Université Abou Bekr Belkaid جامعة أبي بكر بلقايد
تلمسان الجزائر

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: *Système d'Information et de Connaissances (S.I.C)*

Thème

La sélection multi objectifs des services web à base de recuit simulé

Réalisé par :

- **Benzina Youcef**
- **Bekaddour Hassen**

Présenté le 03 juillet 2012 devant le jury composé de MM.

- *A. BENAMAR* *(Président)*
- *HADJILA Fethallah* *(Encadreur)*
- *S. M. CHOUITI* *(Examinateur)*
- *M. MERZOUG* *(Examinateur)*

Année universitaire : 2011-2012

Table de matières

Liste des figures

Liste des tableaux

Introduction générale

Les services web sont devenues à nos jours une technologie très utiliser pour le développement, l'emploie ment et l'intégration des applications sur internet, et cette importance et dû à la révolution au niveau d'architecture des services web tel que l'architecture orientés services (SOA). Cette technologie se base sur un ensemble de standards qui se résume essentiellement à trois technologies, le WSDL pour décrire les services web, l'UDDI pour les découvrir et le SOAP pour invoquer ces services web.

Cependant plusieurs axes de recherche sur la sélection et l'optimisation des services web se déroule, tendis que l'optimisation multi objectifs est l'une des exigences pour la sélection des meilleurs services web et en augmentant la qualité de ces services ont prenons en considération un ensemble de critères, et sella est une conséquence de la grande compétitivité entre les fournisseurs des services d'une part et d'une autre part sella revient aux exigences des utilisateurs qui demandent toujours le parfait et le meilleur.

Plusieurs algorithmes sont proposés dans le domaine de sélection des services web pour résoudre le problème d'optimisation multi objectifs des services web afin de sélectionné selles qui permet l'optimisation d'un ensemble de critères et la vérification d'un ensemble de contraintes, et parmi ces algorithmes en trouve l'algorithme que nous l' avons choisi d'étudier dans notre mémoire: l'algorithme « Recuit Simulé » qui sélectionne et optimise les services web car elle a prouvé son efficacités dans des domaines différent (traitement d'images, conception des circuits électronique, l'organisation du réseau informatique ...) et sella grâce à sa simplicité et sa souplesse (on peut facilement y introduire des contraintes liées au problème étudié).

Le reste du manuscrit est structuré comme suite :

Chapitre I : une présentation des concepts et des principes de base sur les services web qui se repose essentiellement sur trois technologie (SOAP, WSDL, UDDI).

Chapitre II : une description de l'optimisation ainsi que leur type, et pour mieux comprendre en cite un exemple de motivation.

Chapitre III : ce chapitre est consacré pour l'implémentation et la réalisation de l'application qui permet la sélection multi objectif des services web à base d'algorithme recuit simulé.

CHAPITRE I : LES SERVICES WEB

I.1 Définition

D'une façon générale, la notion des services web désigne tout simplement une application déposée par un fournisseur des services sur internet, tel que les services d'allocation des voitures ou de réservation de billet d'avion.

Plusieurs définitions ont été proposées parmi eux celle du World Wide Web Consortium (W3C) **[W3C]** : << A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL Web Services Description Language). Other systems interact with the Web service in a manner prescribed by its description using SOAP(Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards >>

C'est à dire : « 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 (ex. 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(eXtensible Markup Language), en conjonction avec d'autres standards relatifs au web »

Dans cette définition le consortium W3C définit un service Web comme étant une application ou un composant logiciel vérifiant les propriétés suivantes :

* il est identifié par une URI (Uniform Ressource Identifier).

* ses interfaces et ses liens peuvent être décrits en XML.

* sa définition peut être découverte par d'autres services s Web.

* il peut interagir directement avec d'autres services Web `a travers le langage

XML et en utilisant des protocoles Internet.

I.2 Architecture des services web

I.2.1 Architecture de base

Une architecture de base pour les services web intègre trois facteurs essentiels

**** le fournisseur de service (service provider) :***

- définit le service publie
- sa description dans l'annuaire
- réalise les opérations

**** l'annuaire (discovery agency) :***

- reçoit et enregistre les descriptions de services publiées par les fournisseurs
- reçoit et répond aux recherches de services lancées par les clients

**** le client (service requestor) :***

- obtient la description du service grâce à l'annuaire
- utilise le service

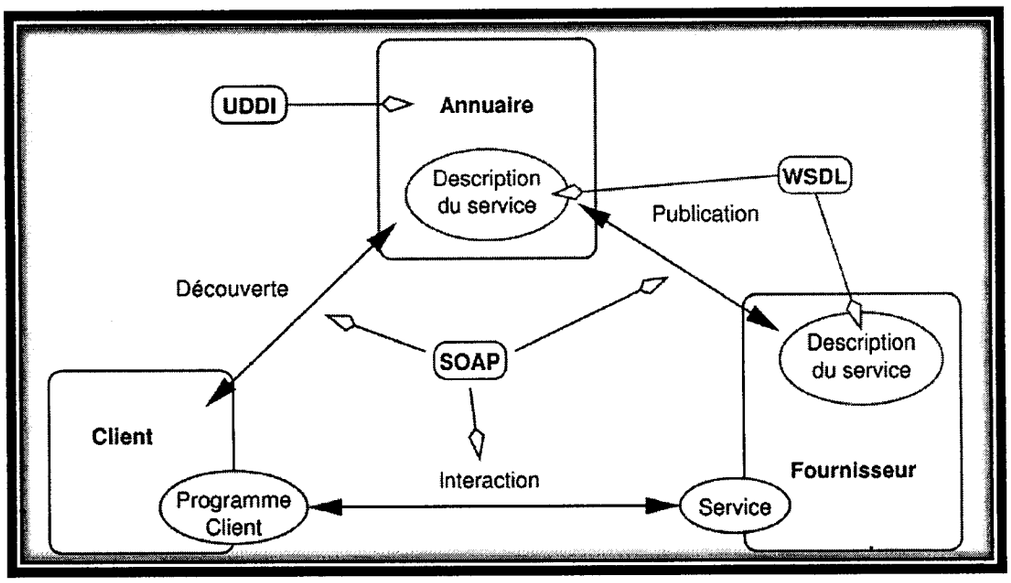


Figure I-1 Architecture de base des services web

Les interactions de base qui existent entre les trois éléments sont les opérations de découverte , publication et interaction .

I.2.2 Architecture orientée composant (SOA)

L'architecture des services Web est une architecture orientée composant (SOA Service Oriented Architecture). L'architecture SOA est un modèle qui définit un système par un e ns emble de composants logiciels distribués qui fonctionnent de concert afin de réaliser une fonctionnalité globale préalablement établie. Les composants dans un système distribué n'opèrent pas dans un même environnement de traitement et sont obligés de communiquer par échanges de messages afin de solliciter des services dans le but d'accomplir le résultat souhaité.

La définition de l'architecture des services Web consiste à mettre en évidences les concepts, les relations entre ces concepts ainsi qu'un ensemble de contraintes entre ces concepts. Les principaux concepts intervenant dans l'architecture des services Web sont:

– le fournisseur du service : désigne le serveur qui héberge les services déployés ;

– le client du service_: représente l'application cliente qui invoque le service ;

– le service : désigne les fonctionnalités d'un agent logiciel qui implémente le service;

– la description du service : c'es t la spécification du service exprimée dans un langage de description interprétable par les machines, c'es t-`a-dire une description technique dans laquelle le service est vu en termes de messages, de types, de protocoles de communication et d'une adresse physique ;

– les messages : c'est t la plus petite unité d'échange entre les clients et les services.

La structure des messages qui permettent l'invocation d'un service doit être exprimée dans la description du service ;

– la ressource : désigne l'identifiant du service, c'est- `a -dire son URI.

