc de la forme (x,x) est une boucle. On dit que y est **un successeur** de x s'il existe un arc (x,y); on dit aussi que x est un prédécesseur de y.(Graphe.fr)

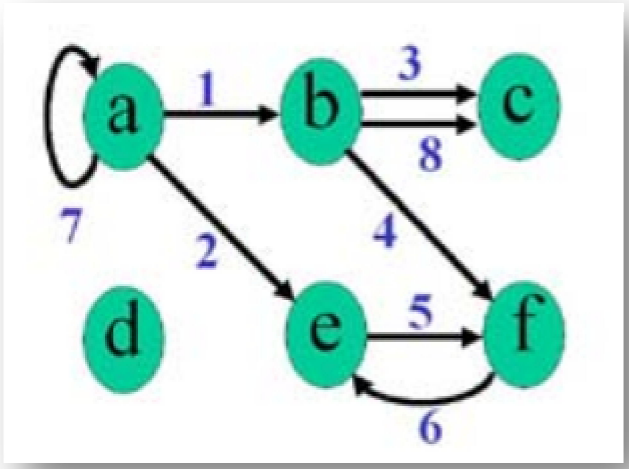


FiG1.2 la représentation sagittale d'un graphe d'ordre 6

1.7.2- Graphe simple

Un **graphe simple** est un graphe sans boucle tel qu'entre deux sommets différents, il y ait au plus un arc.

Dans un graphe simple, on pourra donc noter un arc à l'aide de ses extrémités initiale et finale.

Dans un graphe simple, on peut parler de l'ensemble des arcs.

Deux arcs sont dits adjacents s'ils ont une extrémité en commun.

Deux sommets sont dits adjacents si un arc les relie. (Graphe.fr)

1.7.3- graphe complet

Un graphe est dit complet si deux sommets différents quelconques sont reliés. On parle aussi de clique pour un graphe complet. Quand un graphe n'est pas complet, don ensemble de sommets peut être partitionné en cliques :

On notera **ω** le nombre maximal de sommets d'une clique sous-graphe. On notera **θ** le nombre minimal de cliques nécessaires pour partitionner l'ensemble des sommets du graphe. L'ordre d'un graphe est le nombre de sommets de ce graphe.(graph.fr)

Chapitre 2 :
Composition de services web

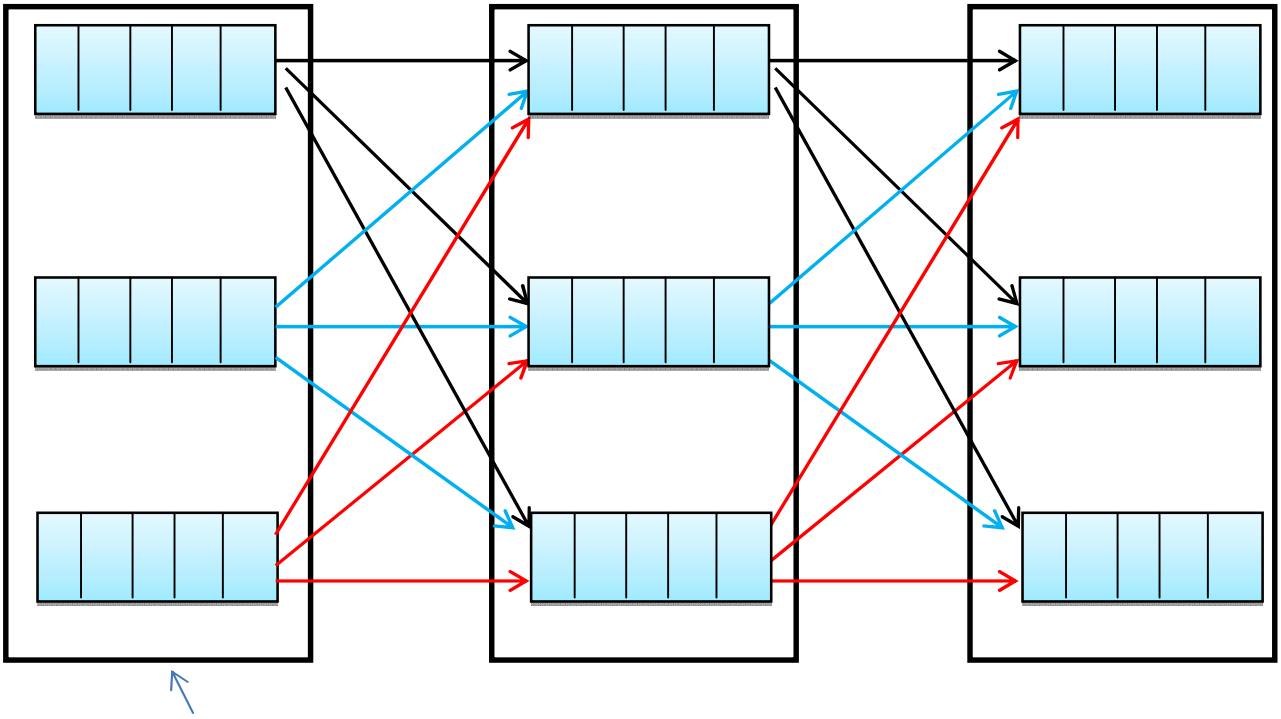
Dans ce chapitre on va présenter le problème que nous voulons de résoudre qui consiste un algorithme qui recherche une solution Pareto optimale pour un problème de décision multicritère en présente des contraintes sur les critères, cet algorithme trie les alternatives en fonction d'un ordre des préférences cet ordre représente l'importance des critères,
Pour cela en commence par la présentation de notre problème en suite en parlera de problème de composition de service web et après on va présenter les déférentes techniques de composition de service web. Enfin, on va définir les approches de service de composition (locale et globale).

2.1- introduction

Le problème que nous voulons de résoudre est que les alternatives sont composées de plusieurs sub-alternatives. Les sub-alternatives sont disponibles et regroupées dans différents ensembles qui sont triés.

La procédure de combinaison de sub-alternatives est nécessaire et pour calculer chaque alternative qui contient les mêmes critères que les sub-alternatives, en utilisant plusieurs opérations comme :la somme, produit, moyenne sur les valeur des critères des sub-alternatives

La figure suivante résumé le problème que nous voulons de résoudre



Ensemble des sub alter

FIG.2.1- Exemple du problème abordé (Betouil, 2015).

La majorité des travaux, utilisent des fonctions d'agrégations pour agréger toutes les valeurs en une seule, puis comparer les alternatives selon cette valeur agrégée. Les solutions obtenues en utilisant ces fonctions d'agrégation ne répondent pas toujours aux préférences du décideur

Dans la composition de services web, il existe plusieurs services qui ont les mêmes fonctionnalités mais différents dans leurs qualités. Il existe plusieurs plans d"exécution avec des paramètres QoS différents

2.2- Services Web

Il y a plusieurs définitions pour services web, on peut citer:

2.2.1 - Définition d'IBM

« Les services web sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services web) peuvent le découvrir et l'invoquer » (Ponge, 2004).

2.2.2- Définition de W3C

« Un service web est un système logiciel identifié par un URI dont les interfaces publiques et les incarnations sont définies et décrites en XML.Sa définition peut être découverte dynamiquement par d'autres systèmes logiciels. Ces derniers peuvent ensuite interagir avec le service web d'une façon décrite par sa définition, en utilisant des messages XML transportés par des protocoles Internet » (Ricardo, 2004).

2.3- Composition de services Web :

Avant de connaitre les techniques de compositions de services web : vous devez avoir la définition premièrement :

La composition de services web est le processus de créer de nouveaux services web à partir de deux ou plusieurs services web déjà présents et publiés sur le web. Un service web est dit composé ou composite lorsque son exécution implique des interactions avec d'autres services web afin de faire appel à leurs fonctionnalités. La composition de services web spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'exception. (Arenaza 2006).

2.3.1- Techniques de composition de services web

Il existe deux grandes familles de techniques de composition de services Web: techniques statiques et les techniques dynamiques.

2.3.1.1- Techniques statiques

Dans cette technique, les services web à composer sont choisis à l'heure de faire l'architecture et le design. Les composants sont choisis et reliés ensemble, avant d'être compilés et déployés (Dustdar et Schreiner,2005).Ceci peut marcher en autant que l'environnement des services web, les partenaires d'affaires, ainsi que les dits composants changent peu ou pas du tout. Microsoft Biztalk et Bea Weblogic sont deux exemples de moteurs de composition statique de services web (Sun et al., 2003). Si les fournisseurs de services proposent d'autres services ou changent les anciens services, des incohérences peuvent être causées, ce qui demanderait un changement de l'architecture du logiciel, voire de la définition du processus et créerait l'obligation de faire une nouvelle conception du système. Dans ce cas, la composition statique des services web est considérée trop restrictive: les composants doivent s'adapter automatiquement aux changements imprévisibles (Sun et al., 2003).