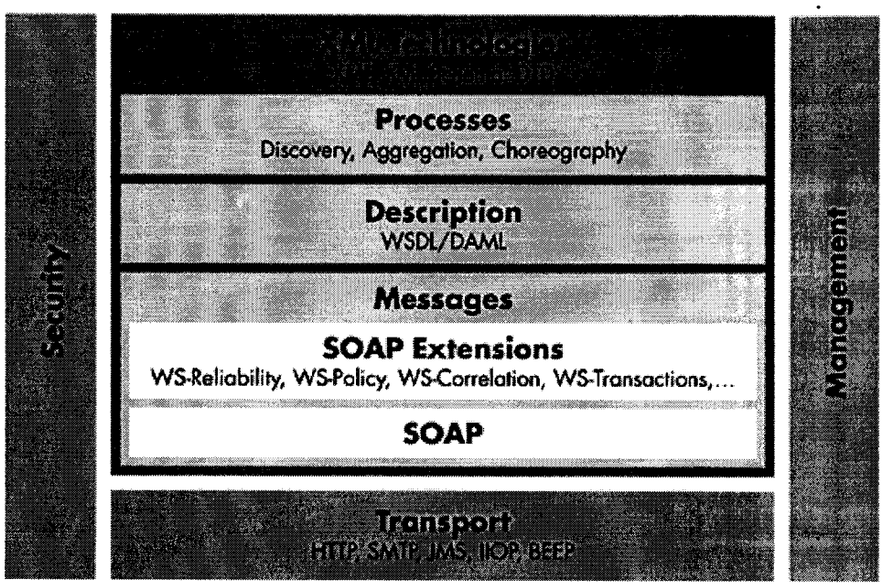


Figure I-2 Architecture des services Web.

Bien que le Web soit constitué de plates -formes totalement hétérogènes ou les intérêts des déférents acteurs du marché s'entremêlent, ceci ne l'a pas empêché de se développer et d'être universel. Ce succès est du essentiellement `a l'édition d'un ensemble de standards ouverts, dont le plus connu est le protocole HTTP (HyperText Transfer Protocol).

L'originalité de l'infrastructure des services Web consiste à mettre en place ces services exclusivement sur la bas e des protocoles les plus répandus sur Internet. Ces protocoles sont répartis selon quatre axes représentés comme suite :

– couche de transport : cette c ouche s'occupe de transporter les messages entre applications en utilisant les protocoles HTTP, FTP(File Transfer Protocol) ou SMTP.

– message XML : il s'agit de formaliser les messages à l'aide d'un vocabulaire XML commun (SOAP).

– description des services : description de l'interface publique des services web (WSDL).

– recherche de services : centralisation des services et de leur description dans un référentiel commun (UDDI Universal Description Discovery & Integration).

I.2.3 Description en couche des services Web

Les services Web emploient un ensemble de technologies qui ont été conçues afin de respecter une structure en couches sans être dépendante de façon excessive de la pile des protocoles. Cette structure est formée de quatre couches majeures :

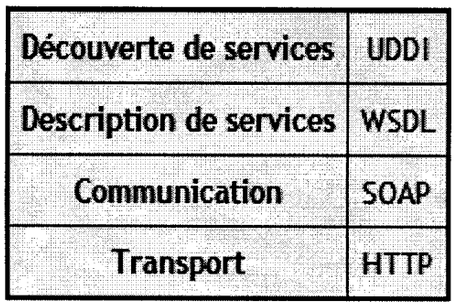


Figure I-3 Couches technologiques des services Web

* Le transport de messages XML est assuré par le standard HTTP.

* SOAP ou XML-RPC prévoit la couche de communication basée sur XML pour accéder à des services Web.

* La description d'un service Web se fait en utilisant le langage WSDL. WSDL expose l'interface du service.

* La publication et la découverte des services Web sont assurées par le biais du référentiel UDDI. Un référentiel UDDI est un catalogue de services Web.

Couche transport

Cette couche est responsable du transport des messages XML échangés entre les applications. Actuellement, cette couche inclut HTTP, SMTP(Simple Mail Transfer Protocol), FTP, et de nouveaux protocoles tels que BEEP (Blocks Extensible Exchange Protocol).

Couche communication

Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. Actuellement, deux styles architecturaux totalement différents sont utilisés pour ces échanges de données. Nous

avons d'un côté l'architecture orientée opérations distribuées (protocoles RPC Remote procedure call) basée sur XML et qui comprend XML-RPC et SOAP et de l'autre côté une architecture orientée ressources Web, REST (Representational State Transfer) qui se base uniquement sur le bon usage des principes du Web (en particulier, le protocole HTTP).

Couche description de service

Cette couche est responsable de la description de l'interface publique du service Web. Le langage utilisé pour décrire un service Web est WSDL qui est la notation standard basée sur XML pour construire la description de l'interface d'un service. Cette spécification définit une grammaire XML pour décrire les services Web comme des ensembles de points finaux de communication (ports) à travers lesquels on effectue l'échange de messages.

Couche publication de service

Cette couche est chargée de centraliser les services dans un registre commun, et de simplifier les fonctionnalités de recherche et de publication des services Web. Actuellement, la découverte des services est assurée par un annuaire UDDI (Universal Description, Discrovery, and Integration).

I.3 Les technologies des services web

Il existe quatre technologies essentielles utilisé dans les services web qui sont : XML, SOAP, WSDL, UDDI.

I.3.1 XML

XML (eXtensible Markup Language) est un format universel qui permet de structurer et d'organiser des documents et des données sur le Web. La version 1.0 de cette recommandation a été publiée par le W3C en février 1998 et ce format est devenu depuis incontournable. Ce succès est bien sûr lié au développement d'Internet mais il tient aussi en grande partie aux objectifs initiaux que le groupe de travail s'était fixé et qui tiennent en quelques mots : simple (à construire, à lire, à traiter), précis (pas d'ambiguïté, règles syntaxiques strictes), universel (conforme à Unicode, indépendant de la plate-forme logicielle), extensible. **[Christian Bernard et al. ,2003]**

Mais peut être la question qui se pose c'est comment le langage XML facilite l'utilisation des services web ?

* La séparation du contenu d'un document, de sa structure et de sa représentation facilite l'échange d'informations entre les différents partenaires.