2.3.1.2- Techniques dynamiques

Dans cette approche, les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur (Osman et al., 2005). Cette approche offre le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur. Ce type de composition peut engendrer de nombreuses applications utiles, qui n'ont pas été prévues à l'étape de conception. Par conséquent, la composition dynamique de services Web est propice dans un environnement, tel que le Web et l'informatique pervasive où les composants disponibles sont dynamiques et les attentes des utilisateurs variables et personnalisées

2.3.2 Caractéristiques de Qualité de Service des Services Web

Dans le cadre des services Web, le W3C a identifié un ensemble de caractéristiques de QoS pertinentes pour le domaine des services (Baligand, 2008) :

- ✓ **Performance :** représente la vitesse avec laquelle un service Web répond à une requête. Cette qualité peut être mesurée en termes de débit, temps de réponse, temps d'exécution, temps de transaction, etc.
- ✓ **Confiance** : qualité d'un service à exécuter certaines fonctions sous certaines conditions et dans un intervalle de temps donné. Il s'agit d'une mesure de la capacité d'un service à maintenir sa qualité (généralement le nombre de coupures de service par jour, semaine, mois).

La confiance est également associée à la garantie de l'ordre et de la bonne livraison des messages.

✓ **Passage à l'échelle :** permet de quantifier le nombre de requêtes auxquelles le service peut faire face dans un intervalle de temps donné.

✓ **Capacité :** c'est le nombre de requêtes qu'il est possible de traiter simultanément.

✓ **Robustesse :** c'est la capacité à fonctionner dans un cas où les données en entrée provoquent des conflits ou sont incomplètes.

✓ **Traitement des exceptions :** représente un grand nombre de situations ne peuvent être prédites à l'avance ce qui implique de supporter les exceptions de façon adaptée.

✓ **Précision :** taux d'erreurs générées par le service.

✓ **Intégrité :** propriété garantissant que l'intégrité des données et des transactions est bien respectée, afin de ne pas aboutir à une situation inconsistante.

✓ **Accessibilité :** capacité à répondre à des requêtes.

✓ **Disponibilité :** le service devrait être prêt pour une consommation immédiate lors d'une invocation. La disponibilité est souvent associée au temps de réparation.

✓ **Interopérabilité :** capacité à pouvoir communiquer quels que soient la plateforme et les langages utilisés.

✓ **Sécurité :** il existe plusieurs traitements permettant de sécuriser les communications entre services Web (Authentification, Autorisation, Confidentialité, Cryptage de données).

2.4- Composition de services web basée QoS :

On suppose qu'il y a un utilisateur qui veut planifier un voyage, pour cela il a besoin de consommer 03 types de services, une réservation d'hôtel, une réservation de billet d'avion et une location de voiture, on note aussi qu'on doit sélectionner un seul service (ou entreprise) de chaque catégorie, en utilisant les critères de QoS (réputation, fiabilité, coûts, temps d'exécution…), en plus l'utilisateur exige des contraintes globales sur chaque critère de QOS, par exemple le coût total des 03 services, ne doit pas excéder une certaine limite. (Hadjila 2014).

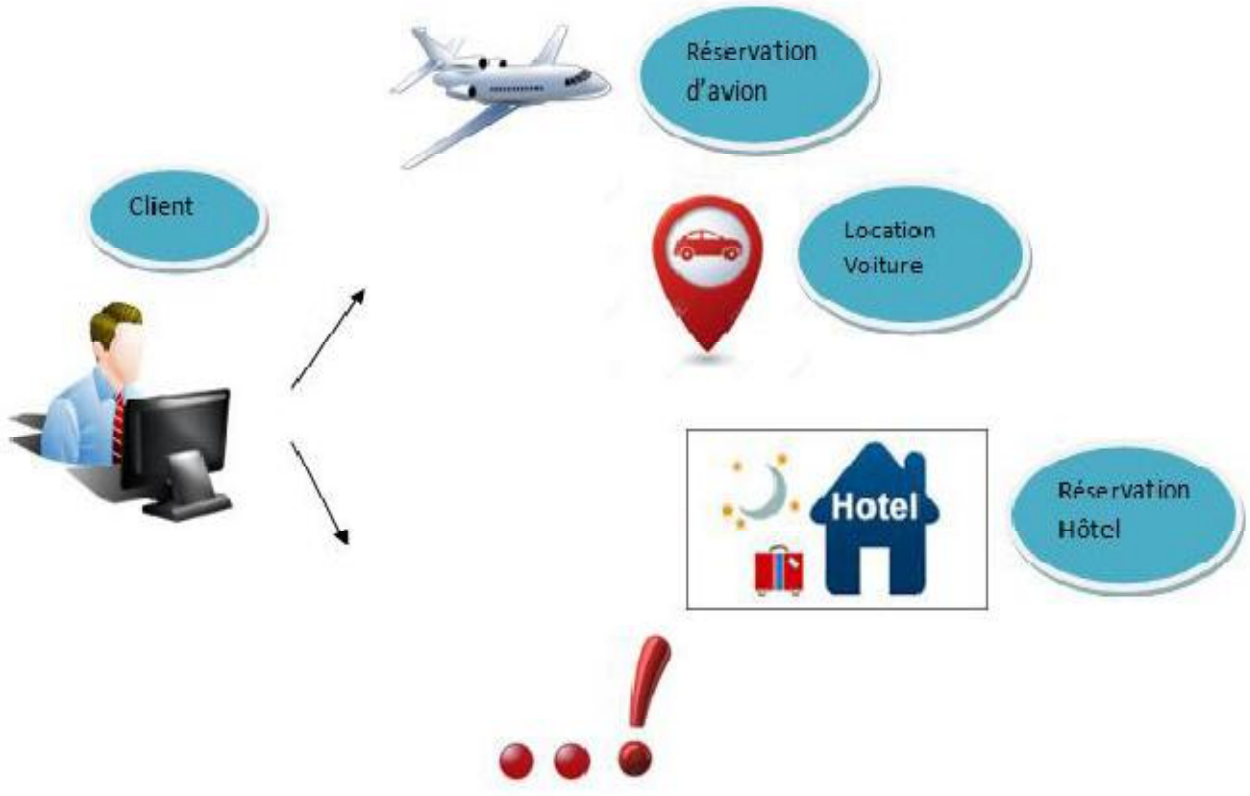


FIG 2.2 Exemple de composition de services (Liu et al., 2011)

Il est clair que ce problème est NP-Hard, en effet il est équivalent au MMKP (multi-dimensional multiple choice knapsack problem) [Pisinger, 1995].

On a plusieurs approches pour résoudre le problème de composition de services web, mais l'approche le plus utilise c'est « *workflow* ». Le workflow de composition est un graphe orienté où chaque noeud représente un service requis (classe de services) tel que pour chaque classe il existe plusieurs services web candidats qui fournissent le même service requis mais diffèrent dans leurs paramètres de qualité de services (QoS).

En effet, chaque service possède des paramètres de QoS (prix, durée, réputation, fiabilité, disponibilité,….) qui sont stockées chez le fournisseur de service.

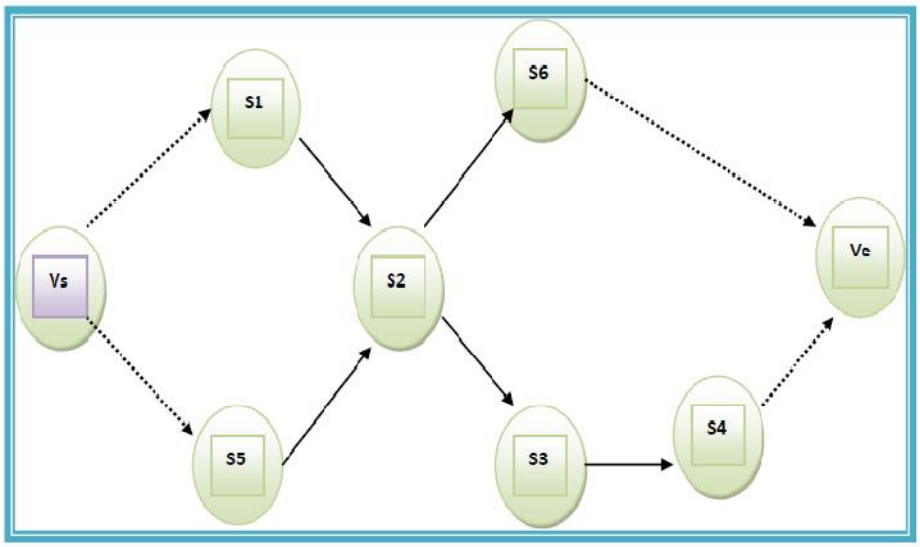


FIG 2.3 Exemple d'un Workflow (Belleili et Betouil, 2011)

Chemin d'exécution (Execution path) :

Un chemin d'exécution est une séquence de nœuds (tâches) $[S_1, S_2, \dots, S_n]$, tel que S_1 est la tâche initiale, S_n est la tâche finale et pour chaque tâche S_i $(1 < i < n)$ on a:

- S_i est un successeur direct d'un des tâches $[S_i, \dots, S_{i-1}]$
- S_i n'est pas un successeur direct de n'importe quel tâche parmi les tâches suivants $[S_{i+1}, \dots S_n]$.
- il n'existe aucune tâche S_j dans $[S_i, \dots, S_{i-1}]$ tel que S_j et S_i appartiennent à deux branches alternatifs du workflow.

On appel les tâches $S_1, S_2, \dots, S_n$ des classes des services ou des tâches.

Plan d'exécution (Execution plan) :

Un plan d'exécution est l'instanciation de toute tâche abstraite dans un chemin d'exécution avec les services concernés.

Un plan $p = \{< S_1, s_{11} >, < S_2, s_{22} >, \dots, < S_n, s_{n1} >\}$ est un plan d'exécution pour un chemin d'exécution W_e ssi :

- $\{S_1, S_2, \dots, S_n\}$ est un ensemble des tâches dans W_e .
- pour chaque paire $< S_i, s_{ij} >$ dans p, le service s_{ij} représente un service concret associé à la tâche S_i.

❖ On va d'abord présenter les critères de qualité dans le cadre des services élémentaires (stand alone), avant de tourner notre attention vers des services composites.

Les critères de qualité pour les services élémentaires :

On a 5 critères de qualité pour les services élémentaires :

(1) prix d'exécution, (2) durée d'exécution, (3) réputation, (4) fiabilité et (5) disponibilité. (Betouil, 2015)

☒ **Prix d'exécution :** on propose une opération op fournie par le service s, le prix d'exécution (s,) c'est le valeur qui à payer par le demandeur de service pour invoquer l'opération op.

☒ **Durée d'exécution :** soit une opération op fournie par le service s, la durée d'exécution (s,) mesure le délai prévu en secondes entre le moment où une requête est envoyée et le moment où les résultats sont reçus. La durée d'exécution est calculée en utilisant la formule suivante (Zeng et al ,2004) :

q_{du} (s,) =Tp(s,op) +$Ttrans$(s,op) ce qui signifie que la durée d'exécution est la somme du temps de traitement $Tprocess$ (s,op) et le temps de

transmission $Ttrans$ (s,) =($\sum_{i=1}^{n} Ti(s,op)$)$/n$. où (s,) est l'observation passé du temps de transmission, et n est le nombre de temps d'exécution observé dans le passé.

☒ **Fiabilité :** La définition de fiabilité (s) est la probabilité que la demande est correctement répondue dans les délais prévus maximum indiqué dans la description du service Web.

La fiabilité est une mesure ayant trait au matériel et / ou la configuration du logiciel de services Web et les connexions de réseau entre les demandeurs et les fournisseurs de service.

La valeur de la fiabilité est calculée à partir des données d'appels passés en utilisant l'expression $qrel$ (s) =($Nc(s)$)$/K$, où $Nc(s)$ est le nombre de fois que le service s a été livré avec succès dans le délai prévu maximum, et K est le nombre total d'invocations.

☒ **Disponibilité** (availability) : La disponibilité (s) d'un service s est la probabilité que le service est accessible. La valeur de la disponibilité d'un service s est calculée en utilisant l'expression suivante qav s =(s) /θ où $Ta(s)$ est le montant total du temps (en secondes) dans lequel le service s est disponible dans les dernières θ secondes (θ est un constant fixé par un administrateur de la communauté de services).

☒ **Réputation** : La réputation (s) d'un service s est une mesure de sa fiabilité. Elle dépend essentiellement de l'expérience de l'utilisateur final d'utiliser le service s. Les utilisateurs finaux peuvent avoir des opinions différentes sur le même service. La valeur de réputation est définie comme le classement moyen donné au service par les utilisateurs finaux, c'est-à-dire $qrep$=($\sum_{i=1}^{n} Ri$)/n où Ri est le classement de l'utilisateur final sur la réputation du service.

Le vecteur de qualité d'une opération op d'un service s est défini par l'expression suivante

$$q\ (s,) = (q_{pr}\ (s,op)\ , q_{du}\ (s,op)\ , q_{av}\ (s)\ , q_{re}\ (s)\ , q_{rep}\ (s)\ )\ (1)$$

Il est à noter que la méthode pour calculer les critères de qualité n'est pas unique, plusieurs autres méthodes peuvent être utilisées.

Les critères de qualité pour les services composites

En peut résumer les fonctions d'agrégation des valeurs QoS d'un service composite pour calculer la valeur QoS globale de service composite dans le tableau suivent

critère	Fonction d'agrégation
Prix	$q_{pr}\ (p) = \sum_{i=1}^{n} q_{pr}(s_i, op\ (t_i))$
durée	$q_{du}\ (p) = CPA(p, q_{du})$
reputation	$q_{rep}(p) = \frac{1}{N} \sum_{i=1}^{n} q_{rep}(si)$
fiabilité	$qrel\ p = \prod_{i=1}^{n} q_{rel}(si$
disponibilité	$q_{av}\ (p) = \prod_{i=1}^{n} q_{av}(s_i)$

Tableau1: les fonction d'agrégations pour les services composites (Zeng et al, 2004).

☒**Pour le prix d'exécution :** Le prix d'exécution (p) d'un plan d'exécution c'est la somme des prix d'exécution des opérations invoquées des services qui participent à p. (par exemple comme fig. 2.2: le prix de vole plus le prix de location de voiture plus le prix d'hôtel).

☒**Pour le durer d'exécution :** La durée d'exécution (p) d'un plan d'exécution est calculée en utilisant (Critical Path Algorithm CPA) (Zeng et al, 2004).

☒**Pour la réputation :** La réputation q_{rep} (p) d'un plan d'exécution p est la moyenne de la réputation des services qui participent à p.

☒**Pour la fiabilité :** La fiabilité (p) d'un plan d'exécution p est le produit les nombres des fiabilités des services qui participent à p.

☒**Pour la disponibilité :** La disponibilité (p) d'un plan d'exécution p c'est le produit les nombres des disponibilité des services qui participent à p.

Le vecteur de qualité du plan d'exécution d'un service composite est défini comme :

$$q\ (p) = (q_{pr}\ (p)\ , q_{du}\ (p)\ , q_{av}\ (p)\ , q_{re}\ (p)\ , q_{rep}\ (p)\ )\ .\ (2)$$

2.4.1- L'approche de services basée QoS pour la composition de services web

Il existe plusieurs approches pour résoudre le problème de composition de services web, la majorité de ces approches supposent que les valeurs de qualité de service sont agrégées en un seul score, c'est-à-dire, elles considèrent une seule fonction objective qui associe des poids aux différents attributs de QOS.

On a 2 types possibles d'approches :

- L'approche globale
- L'approche locale

2.4.1.1- L'approche locale

Elle a pour objectif de choisir le meilleur Web service pour chaque tache individuelle a part entière en considérant des contraintes de QoS relatives a plutôt qu'en considérant des contraintes de QoS globales exprimées pour l'ensembles des taches. Il s'agit de sélectionner pour chaque tache un Web service apte a l'exécuter en tenant en compte les contraintes de l'utilisateur (zeng, 2004)

La figure (fig 2.4) montre l'approche de sélection globale

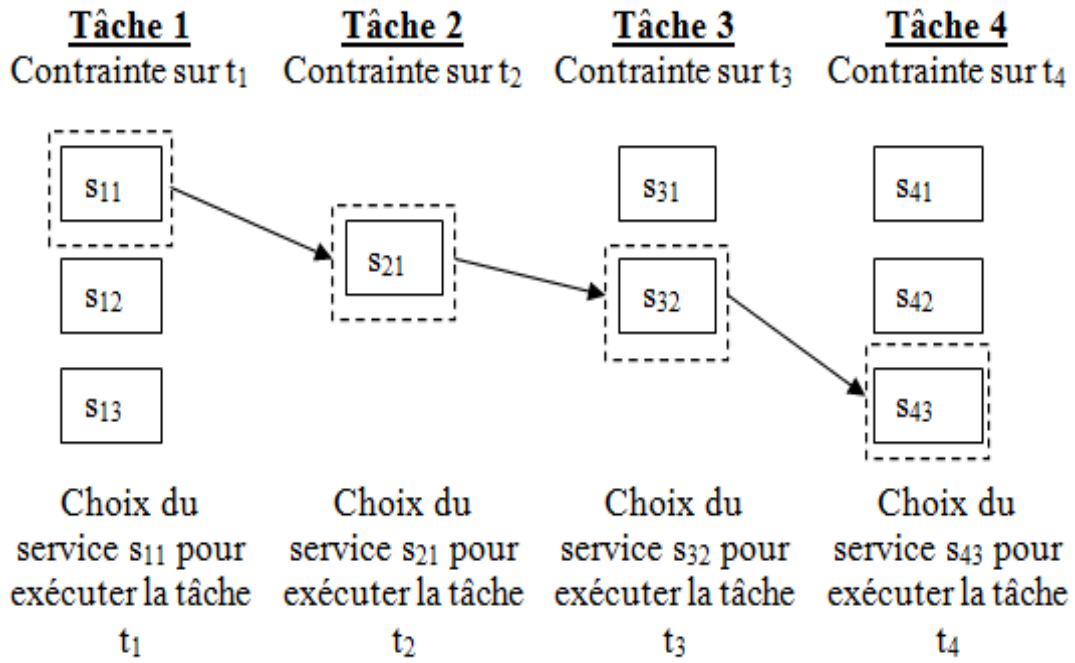


FIG 2.4: **sélection locale** (boudjelaba, 2012)

2.4.1.2- Sélection Globale

Contrairement à la méthode précédente, on choisit la combinaison de Web services qui garantit la meilleure qualité globale en tenant compte des contraintes de QoS et des préférences globales assignées pour l'ensemble des taches.