* Permet la composition de services plus complexes à travers la définition des enchaînements entre les services grâce à BPEL4WS(Business Process Execution Language for Web Services version).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- feuille de style / -->
<?xml-stylesheet type="text/xsl" href="univ_xsl.xsl"?>
<!-- appel a un DTD externe/ -->
<!DOCTYPE universite SYSTEM "univ_dtd.dtd">
<universite nom="Université Abou Bekr Belkaid">
<faculte nom="Faculté de Médecine">
<departement1 nom="Département de Médecine">
Département de Médecine
</departement1>
<departement2 nom="Département de Pharmacie">
Département de Pharmacie
</departement2>
<departement3 nom="Département de Chirurgie Dentaire">
Département de Chirurgie Dentaire
</departement3>
</faculte>
```

I.3.2 SOAP

Le protocole SOAP (Simple Object Access Protocol) est un protocole de communication basé sur XML qui permet aux services Web d'échanger des informations dans un environnement décentralisé et distribué tout en s'affranchissant des plates-formes et des langages de programmation utilisés. Il définit un ensemble de règles pour structurer les messages envoyés. Mais SOAP ne fournit aucune instruction sur la façon d'accéder aux services Web. C'est le rôle du langage WSDL. **[Afef JMAL, 2009]**

Quand un message SOAP arrive dans une entité SOAP, il suivi le processus suivant :

1. Identifier toutes les parties du message SOAP destiné à cette entité.

2. Vérifiez que toutes les parties obligatoires identifiées dans le pas 1 sont soutenues par l'entité et traitez-les en conséquence. Si ce n'est pas le cas rejeter le message.

3. Si l'entité n'est pas la destination suprême du message enlèvent alors toutes les parties identifiées dans le pas 1 avant l'expédition du message.

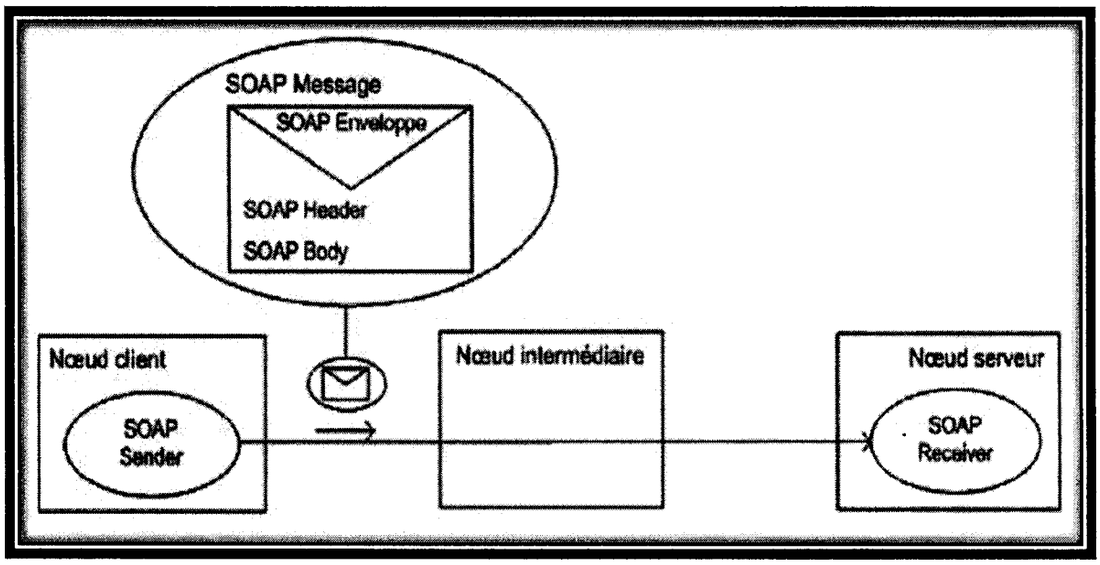


Figure I-4 fonctionnement d'un message SOAP

a. Structure de message SOAP

Un message SOAP est composé

Un de deux parties obligatoires **: l'enveloppe SOAP** et **le corps SOAP** ; et une partie optionnelle **: l'en-tête SOAP**. Et la figure suivante montre mieux la structure d'un message SOAP:

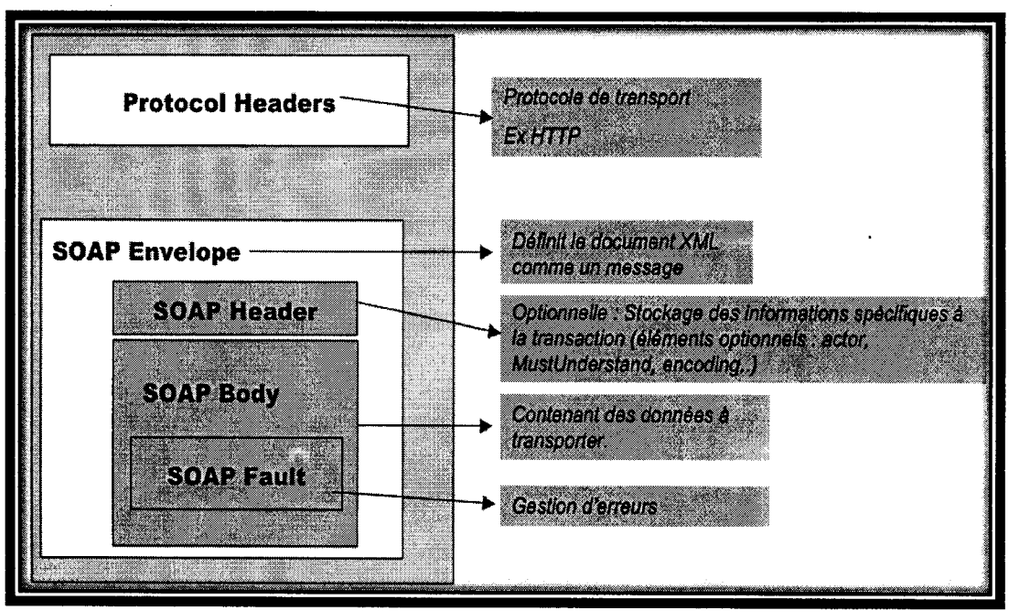


Figure I-5 la structure d'un message SOAP

Il existe deux types de message SOAP :

* Request : il incluse le nom de la méthode à invoquer et la liste de ses paramètres.

* Reponse : il inclut le résultat de la méthode invoqué dans le message Request.

b. Enveloppe SOAP (Envelope)

Un message SOAP est composé d'un élément obligatoire appelé le SOAP enveloppe.

L'enveloppe SOAP définit l'espace de nom (namespace : URI permettant de connaître la provenance de chaque balise) précisant la version supportée de SOAP. Cet espace de

nom est standard et permet de différencier les éléments du schéma (ex. http://www.w3.org/2001/06/soap-envelope/).**[Ricardo DE LA ROSA-ROSERO, 2004]**

Il peut contenir aussi un attribut encodingStyle dont la valeur est une URL vers un fichier de typage XML qui décrira les types applicables au message SOAP.

L'enveloppe SOAP est constituée d'un en-tête facultatif (SOAP header) et d'un corps obligatoire (SOAP body). Une description général du enveloppe SOAP et ses composants est donnée par l'exemple suivant :

```
<?xml version="1.0"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

 <soap:Header>

 <!-- optionnel -->
 ...
 </soap:Header>

 <soap:Body>

 <!-- requis -->
 ...

  <soap:Fault>
  ...
  </soap:Fault>

 </soap:Body>

</soap:Envelope>
```

c. En-tête SOAP (Header)

C'est un bloc optionnel qui contient des informations d'en-têtes sur le message. S'il est présent, ce bloc doit toujours se trouver avant le bloc Body à l'intérieur du bloc Enveloppe.

L'en-tête peut avoir plusieurs fils (SOAP blocks). Ces fils sont utilisés pour ajouter fonctionnalité au message comme l'authentification et la gestion des transactions. L'en-tête peut utiliser les attributs mustUnderstand et/ou SOAP actor pour indiquer comment traiter l'entrée et par qui.

SOAP définit cependant des attributs optionnels applicables sur les éléments fils directs du bloc Header. Ces attributs sont:

* **actor**: cet attribut identifie le destinataire de l'élément fils. En effet, un message SOAP peut traverser plusieurs composants avant d'atteindre le destinataire final du message. Aussi, il peut être nécessaire d'adresser des informations aux composants intermédiaires qui composent le chemin vers le destinataire final. C'est par l'intermédiaire de cet attribut que cela est rendu possible, en lui donnant comme valeur une URL qui correspond à ou aux destinataires de l'élément. En l'absence de cet attribut, l'élément fils est à destination du destinataire final du message SOAP.

* **mustUnderstand**: cet attribut indique si le destinataire de l'élément doit absolument comprendre l'élément ou non. Si cet attribut est à 1, alors le destinataire doit absolument comprendre l'élément. Si ça n'est pas le cas, un message d'erreur (i.e., contenant un bloc Fault) est retourné. Si cet attribut est à 0, l'élément peut être ignoré. C'est le comportement par défaut quand l'attribut n'est pas spécifié.

* **encodingStyle**: la signification de cet attribut est la même que pour le bloc Envelope. Sa portée reste cependant limitée à l'élément fils. Ici encore, cet attribut n'est pas permis par le profil basique.

```
<soap:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope/">
<soap:Header>
 <m:User xmlns:m="http://www.exemple.com/rights/"
      soap:actor="http://www.exemple.com/rights/RightsManager">
  Charles
 </m:User>
```

```
<m:Session xmlns:m="http://www.exemple.com/session/"
        soap:mustUnderstand="1">12AE3C</m:Session>
 <m:Lang xmlns:m="http://www.exemple.com/lang/"
       soap:actor="http://schemas.xmlsoap.org/soap/next"
       soap:mustUnderstand="0">
FR
 </m:Lang>
</soap:Header>
 < soap:Body> ... </ soap:Body>
</ soap:Envelope>
```

Ici, on voit que l'élément User est à destination d'un composant de gestion des droits (RightsManager) et qu'il contient un nom d'utilisateur.

L'élément suivant, Session, est lui à destination du destinataire final du message SOAP. Il indique un numéro de session, et doit absolument être compris par le destinataire pour pouvoir traiter le message.

Enfin, le dernier élément de l'en-tête, Lang est à destination du prochain composant sur le chemin et ne doit pas forcement être compris par ce composant. On notera que l'URL donnée à l'attribut actor pour désigner le prochain composant du chemin n'a normalement pas de signification avec la norme SOAP v1.1, car il s'agit d'une valeur spécifiée dans la norme SOAP v1.2.

d. Le corps SOAP (body)

Le corps SOAP contient l'information destinée au receveur. Il doit fournir le nom de la méthode invoquée par une requête ainsi que les paramètres associés à celle-ci, ou le nom de la méthode pour générer la réponse.

On peut cependant imaginer un exemple de deux contenus de blocs, le premier formulant une requête pour obtenir le prix d'une pomme:

```
<!-- exemple de demande -->
<soap:Body>
  <m:GetPrice xmlns:m="http://www.exemple.com/prices">
    <m:Item>Pomme</m:Item>
  </m:GetPrice>
</soap:Body>
```

Et le second représentant la "réponse" à cette requête:

```
<!-- exemple de réponse -->
<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.exemple.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>
```

Le corps SOAP peut contenir un SOAP **fault**. Ce bloque est utilisé pour transmettre l'information des erreurs durant le traitement du message.

Le bloc Fault : est le seul bloc définit pouvant être contenu dans le bloc Body. Il y est présent uniquement en cas d'erreur afin de remonter l'erreur à l'émetteur du message SOAP.

Ce bloc doit contenir 4 éléments:

* **faultcode**: cet élément contient un code qui identifie l'erreur. SOAP définit 4 codes d'erreur, à savoir, VersionMismatch (namespace du bloc Envelope non valide), MustUnderstand (un élément fils du bloc Header qui devait absolument être compris ne l'a pas été), Client (message mal formatée ou non valide), et Server (problème lors du traitement du message).

Le profil basique indique cependant que des codes d'erreurs supplémentaires peuvent être utilisés à condition que ceux-ci fassent parti d'un namespace spécifié (exemple: m:errorcode).

* **faultstring**: cet élément contient un texte décrivant brièvement l'erreur pour un humain.

* **faultactor**: cet élément contient le composant (i.e., son URL) qui a généré l'erreur.

* **detail**: cet élément contient des informations spécifiques à l'application et concernant l'erreur.

```
<soap:Body>
 <soap:Fault>
  <faultcode>soap:Server</faultcode>
  <faultstring>Impossible de router le message.</faultstring>
  <faultactor>http://www.exemple.com/messageDispatcher</faultactor>
  <detail>
   <m:error xmlns:m="http://www.exemple.com/errors">
    E_NO_ROUTE
   </m:error>  </detail>
 </soap:Fault>
</soap:Body>
```

Ici, l'erreur provient d'un composant de répartition des messages (messageDispatcher) qui n'a pas été en mesure de traiter le message (i.e., de le diriger vers le destinataire final). On constate le passage d'informations spécifiques à l'application dans l'élément détail.

I.3.3 WSDL

Comme son nom l'indique, WSDL permet de décrire des services web. Plus Précisément WSDL décrit de manière formelle les interfaces des services Web.
WSDL décrit un service web en deux étapes fondamentales : une étape abstraire constitué de la définition d'interface de service web, et une deuxième étape concrète qui est constitué d'implémentation de service web.

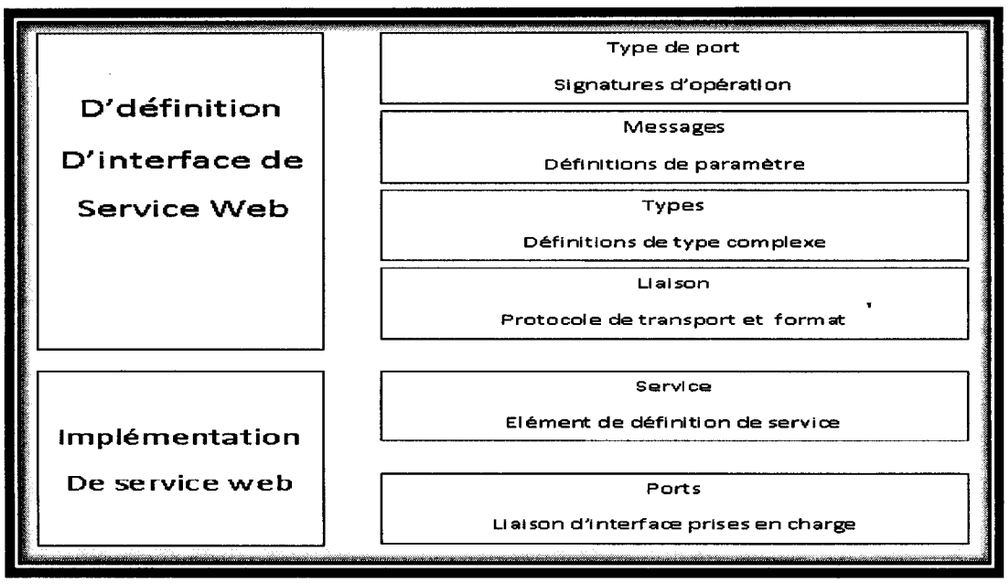


Figure I-6 Architecture WSDL

Un document WSDL se compose d'un ensemble d'éléments décrivant les types de données utilisés par le service, les messages que le service peut recevoir, ainsi que les liaisons SOAP associées à chaque message. Le schéma suivant illustre la structure du langage WSDL qui est un document XML, en décrivant les relations entre les sections constituant un document WSDL.

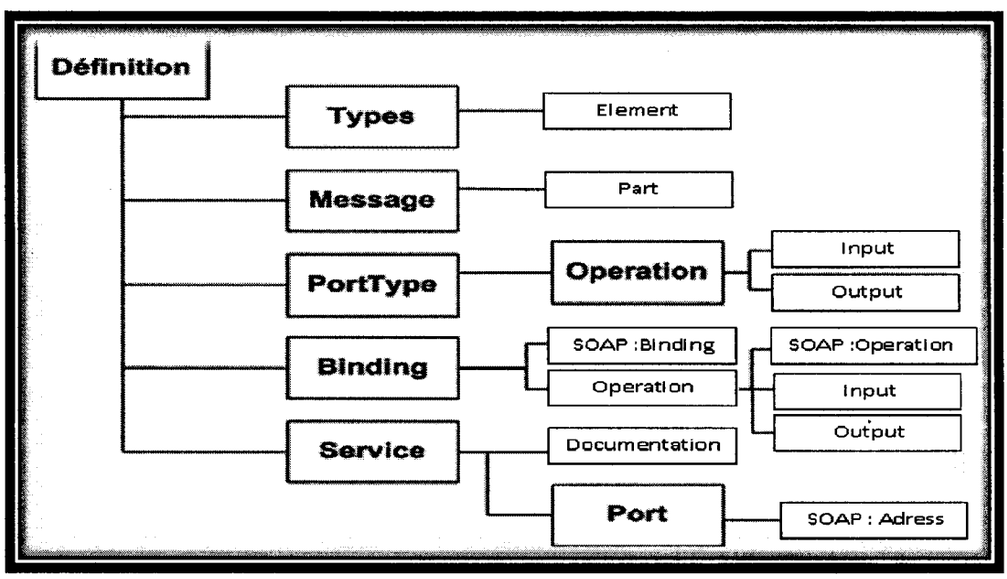


Figure I-7 Structure d'un document WSDL

Le tableau suivant reprend en détail chaque composant majeur d'un document WSDL :

Tableau I.1 Composition d'un fichier WSDL

Composant WSDL	Description
wsdl:definitions	Nœud racine du document qui donne le nom du service et déclare les différents espaces de nommage
wsdl:documentation	Contient de la documentation ou des commentaires (en général, du texte) destinés à être exploités par la personne souhaitant mettre en œuvre le Service Web. Peut être utilisé par n'importe quel composant WSDL
wsdl:types	Déclaration ou import de tous les types de données référencés par les attributs element ou type d'un message. A noter que c'est le plus souvent l'encodage XML Schéma qui s'applique (préfixé par l'espace de nommage xsd) en particulier dans le cas du modèle Document/Literal (le message est alors intégralement défini par un schéma XML)
wsdl:message	Un message est composé de un à plusieurs composants wsdl:part (réunis, ils définissent les paramètres du message ou les valeurs retournées par le message) et décrit un message unique (requête ou réponse)
wsdl:part	Fait référence à un paramètre ou à un ensemble de paramètres d'un message. Détermine le type de données par utilisation de l'attribut type ou element : On peut alors directement spécifier son type d'encodage dans le cas d'un paramètre unique (type) ou référencer le type de données présentes dans le composant wsdl:types (élément)
wsdl:portType	Définition abstraite des prototypes des méthodes à appeler. Combine plusieurs composants wsdl:message afin de spécifier une opération (composant wsdl:operation)

wsdl:operation	Définition abstraite du prototype d'une méthode. Chaque operation fait en général référence à un message en entrée (wsdl:input) et à un ou plusieurs messages en sorties (wsdl:output et wsdl:fault) selon le modèle d'interaction du message (one-way ou request-response)
wsdl:binding	Définition concrète de l'appel à une ou plusieurs méthodes. Spécifie le protocole de communication, le type de l'échange (RPC ou Document) et le style de chaque opération (Literal ou Encoded)
wsdl:service	Point d'entrée du document WSDL. Regroupe un ensemble de services définis par le ou les composants wsdl:port
wsdl:port	Définition d'un point d'entrée pour un service et référence le composant wsdl:binding permettant son traitement

a. Éléments d'une définition WSDL

La définition contient le nom du service décrit et les name spaces faisant référence aux types utilisés dans le document.