Afin de pouvoir appliquer une stratégie de sélection globale, on calcule la valeur des critères de QoS relative au service composite. Le calcul de ces critères s'appuie sur l'application de fonctions d'agrégation (qui peuvent être une somme, un produit, un minimum, . . .) Ensuite, on choisit la combinaison qui offre les meilleures valeurs de QoS parmi toutes les combinaisons possibles par rapport aux contraintes et aux préférences de l'utilisateur. La Fig.2 décrit la stratégie de sélection globale.

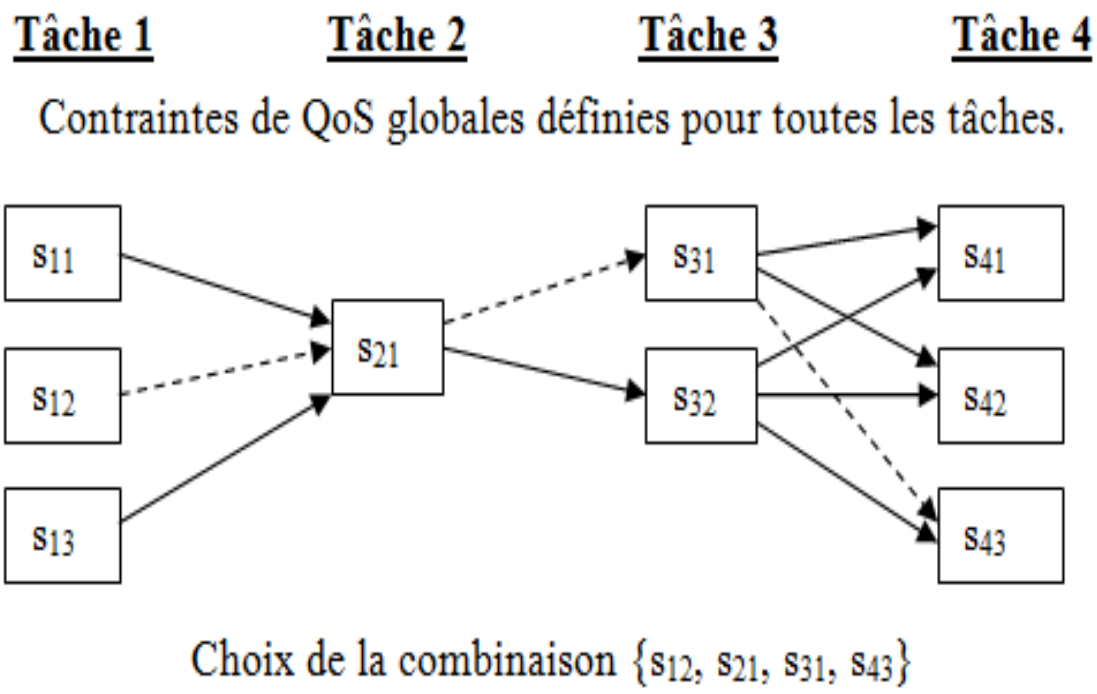


FIG 2.4 : sélection d'une composition de web service

Chapitre 3

Conception et

Implémentation

Deux parties dans ce chapitre, dans la première partie on présente notre conception qui consiste à proposer un algorithme basée sur les contraintes d'un décideur et leur ordre de préférence pour rechercher une solution de compromis entre les critères.

À la première tempe en basent sur la norme de Tchebycheff qui consiste à identifier le vecteur d'utilité le plus équilibré mais cette norme ignore l'ordre de préférence de décideur.

Alors en utilisent un algorithme basée sur les contraintes d'un décideur et leur ordre de préférence qui consiste à trier les alternatives en fonction d'un ordre lexicographique des préférences, cet ordre représente l'importance des critères pour l'utilisateur.

L'algorithme proposé a été appliqué au problème de la composition de service web pour la sélection d'un plan d'exécution qui satisfait autant que possible les exigences QoS et préférence des utilisateurs (maximisation/minimisation ; ordre de préférence)

Dans la deuxième partie en utilise le langage de programmation Visual basic pour applique cet algorithme et une base de donnée Access pour implémenter les trois services web et leurs qualité de service

3.1-Conception

Le problème à résoudre se caractérise par la présence de trois types d'alternatives et que chaque alternative soit composée de cinq sub-alternatives, alors le problème peut être décrit comme suit :

Étant donné un décideur qui a des préférences sur un ensemble de cinq critères (prix, durée, réputation, fiabilité, disponibilité) représentés par un pré ordre

Ce décideur a aussi des contraintes c= {c1, c2, c3, c4, c5} sur les valeurs des différents critères.

Comme un titre d'exemple, on a basé sur trois classes de services qui sont :

- **Réservation d'avion**
- **Location voiture**
- **Réservation d'hôtel**

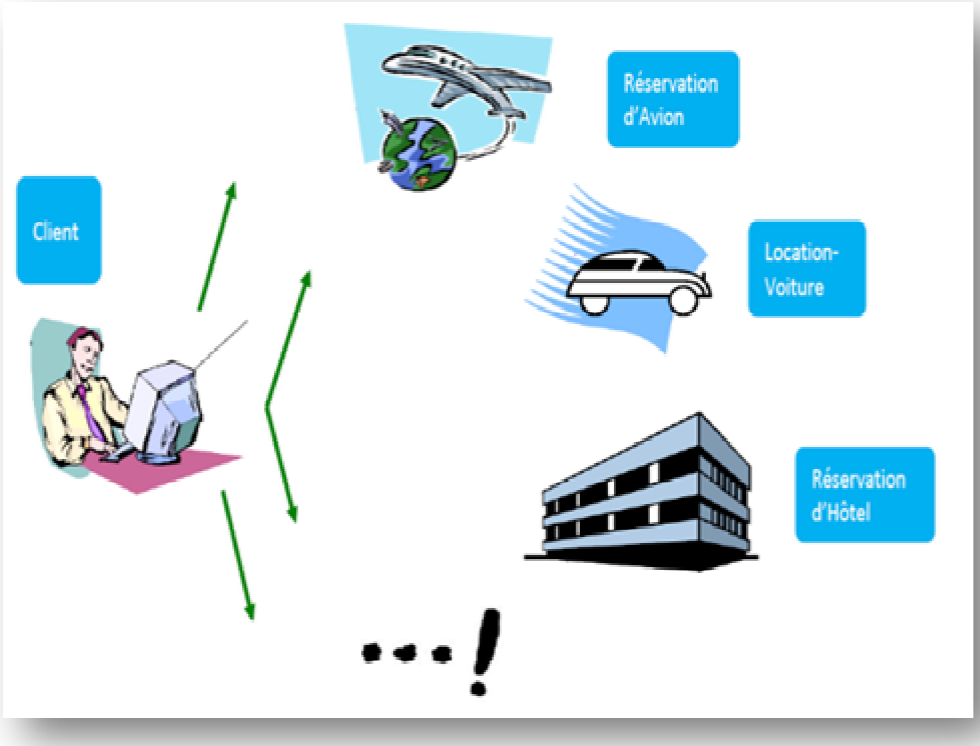


FIG3.1- Exemple de composition de services (Yu et bouguettaya.2010)

Dans la partie suivante, on va présenter notre conception.

3.2-les étapes de conception

3.2.1-Première Etape

Dans chaque classe de service on a plusieurs services candidats. Dans notre travail, on prend 50 services comme exemple, alors on présente trois tableaux chaque tableaux a 50 ligne, chaque ligne contient 6 colonnes (identificateur service web, prix, durée, réputation, fiabilité, disponibilité)

TBL_VOL					
ID	**prix**	**durée**	**reput**	**fiab**	**availability**
v1	131	60	482	85	16
v2	139	83	3321.4	89	1.4
v3	160	63	126.17	98	12
v4	134	97	107	87	1.9
v5	129	90	107.57	80	1.7
v6	164	99	255	98	1.3
v7	142	71	136.71	76	2.8
v8	136	73	102.62	91	15.3
v9	137	95	93.37	96	13.5
v10	132	76	133	86	7.7

Tab3.1 représenter le service web vole

TBL_LOC_VOIT					
ID	**prix**	**duree**	**reput**	**fiab**	**availability**
l1	137	65	145.25	56	11.8
l2	123	63	208.5	100	11.3
l3	135	91	305.4	89	1.4
l4	139	68	99	88	0.7
l5	126	73	539.72	23	4
l6	164	82	929	86	5.3
l7	157	75	401.14	43	4.1
l8	146	95	224	75	14.8
l9	156	63	424.57	50	1.7
l10	140	75	164.5	72	12.5

Tab3.2 représenter le service web voiture

TBL_HOTEL					
ID	**prix**	**duree**	**reput**	**fiab**	**availability**
h1	146	68	371.35	87	1.4
h2	159	75	206.2	87	3.7
h3	141	76	135.33	86	10.1
h4	148	94	144.8	91	18.8
h5	137	86	116.5	56	7
h6	158	68	60.84	70	10.5
h7	130	86	291	99	1.1
h8	164	94	151	93	3.7
h9	128	71	303.43	85	1.8
h10	149	95	298	99	10.3

Tab3.3 représenter le service web hôtel

3.2.2-Deuxième Etape

Faire toutes les compositions possibles de vecteur d'utilité de chaque service web avec fonction additive pour les critères prix et la durée et la fonction moyenne pour les autres critères. Dans notre cas, on a 50 lignes dans chaque tableau, alors le résultat contient 125000 alternatives possibles.