```
<definitions name="GetStockQuote
targetNamespace="http://advocatemedia.com/GetStockQuote.wsdl"
xmlns:myns = "http://advocatemedia.com/GetStockQuote.wsdl"
xmlns:myXsd = "http://advocatemedia.com/GetStockQuote.xsd"
xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap"
xmlns="http://schemas.xmlsoap.org/wsdl/">
</definitions>
```

La définition peut contenir les éléments suivants :

- **<types>** Contient les définitions de types utilisant un système de typage (comme XSD).

```
<!-- type defs -->
<types>
<xsd:schema targetNamespace="urn:xml-soap-address-demo"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<xsd:complexType name="phone">
    <xsd:element name="areaCode" type="xsd:int"/>
    <xsd:element name="exchange" type="xsd:string"/>
    <xsd:element name="number" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="address">
    <xsd:element name="streetNum" type="xsd:int"/>
   <xsd:element name="streetName" type="xsd:string"/>
  <xsd:element name="city" type="xsd:string"/>
   <xsd:element name="state" type="xsd:string"/>
   <xsd:element name="zip" type="xsd:int"/>
   <xsd:element name="phoneNumber" type="typens:phone"/>
</xsd:complexType>
</xsd:schema>
</types>
```

- **<message>** Décrit les noms et types d'un ensemble de champs à transmettre et peut être composé de plusieurs parties « **Part(nom, type ou element** ».

(part peut être défini comme un type (simple ou complexe) ou un élément).

```
<!-- message declns -->
<message name="AddEntryRequest">
<part name="name" type="xsd:string"/>
<part name="address" type="typens:address"/> </message>
<message name="GetAddressFromNameRequest">
<part name="name" type="xsd:string"/> </message>
<message name="GetAddressFromNameResponse">
<part name="address" type="typens:address"/>
</message>
```

- **<portType>** A chaque portType sont associées des opérations, correspondant aux méthodes. Pour chaque méthode on définit le message d'entrée et de sortie. Les opérations peuvent être de natures différentes : unidirectionnelle,

requête/réponse, sollicitation/réponse et notification

portType(name, operation(name,(input msg, output msg)))

```
<!-- port type declns -->
<portType name="AddressBook">
<!– One way operation -->
<operation name="addEntry"> <input message="AddEntryRequest"/>
</operation>
<!– Request-Response operation -->
<operation name="getAddressFromName">
<input message="GetAddressFromNameRequest"/>
<output message="GetAddressFromNameResponse"/>
</operation> </portType>
```

- **<binding>** Spécifie une liaison d'un <porttype> à un protocole concret (SOAP1.1, HTTP GET/POST, MIME, …).

Binding(name, type, (?:binding, style, transport),(operation(?:operation, ? :action (input(?:boby, encodingStyle, namespace, use), output(?:boby, encodingStyle, namespace, use))

```
<!-- binding declns -->
<binding name="AddressBookSOAPBinding" type="AddressBook">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="addEntry">
<soap:operation soapAction=""/>
<input> <soap:body use="encoded" namespace="urn:AddressFetcher2"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/> </input>
<output> <soap:body use="encoded" namespace="urn:AddressFetcher2"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/> </output>
</operation>
<operation name="getAddressFromName">
<soap:operation soapAction=""/>
<input> <soap:body use="encoded" namespace="urn:AddressFetcher2"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/> </input>
<output> <soap:body use="encoded" namespace="urn:AddressFetcher2"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
</operation>
</binding>
```

- **<service>** Une collection de points d'entrée (endpoint) relatifs et permet de spécifier le nom, où se trouve la documentation concernant le service mais aussi où envoyer les messages d'invocation.

Service(name, port(name, binding,?:address(location)),
documentation())