Voici l'algorithme de composition

```
Pour i=1 jusqu'à CN
  Pour j=1 jusqu'à CN
    Pour k=1 jusqu'à CN
        Début
              Comp := ( Vi.prix+ Hi.prix+ Ki.prix ;
                        Vi.dure+ Hi.dure+ Ki.dure ;
                        (Vi.reput+ Hi. reput + Ki. reput)/3 ;
                        (Vi.fiab+ Hi. fiab + Ki. fiab)/3 ;
                        (Vi.avail+ Hi. avail + Ki. avail)/3      ;) ;
        Fin ;
        Fin ;
        Fin.
```

Après l'application de cet algorithme on obtient notre espace d'alternatives A, chaque alternative est composée de plusieurs sub-alternative, une alternative est caractérisée par cinq critères chaque valeurs de ces cinq critères est calculée a partir des valeurs des critères des sub-alternatives en utilisent l'opération somme pour les deux critères (prix, durée) est l'opération moyenne pour les autres critères.

3.2.3-Troisième Etape

On applique les contraintes d'utilisateur sur les valeurs de cinq critères avec la minimisation pour la durée et le prix et la maximisation pour les trois autres critères alors On élimine tous les vecteurs qui ne respectent pas la première contrainte d'utilisateur et pour chaque nouveau ensemble de vecteur d'utilité éliminer les vecteurs qui ne respectent pas la deuxième contrainte et faire la même chose pour les autres contraintes , on obtient alors un espace d'alternatives dont chaque alternative respecte les contraintes d'utilisateur sur chaque vecteur résultat

Voici l'algorithme de l'élimination

Algorithme Elimination

```
Lire Prix ;  Lire Durée ;  Lire réputation ;  Lire fiabilité ;  Lire disponibilité ;
Début
    Procédure éliminationPrix  (Prix )
        Début
            Pour i=1 Jusqu'un  N faire
            Si tab_comp. Prix (i) > Prix  alors
                Sup_enregestrement
            FinSI
        Fin
    Procédure élimination Durée  (Durée )
        Début
            Pour i=1 Jusqu'un  N faire
            Si tab_comp. Durée  (i) > Durée  alors
                Sup_enregestrement
            finSI
        Fin
```

```
Procédure élimination réputation    (réputation  )
        Début
                Pour i=1 Jusqu'un  N faire
                Si tab_comp. réputation   (i) < réputation    alors
                        Sup_enregestrement
                finSI
        Fin
Procédure élimination fiabilité    (fiabilité  )
        Début
                Pour i=1 Jusqu'un  N faire
                Si tab_comp. fiabilité   (i) < fiabilité    alors
                        Sup_enregestrement
                finSI
        Fin

Procédure élimination disponibilité    (disponibilité )
        Début
                Pour i=1 Jusqu'un  N faire
                Si tab_comp. disponibilité   (i) < disponibilité    alors
                        Sup_enregestrement
                FinSI
        Fin
Fin
```

3.2.4-Quatrième Etape

Alors sur ces tableaux composites on applique au premier temps la norme de Tchebycheff

Puis on applique l'algorithme de recherche

Rappel :

La norme de Tchebycheff est un agrégateur qui consiste d'abord à identifier deux vecteurs (le point idéal et le point Nadir), puis à comparer les vecteurs selon leurs distances par rapport à ces points.

Et après l'application de la norme Tchebycheff, on a trouvé que cette dernière ne tienne pas en compte l'ordre de préférence de l'utilisateur, et elle se base seulement sur les contraintes d'utilisateur. Autrement, l'application de la norme Tchebycheff nous donne un résultat à valeurs équilibrés et elle est mieux adaptée aux concours de recrutement, car elle nous offre un résultat équilibré dans toutes les valeurs. Pour cette raison, qu'on est basé sur l'algorithme de recherche qui est basé sur les valeurs saisies par l'utilisateur et l'ordre de préférence. Alors le résultat va changer avec le changement de ce dernier.

3.2.5-Cinquième Etape

Les vecteurs d'utilité des services composites sont d'abord triés en fonction de préférences de l'utilisateur est c'est ça la déférence entre la norme de Tchebycheff est l'application de cet algorithme, dans notre cas on a : Cinque critères « Prix, Durée, réputation, fiabilité, disponibilité » et l'utilisateur indique l'ordre de préférence de chaque critère.

Le rang i d'un attribut de qualité dans un vecteur d'utilité doit correspondre à son rang dans les préférences utilisateur.

On note $U_{\succsim}$ le vecteur ordonné du vecteur d'utilité u.

Pa exemple, si les préférences de l'utilisateur sont représentées comme suit :

Prix > Durée > réputation > fiabilité > disponibilité

Et le vecteur d'utilité originale est u =(*Durée=200, disponibilité=100, réputation=300, Prix=500 ; fiabilité=150)*

Alors le vecteur ordonné de U est $U_{\succsim}$ =*(500 ;200 ;300;150;100)*

3.2.6-Sixième Etape

Une fois les vecteurs d'utilité sont obtenus, l'algorithme de recherche basé sur un prèodre lexicographique fonctionne comme suit :

Dans la première itération, les vecteurs d'utilité sont triés par ordre décroissant de l'utilité de l'attribut de qualité le plus préféré en fonction des préférences de l'utilisateur.

Si

On a seulement un vecteur d'utilité avec la plus grande utilité pour l'attribut de qualité le plus préféré

ALORS

L'algorithme se terminer **AFFICHER** le résultat ;

Sinon

Aller septième_ *Etape*

3.2.7-Septième Etape

Cette étape exécutée si on est confronté à plusieurs vecteurs avec la même valeur pour l'utilité de l'attribut le plus préféré de qualité, alors on utilise un péordre lexicographique pour trier les meilleurs vecteurs d'utilité les mieux classés.

Le principe est comme suit :

On prend la version ordonnée des L vecteurs les mieux classés, qui ont la même utilité pour l'attribut le plus préféré , ($U^1_{\gtrsim}$, $U2_{\gtrsim}$, ..., $U^L_{\gtrsim}$) .

Pour simplifier la notation , on va omettre le symbole $\gtrsim$ dans la version ordonnée des vecteurs d'utilité ($U^1_{\gtrsim}$ =U^i) .

Puis on applique la recherche lexicographique pour ces vecteurs d'utilité ordonnés

$U^i = (U_1^j, U_2^j, \dots, U_{r1}^j)$ et $U^k = (U_1^k, U_2^k, \dots, U_r^k)$ ou r est le nombre d'attributs de qualité .

$$(u_1^j, u_2^j, \dots, u_r^j) \succ (u_1^k, u_2^k, \dots, u_r^k) \Leftrightarrow \exists m > 0, \forall i < m, (u_i^j = u_i^k) \wedge (u_m^j > u_m^k)$$

Autrement dit, si une des utilités des m^{iéme} attribut de qualité $U_m^k > U_m^k$ et tous les utilités précédents sont égales.

Plus simplement, on compare les premières utilités. Si elles sont égales, on compare les deuxièmes utilités et ainsi de suite.

La relation entre les premières utilités correspondantes qui ne sont pas égales détermine la relation entre les vecteurs d'utilité entiers.

Le vecteur de la plus haute valeur de l'utilité correspondante est celui qui est le mieux classé.

Après cette étape, un préordre lexicographique est obtenu, le vecteur résultat est acceptable car le tableau composé après l'élimination contient seulement les vecteurs qui respectent les contraintes de l'utilisateur.

3.2-Implémentation

Pour implémenter cette conception, on utilise Visual Studio version 2010

3.2.1-Définition :

Visual Studio est un environnement de développement intégré (IDE intégrâtes développent environnement) développé par Microsoft.

Un IDE est bien plus qu'un traitement de texte, il offre un cadre pour le développement de code , son test et sa mise au point et également un environnement pour son exécution (le framework.NET)

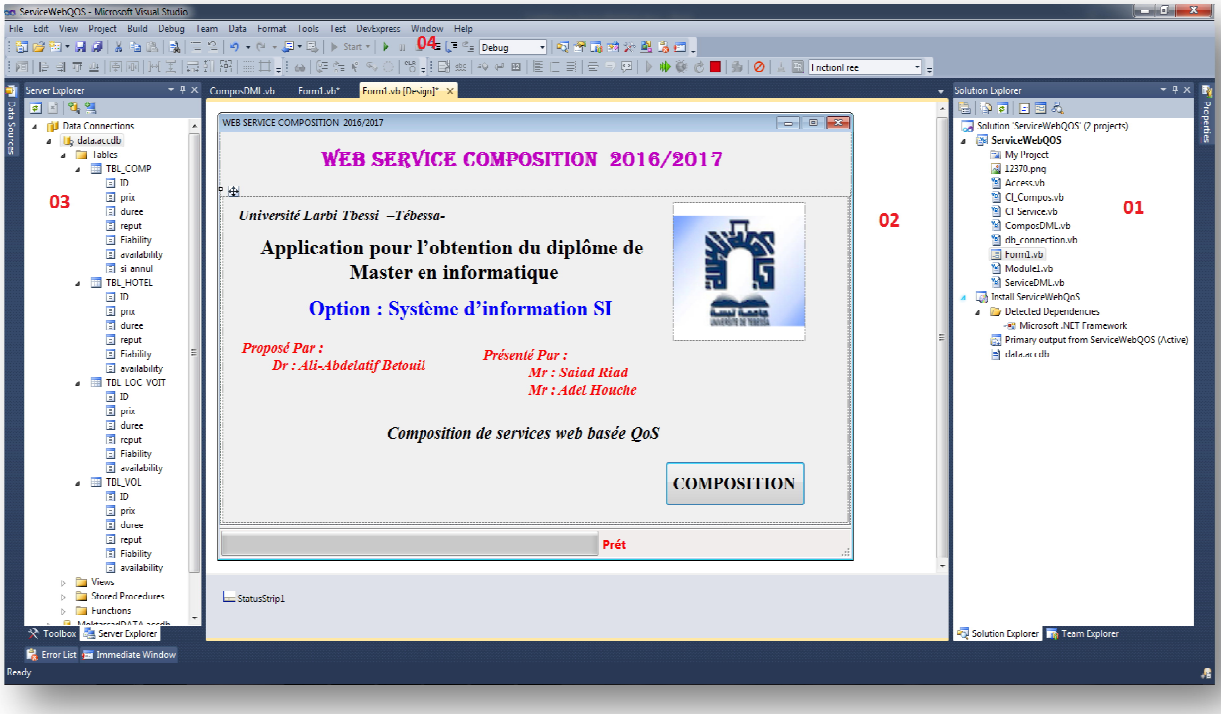


FIG 3.2 représenter l'environnement de développement intégré

3.2.2- Explication du l'environnement :

✓ 01 Explorateur de solution

L'Explorateur de solutions est une fenêtre Outil de l'environnement de développement intégré (IDE) de Visual Studio. Elle affiche le contenu d'une solution, notamment les projets de la solution et les éléments de chaque projet. Comme pour les autres fenêtres Outil de Visual Studio, vous pouvez en contrôler les paramètres physiques, notamment sa taille, son emplacement et sa nature ancrée ou flottante. (microsoft.com 2017)

✓ 02 Création des fenêtres

L'infrastructure crée automatiquement la plupart des fenêtres dont vous avez besoin dans un programme d'infrastructure. Mais si vous avez un but bien déterminé, vous pouvez créer vos propres fenêtres — y compris les fenêtres enfants de fenêtres frames ou de vues — en plus des fenêtres fournies par l'infrastructure. (microsoft.com 2017)

✓ 03 l'Explorateur de serveurs

Utiliser l'Explorateur de serveurs pour afficher et extraire des informations de toutes les bases de données installées sur un serveur. Vous pouvez afficher les tables, vues, procédures stockées et fonctions de bases de données dans l'Explorateur de serveurs. Vous pouvez aussi développer chaque table pour afficher ses colonnes et ses déclencheurs, ainsi que cliquer avec le bouton droit sur une table pour sélectionner le Concepteur de tables dans le menu contextuel . (microsoft.com 2017)

✓ 04-Barre d'outils

Affiche les barres d'outils disponibles dans l'environnement de développement intégré (IDE, *Integrated Development Environment*) et les barres d'outils que vous créez. Lorsqu'une barre d'outils est disponible dans l'environnement de développement intégré, une coche figure à gauche de son nom dans cette boîte de dialogue. (microsoft.com 2017)

3.2.3- QWS Dataset

On a testé notre système sur un ensemble de QWS Dataset. Dataset est un ensemble des services web avec leurs paramètres de qualité de services(QoS). Notre système est testé par la version (2.0) de QWS Dataset qui est développés par Eyhab AlMasri (ealmasri@uoguelph.ca) et Dr. Qusay H. Mahmoud (qmahmoud@uoguelph.ca) . Cette version représente 2507 services web qui existent dans le web, la dernière mise à jour : 05 juin 2008. La figure suivante montre une partie du QWS Dataset qu'on a utilisé.

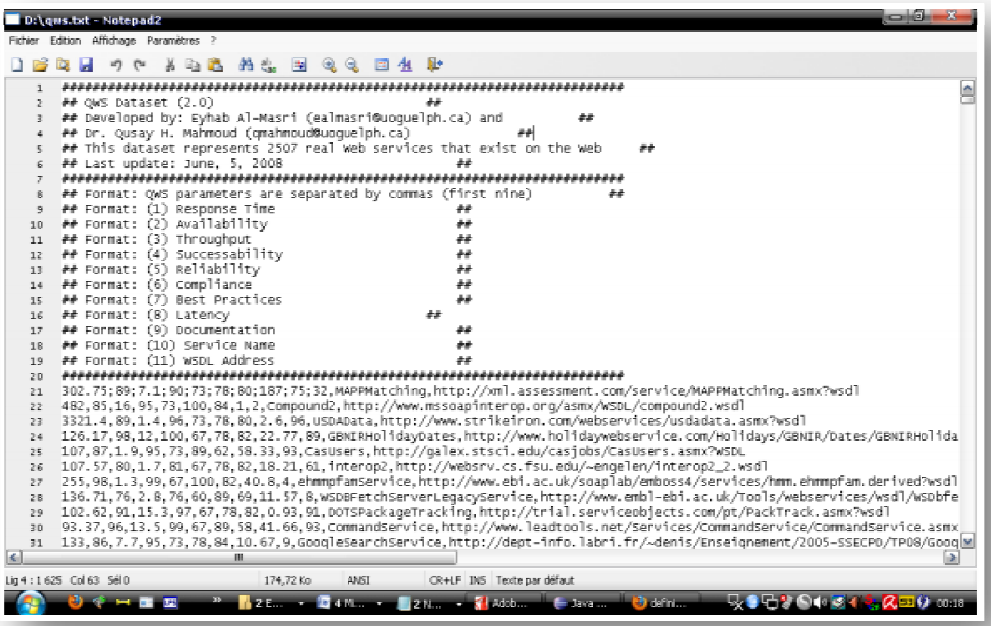


FIG 3.3- QWS Dataset version 2.0

3.2.4-Code source de l'implémentation

3.2.4.1-code source pour crée la table composition

Faire toutes les compositions possibles de vecteur d'utilité de chaque service web avec fonction additive (c.Prix = v.Prix + h.Prix + l.Prix) pour le critère prix, la durée et fonction moyenne (c.Fiability = (v.Fiability + h.Fiability + l.Fiability)/ 3) pour les autres critères, on fait la composition à chaque fois qu'on utilise l'application par ce que il n'existe pas une seule composition, à chaque fois une mise à jour du fichier data set dess service web est disponibles.

Voici le code source qui fait la fonction composition

```
SubGet_Comp()
ForEach v AsCl_ServiceInListVol
ForEach h AsCl_ServiceInListHotel
ForEachl AsCl_ServiceInListVoit
Dim c AsNewCl_Compos
        c.ID = (v.ID.ToString&"|"&h.ID.ToString&"|"&
l.ID.ToString).ToString
```

```
c.Prix = v.Prix + h.Prix + l.Prix
c.Duree = v.Duree + h.Duree + l.Duree
c.Reput= (v.Reput + h.Reput + l.Reput)/3
c.Fiability = (v.Fiability + h.Fiability + l.Fiability)/ 3
c.Availability = (v.Availability + h.Availability +
l.Availability) / 3
CMD_cmp.Obj = c
CMD_cmp.InsertInto()
Next
Next
Next
EndSub
```

3.2.4.2-code source éliminer

Dans la table composition on utilise la troisième colonne pour garder la même table et cette option pour éviter chaque fois la création de la table composition, alors si le vecteur est acceptable selon les contraintes d'utilisateur on ajoute 0 dans le champ sinon on ajoute 1