```
<service name="AdvocateMediaGetStockQuotes">
 <documentation>Simple Web Service to Retrieve
a stock quote</documentation>
<port name="GetStockQuotePort" binding="myns:GetStockQuoteBindingName">
<soap:address location="http://advocatemedia.com/GetStockQuote"/>
 </port> </service>
```

I.3.4 UDDI

L'annuaire des services UDDI (Universal Description Discovery and Integration) est un standard pour la publication et la découverte des informations sur les services Web. La spécification UDDI vise à créer une plate-forme indépendante, un espace de travail (framework) ouvert pour la description, la découverte et l'intégration des services des entreprises.

Dans un environnement ouvert comme Internet, la description d'un service Web n'est d'aucune utilité s'il n'existe pas de moyen de localiser aussi bien les services Web que leurs descriptions WSDL : c'est le rôle des référentiels UDDI. La spécification UDDI constitue une norme pour les annuaires de services Web. Les fournisseurs disposent d'un schéma de description permettant de publier des données concernant leurs activités, la liste des services qu'ils offrent et les détails techniques de chaque service. La spécification offre également une API aux applications clientes, pour consulter et extraire des données concernant un service et/ou son fournisseur.

a. Consultation de l'annuaire

L'annuaire UDDI se concentre sur le processus de découverte de l'architecture orientée services (SOA), et utilise des technologies standards telles que XML, SOAP et WSDL qui permettent de simplifier la collaboration entre partenaires dans le cadre des échanges commerciaux. L'accès au référentiel s'effectue de différentes manières.

ainsi qu'une référence à l'organisation proposant le service et un ou plusieurs « bindingTemplate ».

*** BindingTemplate (modèle de rattachement)**

UDDI permet de décrire des services Web utilisant HTTP, mais également des services invoqués par d'autres moyens (SMTP, FTP...). Les « bindingTemplates » donnent les coordonnées des services. Ce sont les pages vertes de l'annuaire UDDI. Ils contiennent notamment une description, la définition du point d'accès (une URL) et les éventuels « tModels » associés.

*** tModel**

Les « tModels » sont les descriptions techniques des services. UDDI n'impose aucun format pour ces descriptions qui peuvent être publiées sous n'importe quelle forme et notamment sous forme de documents textuels (XHTML, par exemple). C'est à ce niveau que WSDL intervient comme le vocabulaire de choix pour publier des descriptions techniques de services.

c. La recherche d'un service web

Les standards UDDI sont implémentés dans des API(Application Programming Interface), permettant l'accès et la manipulation, il existe deux types d'API (API d'interrogation, API de publication)

Cette API comporte neuf fonctions:

• **fonction find_binding** : cette fonction recherche l'existence d'une liaison spécifique à l'intérieur d'un service métier. Elle renvoie un message bindingDetail.

• **fonction find_business** : cette fonction recherche l'existence d'information sur une ou plusieurs entités métier. Elle renvoie un message businessList.

• **fonction find_service** : cette fonction recherche l'existence de services métier spécifiques à l'intérieur d'une entité métier. Elle renvoie un message serviceList.

• **fonction find_tModel** : cette fonction recherche l'existence d'un ou plusieurs services types. Elle renvoie un message tModelList.

• **fonction get_bindingDetail** : cette fonction recherche une information complète sur une liaison spécifique à l'intérieur d'un service métier. Elle renvoie un message bindingDetail.

• **fonction get_businessDetail** : cette fonction recherche une information complète sur une ou plusieurs entités métier. Elle renvoie un message businessDetail.

• **fonction get_businessDetailExt** : cette fonction recherche une information étendue sur une ou plusieurs entités métier. Elle renvoie un message businessDetailExt.

• **fonction get_serviceDetail** : cette fonction recherche une information complète sur un service métier spécifique à l'intérieur d'une entité métier. Elle renvoie un message serviceDetail.

• **fonction get_tModelDetail** : cette fonction recherche une information complète sur un service type. Elle renvoie un message tModelDetail. **[Christian Bernard et al. ,2003]**

I.4 Cycle de vie d'un service web

Le cycle de vie d'un service Web se déroule de la façon suivante : une fois créé, le service est déployé sur le réseau (local ou Internet). Puis un utilisateur ayant des besoins spécifiques va rechercher un service correspondant à ses besoins à l'aide d'un annuaire spécialisé. Enfin, une fois le service trouvé, l'utilisateur va invoquer le service : une communication va être mise en place entre l'utilisateur et le service Web. La figure suivante montre en détaille le cycle de vie d'utilisation :

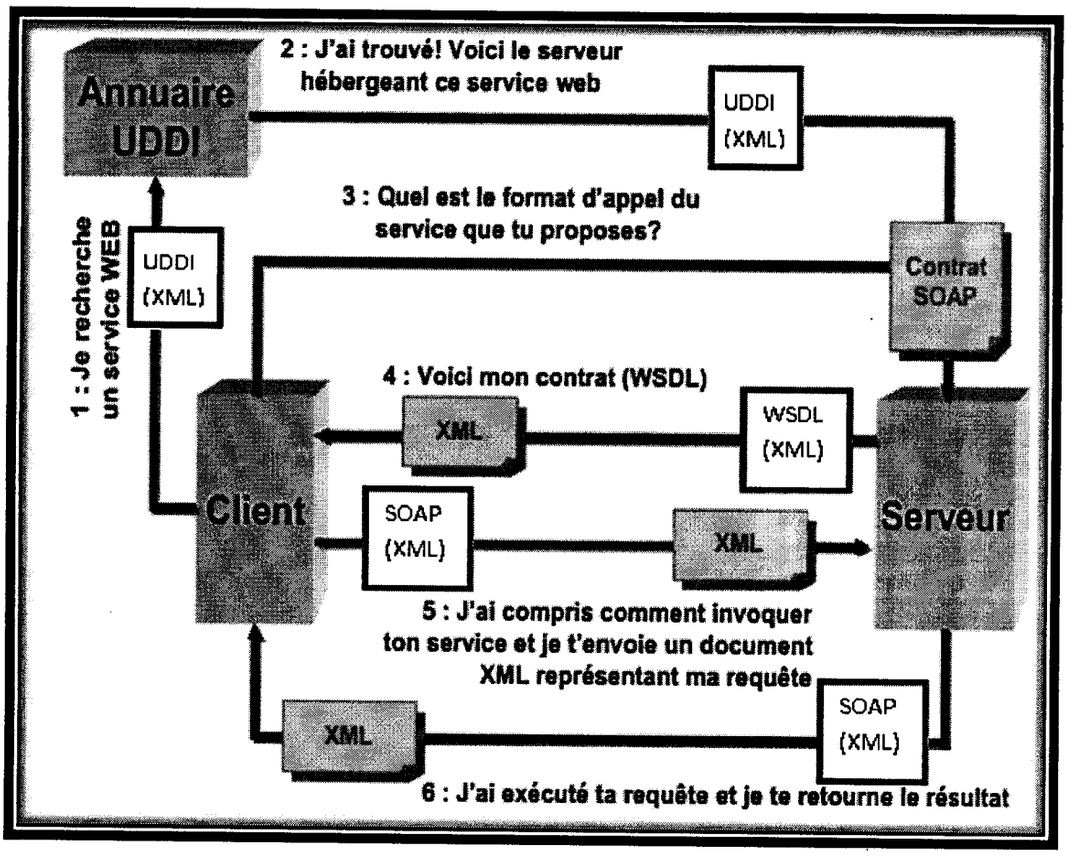


Figure I-8 Cycle de vie de service web

Ce cycle de vie fait appel à trois grandes technologies :

SOAP: Simple Object Access Protocol

WSDL: Web Services Description Language

UDDI: Universal Description Discovery & Integration

Et la figure suivante montre encor mieux le cycle de vie d'un service web

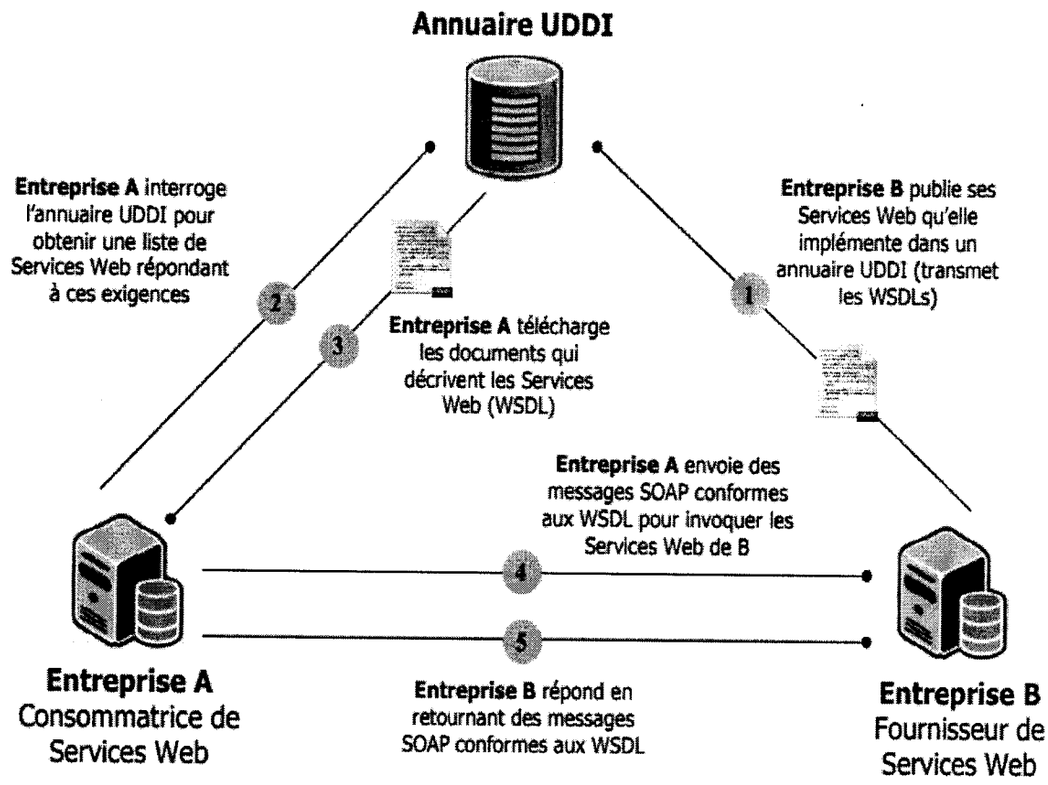


Figure I-9 le cycle de vie des services web

CHAPITRE II : ETAT DE L'ART SUR LA SÉLECTION

II.1 Exemple de motivation

L'exemple suivant concerne trois services :

- Un service de réservation de billet d'avion
- Un deuxième service de réservation hôtelière
- Un troisième service d'allocation de voiture

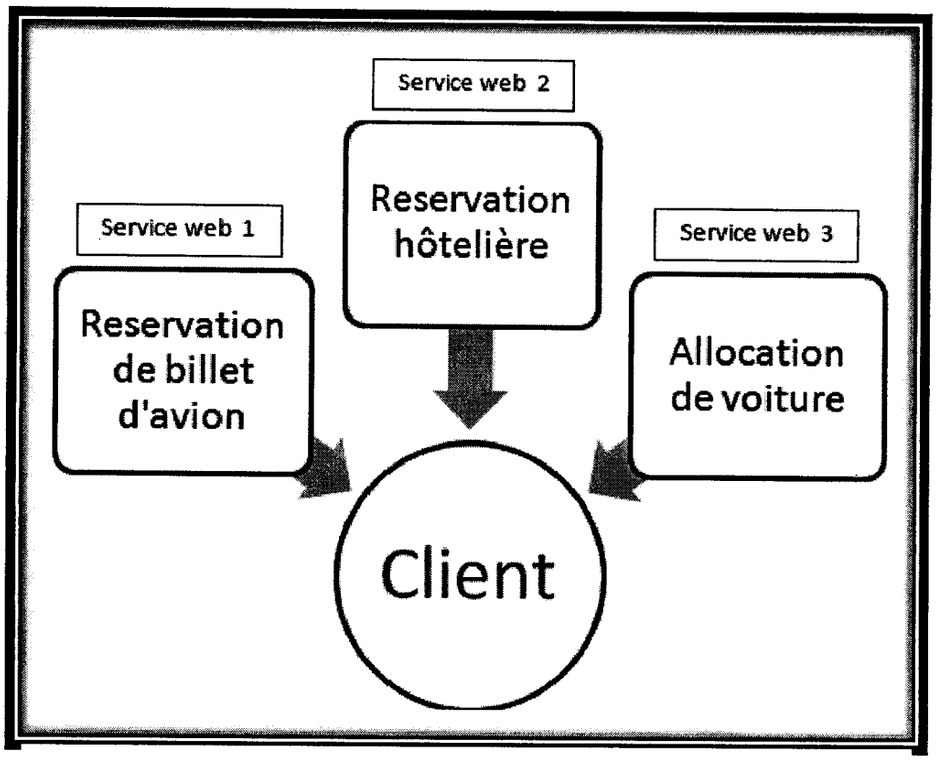


Figure II.1 Exemple de motivation

Et chaque service est proposé par une compagnie de telle sort à que chaque service doit respecter ou vérifier un ensemble de contraintes :

* cout : se critère représente le cout du service

Cout(Sol) = cout(S1) + cout(S2) + cout(S3)

* Réputation : chaque agence à sa propre réputation dans le marché

Rep(Sol) = rep(S1) + rep(S2) + rep(S3)

* Disponibilité : se critère vise a vérifié la disponibilité du service

Disp(Sol) = $\log_2$(S1) + $\log_2$(S2) + $\log_2$(S3)