Voici le code pour éliminer tous les vecteurs non acceptables selon les contraintes d'utilisateur

```
Public TableName AsString = "TBL_COMP"
SubEliminerParPrix(ByVal Prix AsDouble)
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and prix > "&Replace(Prix, ",", "."))
EndSub

SubEliminerParDuree(ByValDureeAsDouble)
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and duree> "&Replace(Duree, ",", "."))
EndSub

SubEliminerParReput(ByVal Reput AsDouble)
```

```
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and reput < "&Replace(Reput, ",", "."))
EndSub

SubEliminerParFiability(ByValFiabilityAsDouble)
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and Fiability< "&Replace(Fiability, ",", "."))
EndSub

SubEliminerParAvailability(ByValAvailabilityAsDouble)
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and availability< "&Replace(Availability, ",", "."))
EndSub


SubEliminerLesCOMPOS(ByVal Compo AsCl_Compos)
Dim cmd AsNewComposDML
cmd.EliminerParPrix(Compo.Prix)
cmd.EliminerParDuree(Compo.Duree)
cmd.EliminerParReput(Compo.Reput)
cmd.EliminerParFiability(Compo.Fiability)
cmd.EliminerParAvailability(Compo.Availability)
EndSub
```

3.2.4.3-code source Tchebycheff

Après l'élimination, on a importer tous les vecteurs acceptables de la table composition c-à-d les vecteurs que leurs champs contiennent 1 , dans un nouveau variable List et faire ca avec l'instruction (If list.Count> 0 Then)

PIDEAL.(Duree.Reput.Fiability.Availability) nouvelle variable qui contient le max de chaque critère c-à-d le point idéal

PNADIR.(Duree.Reput.Fiability.Availability) nouvelle variable qui contient le min de chaque critère c-à-d le point nadir

Voici le code source qui fait ca

```
 FunctionGetPoinIDEAL() AsCl_Compos
Dim list As List(OfCl_Compos) = GetAllList()
Dim pIdl As New Cl_Compos
If list.Count> 0 Then
With pIdl
 .ID = "PIDEAL"
.Prix = (From x In list Selectx.Prix).Max
.Duree = (From x In list Selectx.Duree).Max
.Reput = (From x In list Select x.Reput).Max
.Fiability = (From x In list Select x.Fiability).Max
.Availability = (From x In list Select x.Availability).Max
.SiAnnuler = 0
EndWith
EndIf
ReturnpIdl
EndFunction

FunctionGetPoinNADIR() AsCl_Compos
DimlistAsList(OfCl_Compos) = GetAllList()
DimpNdrAsNewCl_Compos
Iflist.Count> 0 Then
WithpNdr
            .ID = "PNADIR"
.Prix = (From x InlistSelectx.Prix).Min
.Duree = (From x InlistSelectx.Duree).Min
.Reput = (From x InlistSelectx.Reput).Min
.Fiability = (From x InlistSelectx.Fiability).Min
.Availability = (From x InlistSelectx.Availability).Min
.SiAnnuler = 0
EndWith
EndIf
ReturnpNdr
```

```
EndFunction
PublicFunctionGetAllListTransformerTchibichaff() AsList(OfCl_Compos)
DimpidlAsCl_Compos = GetPoinIDEAL()          333
DimpndrAsCl_Compos = GetPoinNADIR()

DimlstAsNewList(OfCl_Compos)
DimtblAsDataTable = ExecutReader(GetSelectAll())
DimdRowAsDataRow
lst.Clear()
ForEachdRowIntbl.Rows
DimrowAsNewCl_Compos
Withrow
.ID = dRow("ID")
.Prix = (pidl.Prix - dRow("PRIX")) / (pidl.Prix - pndr.Prix)
.Duree = (pidl.Duree - dRow("Duree")) / (pidl.Duree - pndr.Duree)
.Reput = (pidl.Reput - dRow("Reput")) / (pidl.Reput - pndr.Reput)
.Fiability = (pidl.Fiability - dRow("Fiability")) / (pidl.Fiability - pndr.Fiability)
.Availability = (pidl.Availability - dRow("Availability")) / (pidl.Availability -
pndr.Availability)
EndWith
lst.Add(row)
Next
Returnlst
EndFunction
FunctionGetBestResultasCHIBICHEF(ByValnbrAfficherAsInteger) AsList(OfCl_Compos)
DimlstAsList(OfCl_Compos) = GetAllListTransformerTchibichaff()
' Dim l AsList(Of Cl_Compos) =
Return (From x InlstOrderByx.GetMaximumValue).Take(nbrAfficher).ToList
EndFunction
```

3.2.4.4-code source Algorithme de recherche

On a utilisé un algorithme basé sur les contraintes d'un décideur et son ordre de préférence qui consiste à trier les alternatives en fonction d'un ordre lexicographique des préférences, cet ordre représente l'importance des critères pour l'utilisateur

```
SubEliminieSequencePrix()
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and  prix> (select min(prix) from "& TableName &" WHERE si_annul='0')")
EndSub

SubEliminieSequenceDuree()
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and  duree> (select min(duree) from "& TableName &" WHERE
si_annul='0')")
EndSub

SubEliminieSequenceReput()
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and  reput< (select max(reput) from "& TableName &" WHERE
si_annul='0')")
EndSub

SubEliminieSequenceFiability()
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and  Fiability< (select max(Fiability) from "& TableName &" WHERE
si_annul='0')")
EndSub

SubEliminieSequenceAvailability()
Access.ExecuteNonQuery("UPDATE "& TableName &" SET si_annul='1' WHERE
si_annul='0' and  availability< (selectmax(availability) from "& TableName &" WHERE
si_annul='0')")
EndSub
```

3.2.5- temps d'exécution d'algorithme

Ce que l'on entend par complexité des algorithmes est une évaluation du coût d'exécution d'un algorithme en termes de temps (complexité temporelle) Ce coût d'exécution dépend de la machine sur laquelle s'exécute l'algorithme, de la traduction de l'algorithme en langage exécutable par la machine.

Mais nous ferons ici abstraction de ces facteurs, pour nous concentrer sur le coût des actions résultant de l'exécution de l'algorithme, en fonction d'une "taille" *n* des données traitées. Ceci permet en particulier de comparer deux algorithmes traitant le même calcul. Nous verrons également que nous sommes plus intéressés par un comportement asymptotique (que se passe -t'il quand *n* tend vers l'infini?) que par un calcul exact pour n fixé. Enfin, le temps d'exécution dépend de la nature des données (par exemple un algorithme de recherche d'une valeur dans un tableau peut s' arreter des qu'il a trouvé une occurrence de cette valeur. Si la valeur se trouve toujours au début du tableau, le temps d'exécution est plus faible que si elle se trouve toujours à la fin). Nous nous intéresserons d'abord dans ce qui suit à la complexité en "pire des cas" , qui est une manière, pessimiste, d'ignorer cette dépendance (on evaluera le temps d'exécution, dans le cas évoqué ci-dessus, en supposant qu'il faut parcourir tout le tableau pour trouver la valeur cherchée).

3.2.5.1-Expérimentations

On a effectué des simulations pour évaluer la performance de l'algorithme proposé et la norme de Tchebycheff alors en calcule le temps d'exécution pour créer la table composition et dans notre cas 50 lignes dans chaque tableau, donc la table composition contient 125000 lignes ce temps est le même pour les deux approches.

Voici le code source pour calculer le temps d'exécution de chaque approche :

```
startTim = DateTime.Now.Ticks

….. le procédure exécution pour chaque approche

endTim = DateTime.Now.Ticks

Dim timeElapsed As TimeSpan = New TimeSpan(endTim - startTim)
```

La façon pour calculer le temps d'exécution est simple, On garde le premier temps et le dernier temps et on fait la soustraction.

Dans notre cas, on trouve que l'algorithme de recherche est plus performant que l'algorithme de Tchebycheff en fonction de temps.

Pa exemple en trouve les résultats suivants

Tempe d'exécution de l'algorithme de recherche = 0.631836 sec

Tempe d'exécution de l'algorithme Tchebycheff = 0.835421 sec

3.2.6-interface d'application

L'interface d'application contient un seul bouton nomé "composition " qui fait la composition de trois tableaux dans un seul tableau (voir 3.2.3.1 page 10) et contient la barre d'état qui indique le nombre de vecteur composé

FIG 3.4 interface d'application

Lorsqu'on clique sur le bouton de composition une fenêtre s'affiche juste après la création de la table de composition, cette fenêtre permet à l'utilisateur de saisir les contraintes sur les valeurs.

FIG 3.5 fenêtre de saiser les contraintes sur les valeur

Lorsqu'on clique sur le bouton Valider une fenêtre s'affiche pour permettre à l'utilisateur de choisir la méthode de recherche

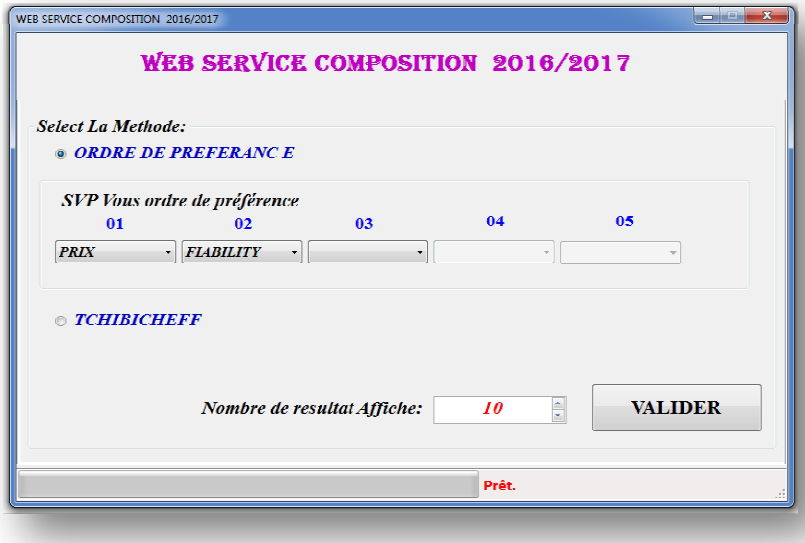


FIG 3.6 fenêtre pour choisie la méthode de recherche

Lorsqu'on clique sur le bouton Valider une fenêtre s'affiche et contient le résultat en détails.