* Fiabilité : se critère calcul le degré d'erreur

Fiab(Sol) = $\log_2$(S1) + $\log_2$(S2) + $\log_2$(S3)

* Temps d'exécution : se critère calcul le temps nécessaire pour exécute et accomplir le service.

Temps(Sol) = temps(S1) + temps(S2) + temps(S3)

Donc on dispose d'un service Sol qui répond à une requête complexe:

Sol = (S_1 , S_2 , S_3)

Chaque compagnie doit préciser les valeurs des cinq contraintes de tel sort à respecter une fonction globale qui représente une exigence pour le client et qui est la suivante :

Fg(Sol) = (Cout(Sol) , Rep(Sol) , Disp(Sol) , Fiab(Sol) , Temps(Sol))

Dans notre cas nous avons trois classes de services et pour chaque classe nous avons sept instance qui représente les compagnies qui propose se type de service

compagnie	Service web S1
1	[50 , 0.6 , 0.7 , 0.2 , 8]
2	[55 , 0.8 , 0.8 , 0.15 , 7]
3	[60 , 0.7 , 0.9 , 0.26 , 10]
4	[45 , 0.77 , 0.6 , 0.3 , 9]
5	[90 , 0.9 , 0.85 , 0.18 , 7]
6	[68 , 0.74 , 0.4 , 0.08 , 6]
7	[58 , 0.6 , 0.6 , 0.16 , 9]

Tableau II.1 Service 1

compagnie	Service web S2
1	[30 , 0.7 , 0.56 , 0.38 , 13]
2	[25, 0.4 , 0.6 , 0.17 , 9]
3	[34 , 0.8 , 0.78 , 0.36 , 9]
4	[28 , 0.77 , 0.9 , 0.14 , 15]
5	[40 , 0.9 , 0.7 , 0.09 , 16]
6	[33 , 0.65 , 0.75 , 0.29 , 7]
7	[25 , 0.55 , 0.77 , 0.17 , 15]

Tableau III.2 Service 2

compagnie	Service web S3
1	[15 , 0.7 , 0.8 , 0.1 , 22]
2	[20 , 0.67 , 0.55 , 0.03 , 9]
3	[22 , 0.8 , 0.66 , 0.2 , 10]
4	[30 , 0.49 , 0.9 , 0.06 , 8]
5	[24 , 0.7 , 0.8 , 0.11 , 10]
6	[19 , 0.77 , 0.8 , 0.25 , 14]
7	[18 , 0.8 , 0.59 , 0.1 , 10]

Tableau II.3 Service 3

Pour chaque client en doit trouver la meilleur combinaison qui respecte les critères spécifier par se dernier.

Exemple : si on prend la combinaison Sol (1 , 4 , 6) alors :

$$\begin{cases} \text{Cout(Sol)} = 50 + 28 + 19 = 97 \\ \text{Rep(Sol)} = 0.6 + 0.77 + 0.77 = 2.14 \\ \text{Disp(Sol)} = \log_2(0.7) + \log_2(0.9) + \log_2(0.8) = -0.29 \\ \text{Fiab(Sol)} = \log_2(0.2) + \log_2(0.14) + \log_2(0.25) = -2.15 \\ \text{Temps(Sol)} = 8 + 15 + 14 = 37 \end{cases}$$

Et de cette façon la fonction globale sera la suivante :

Fg(Sol) = (97 , 2.14 , -0.29 , -2.15 , 37)

II.2 concepts d'optimisation

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.

Donc un problème d'optimisation se définit comme la recherche du minimum ou du maximum (l'optimum) d'une fonction donnée.

$$s^* = \min(f(s) \setminus \mathrm{s} \in S)$$

Pour l'optimisation il existe deux approches :

- l'optimisation mono objectif ;

- l'optimisation multi objectif **[Collette et al. 2002],** ou il s'agit d'optimiser simultanément plusieurs objectifs contradictoires ;

II.3 L'optimisation mono objectif

Elle est utilisée pour leur simplicité de mise en œuvre et leur grande généricité. En effet, les objectifs du problème d'optimisation sont transformés en une seule fonction objective. Dans ce cas, le décideur est supposé quantifier a priori l'importance de chaque critère afin de construire une fonction unique. Le processus d'optimisation mono-objectif est ensuite lancé afin de déterminer la solution « optimale ». **[MOUELHI Ouael, 2010]**
L'exemple le plus répondue de se type d'optimisation est la programmation linéaire.

II.3.1 la programmation linéaire :

La programmation linéaire est un outil très puissant de la recherche opérationnelle. C'est un outil générique qui peut résoudre un grand nombre de problèmes. En effet, une fois un problème modélisé sous la forme d'équations linéaires, des méthodes assurent la résolution du problème de manière exacte. On distingue dans la programmation linéaire, la programmation linéaire en nombres réels, pour laquelle les variables des équations sont dans IR+ et la programmation en nombres entiers, pour laquelle les variables sont dans IN. Bien entendu, il est possible d'avoir les deux en même temps.

Une des méthodes les plus connues pour résoudre des programmes linéaires en nombre réels est la méthode du Simplex.

La programmation linéaire permet la résolution d'un programme linéaire. Un programme linéaire est un système d'équations ou d'inéquations appelées "contraintes" qui sont linéaires (c'est-à-dire que les variables ne sont pas élevées au carré, ne servent pas d'exposant, ne sont pas multipliées entre elles...). Et à partir de ces contraintes, on doit optimiser une fonction également linéaire appelée objectif.

Mais avant de cité deux méthodes utiliser dans le domaine de la programmation linéaire il faut savoir qu'il ya des logiciel qui sont utiliser dans se domaine tel que LP Solver comme le montre la figure suivante :

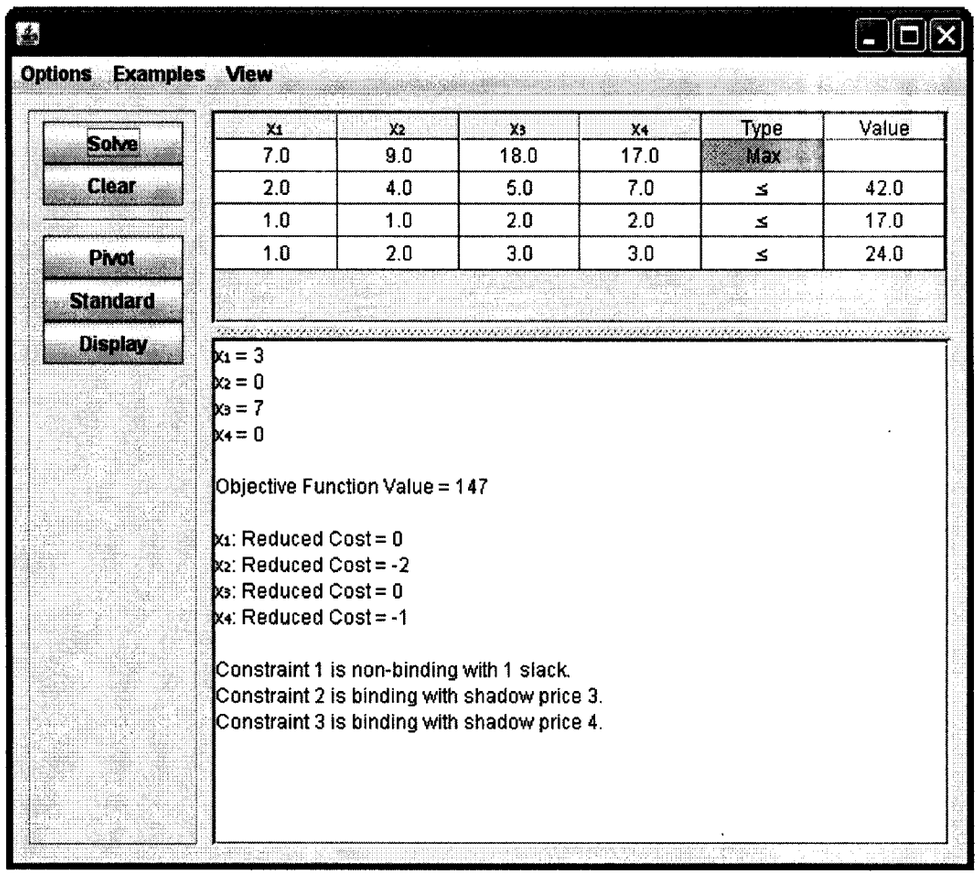


Figure II.2 LP Solver

a. Simplex

La méthode du simplexe permet la recherche d'un optimum économique. Il s'agit par exemple, dans le cas d'un programme de production, de déterminer quelles seront les quantités à fabriquer pour optimiser le résultat, et ce en tenant compte des contraintes de production.

Par exemple une entreprise fabrique quatre produits. La fabrication de chaque produit nécessite une certaine quantité de ressources. Les ressources consommées, les stocks des ressources et les bénéfices des produits sont récapitulés dans le tableau suivant :

	Produit 1	Produit 2	Produit 3	Produit 4	Stock
Ressource A	2	4	5	7	42
Ressource B	1	1	2	2	17
Ressource C	1	2	3	3	24
Bénéfice	7	9	18	17	

Tableau II.4 Exemple de simplex

On souhaite établir un plan de production de façon à maximiser le chiffre d'affaires. Appelant x1; x2; x3; x4 les quantités respectives de produits 1, 2, 3, 4, le problème admet la modélisation suivante :

Maximiser $Z = 7x1 + 9x2 + 18x3 + 17x4$

$$\begin{cases} 2x1 + 4x2 + 5x3 + 7x4 <= 42 \\ x1 + x2 + 2x3 + 2x4 <= 17 \\ x1 + 2x2 + 3x3 + 3x4 <= 24 \end{cases} \qquad (x1 ; x2 ; x3 ; x4 >= 0)$$

Apre la formalisation du problème on doit suivre un ensemble d'étapes pour arriver enfin à la solution optimal qui est dans notre cas la suivante :

$$\begin{cases} X_5 = 1 + x_6 - x_2 + x_7 - 2x_4 \\ X_1 = 3 - 3x_6 + x_2 + 2x_7 \\ X_3 = 7 + x_6 - x_2 - x_7 - x_4 \end{cases}$$

$$Z = 147 - 3x_6 - 2x_2 - 4x_7 - x_4$$