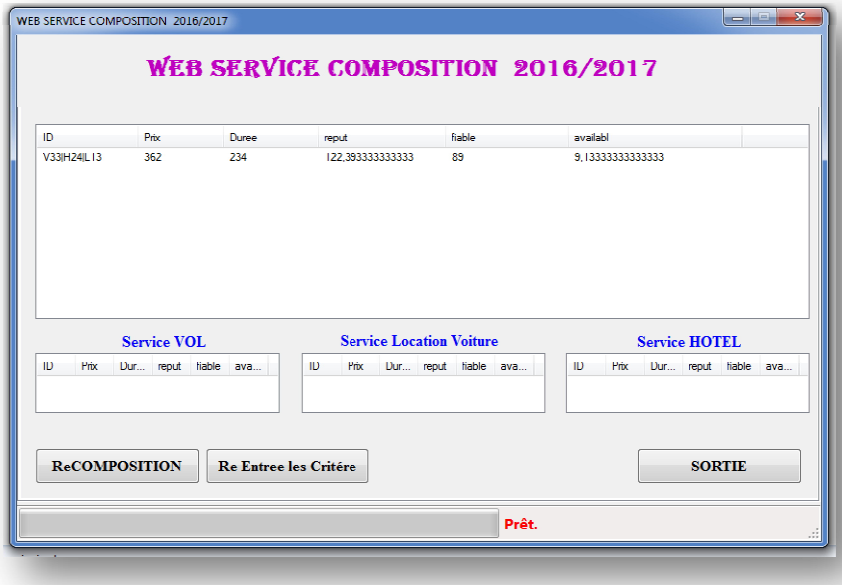


FIG 3.7 fenêtre pour afficher le résultat

3.2.7-les étapes d'installation

Les étapes d'installation sont simples comme touts les logiciels.

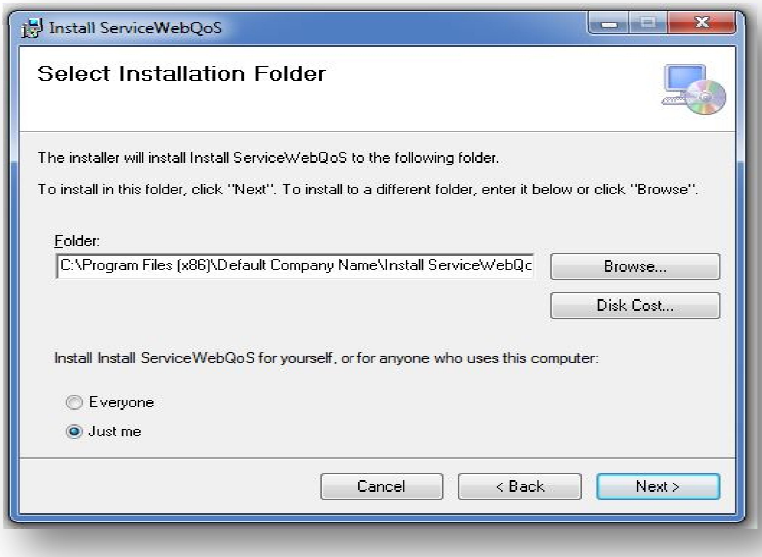


FIG 3.8 fenêtre d'installation

Conclusion

Nous avons proposé un cadre de composition automatique du service Web basée QoS qui est flexible dans la satisfaction de plusieurs exigences de QoS et considère également les préférences de l'utilisateur, et nous avons utilisé la norme de Tchebycheff comme un algorithme de décision multicritère pour choisir la meilleur composition.

Puis, nous avons utilisé un algorithme qui consiste à rechercher une solution Pareto optimale à un problème de décision multicritère en présence des contraintes sur les critères. Cet algorithme trie les alternatives en fonction d'un ordre lexicographique des préférences ; cet ordre représente l'importance des critères pour l'utilisateur.

L'algorithme proposé a été appliqué au problème de la composition de services web pour la sélection d'un plan d'exécution qui satisfait autant que possible les exigences QoS et préférences des utilisateurs.

Perspectives

1. l'utilisateur exprime ses préférences sur les critères de QoS par un preodre. Par exemple : $durée \succ prix \succ réputation \succ disponibilité \succ fiabilité$. Ce genre de préférences est simple, parfois, l'utilisateur a des préférences plus complexes, comme les préférences conditionnelles, par exemple, pour une condition c_1, on a $durée \succ prix$, mais pour une autre condition c_2, on a $prix \succ durée$. L'approche proposée ne prend pas en compte ce genre des préférences complexes, donc, il est intéressant d'améliorer cette approche pour obtenir des bonnes solutions dans ce cas.
2. Les algorithmes proposés pour filtrer les plans d'exécution sont un peu couteux en temps, pour cela, l'amélioration de ces algorithmes ou la proposition des nouvelles techniques pour filtrer les différents plans d'exécution peut être considérée comme une amélioration importante de cette approche.
3. Les approches basées sur les algorithmes génétiques ont montré leur efficacité dans le domaine de la composition des services web. L'hybridation de notre approche avec d'autres approches basées sur les algorithmes génétiques peuvent donner des résultats meilleurs.

Bibliographie

- [Arenaza, 2006] NereaArenaza, Composition semi-automatique des Web services, Projet de Master, Ecole Polytechnique Fédérale de Lausane, Février 2006.
- Baligand F (2008). « Une Approche Déclarative pour la Gestion de la Qualité de Service dans les Compositions de Services ». Ecole Nationale Supérieure des Mines de Paris. Juin 2008.
- Betouil A.A (2015), «contribution à la recherche du meilleur compromis dans la décision multicritère » Thèse de doctorat. UNIVERSITE DE ANNABA, Algérie.
- boudjelaba, H (2012),«Sélection des web services Sémentiques » Thèse de magister. UNIVERSITE DE BEJAIA, Algérie
- Dubus J-P (2010). « Prise de décision multiattribut avec le modèle GAI ». Thèse de Doctorat, Université Pierre et Marie Curie. France
- [Dustdar et Schreiner, 2005] Dustdar, S., Schreiner, W. 2005. « A Survey on Web Services Composition », InJ.Web and Grid Services, Vol.l, No.l.
- Fodor, J et Roubens, M (1994) « FuzzyPreferenceModelling and Multi-CriteriaDecisionAid» , Kluwer Academic Publisher
- Grabisch, M. et Perny, P. (2003). « Agrégation multicritère ». In Bouchon-Meunier, B. et Marsala, C., éditeurs : Logique floue, principes, aide à la décision, chapitre 3, pages 81–120. ix, 17
- Hadjila F (2014), «Composition et interopération des services web sémantiques » Thèse de doctorat. UNIVERSITE DE TLEMCEN. Algérie.
- Kostreva, M et Ogryczak, W (1999) « Linearoptimizationwith multiple equitablecriteria », RAIRO Operations Research, vol. 33, pp. 275_297.
- microsoft.com 2017 site web officielle de micro soft
- [Osman et al., 2005] T. Osman, D. Thakker, and D. Al-Dabass. Bridging the Gap between Work-flow and Semantic-based Web services Composition.In Proc. of the Web Service Composition Workshop WSCOMPS05, 2005.
- [Ponge, 2004] Julien Ponge ; Comptabilité et substitution dynamique des web Services ; Mémoire de fin d'étude, Université Blaise Pascal Clermont II ; Juillet 2004.
- Queiroz, S (2008). « Modèles graphiques décomposables pour la décision individuelle et collective ». Thèse de Doctorat LIP6 - université Paris 6.

- [Ricardo, 2004]Ricardo DE LA ROSA-ROSERO ; Découverte et Sélection de Service Web pour une application Mélusine ; l'Institut d'Informatique et de Mathématiques Appliquées de Grenoble, le 15 septembre 2004
- [Sun et al., 2003] Sun, H., Wang, X., Zhou, B. and Zou, P. 2003. «Research and Implementation of Dynamic Web Services Composition », APPT 2003, LNCS 2834, Springer-Verlag Berlin Hbidelberg, pp.457--466.
- Yu Q, Bouguettaya A. (2010) « Foundations for Efficient Web Service Selection ». Springer Science+Business Media, 2010.
- Zeng L, Benatallah B, Ngu A.H.H, Dumas M, Kalagnanam J, and Chang H (2004) . « Qos-aware middleware for web services composition ». IEEE Transactions on Software Engineering, 30(5):311– 327.