La solution basique associée $x_6 = x_2 = x_7 = x_4 = 0$, $x_5 = 1$, $x_1 = 3$, $x_3 = 7$, correspond au sommet (3; 0; 7; 0) du polyèdre des contraintes. Elle définit le plan de production suivant : on fabrique 3 unités du produit 1 et 7 unités de produit 3. Il ne nous reste qu'une unité de la ressource A. Le bénéfice est Z = 147. De plus, tous les coefficients de la dernière ligne sont négatifs, donc on ne peut pas se déplacer vers un sommet voisin en augmentant la fonction du profit. Nous avons alors trouvée la solution optimale du problème.

b. programmation en nombre entier

La programmation en nombres entiers concerne les programmes d'optimisation sous contraintes pour lesquels les variables doivent prendre des valeurs entières.
Les techniques (continues) utilisées en programmation linéaire (ou non linéaire) sont mathématiques et basées sur les notions de distance, de dérivée ou de gradient.
Elles ne peuvent généralement pas être appliquées au cas des programmes en nombres entiers (PNE), qui sont par définition discontinus.
Les méthodes d'arrondissent généralement peu efficaces pour les PNE.

Exemple : Le problème du sac à dos consiste à choisir des objets (poids, valeur) de façon à maximiser la somme des valeurs de ces objets sans que le poids total de ces objets ne dépasse la capacité du sac à dos.

Il existe deux manières de classer les algorithmes de résolution pour les PNE ; le premier classement :

- A caractère général : peut résoudre n'importe quel PNE, mais est potentiellement inefficace.

- A caractère spécifique : étudié pour un type de problème particulier, mais potentiellement efficace.

Le deuxième classement possible :

- Optimal : l'algorithme est certain de trouver la solution optimale.
- Heuristique : l'algorithme trouve une bonne solution, que l'on espère proche ou égale) à l'optimum.

Il en découle donc quatre "catégories" à considérer.

II.4 L'optimisation multi objectifs

L'optimisation multi objectif est née du besoin en industrie de satisfaire plusieurs critères contradictoires, simultanément. Les bases de cette optimisation ont été posées par Pareto et Edgeworth au 19ème siècle. Ses théories trouvent leurs premiers applications en économie et ces dernières années dans les sciences pour l'ingénieur .

L'optimisation mono objectif consiste à maximiser (ou minimiser) une seule fonction objectif par rapport à un ensemble de paramètres. Or, la plupart des problèmes d'optimisation réels sont décrits à l'aide de plusieurs objectifs souvent contradictoires devant être optimisés simultanément. La solution optimale correspond alors aux meilleurs compromis possibles permettant de résoudre le problème. Il s'agit de l'optimisation multi objectifs, qui fournit aux décideurs un ensemble de solutions optimales.

II.5 Les méthodes d'optimisation multi objectifs

Les méthodes d'optimisation des problèmes multi objectifs sont très variées, certain des problèmes d'optimisation peuvent être résolus par des méthodes exactes simple, qui permettent de trouver à coup sur la (ou les) meilleure(s) solution(s) optimale(s). Mais la plupart des problèmes étudies en optimisation appartiennent à une classe qui rassemble des problèmes pour les quels on ne connait pas l'algorithme exacte rapide

dont la résolution exacte n'est pas possible et le problème de voyageur de commerce et la meilleure preuve.

Deux grandes classes de méthodes existent pour résoudre le problème d'optimisation multi objectifs : les algorithmes exacts et l'heuristique.

II.5.1 Algorithmes exacts

Ces méthodes permettent d'obtenir l'optimum global de manière exacte pour certains types de problèmes. Elles sont efficaces seulement lorsque le PMO à résoudre est bi-objectif et l'espace de recherche est de petite taille.
Parmi ces algorithmes ont trouvent :

**** méthodes SEP (séparation et évaluation progressive) :***

Ces méthodes sont connues sous le nom de Branch and Bround, ou recherche arborescente. Ce sont des méthodes exactes donnant l'optimum. Elles ne balayent pas tous l'espace de recherche pour trouver la meilleure solution mais elles utilisent des bornes pour évaluer les solutions en cours de construction. Les bornes utilisées sont soit supérieures ou inferieurs selon que le problème est une maximisation ou une minimisation.
Le principe de ces méthodes est de diviser l'espace des solutions en sous-ensembles de plus en plus petits réduisant ainsi le problème en sous-problèmes.

Elles représentent l'espace des solutions sous forme d'un arbre. Elles démarrent en considérant le problème et toute l'espace de solutions, ce qui correspond à la racine de l'arbre. Chaque branche de l'arbre, quant à elle, correspond à une décision (instanciation d'une variable, par exemple). A ce niveau, l'espace de recherche est divisé en 2 ou plusieurs sous régions. Chaque sous arbre correspond à un sous problème. Certaines branches de l'arbre ne sont pas générées grâce à l'évaluation. Cette dernière permet d'évaluer les solutions partielles (correspondant aux nœuds) et voir l'impact de la décision prise: Est-ce que la décision prise est bonne ? Doit-on continuer sur ce chemin ? Certains nœuds sont éliminés et leurs sous arbres ne sont pas générés. La fonction d'évaluation permet donc d'éliminer des branches de l'arbre

et localiser assez rapidement la solution optimale. Ces méthodes ont un temps d'exécution exponentiel mais elles présentent une amélioration si on les compare aux méthodes faisant une énumération totale de l'espace des solutions.

II.5.2 Heuristique

Afin d'améliorer le comportement d'un algorithme dans son exploration de l'espace des solutions d'un problème donné, le recours à une méthode heuristique (du verbe grec heuriskein, qui signifie « trouver ») permet de guider le processus dans sa recherche des solutions optimales.

Dans la littérature, les méthodes heuristiques sont réparties en deux classes :

Les algorithmes spécifiques à un problème donné, qui utilisent des connaissances du domaine, et les algorithmes généraux applicables à une grande variété de problèmes multi objectifs : les méta heuristiques.

Exemple :

Pour réaliser un mémoire, un étudiant doit faire quelques recherches bibliographiques. En plus des nombreuses sources d'informations classiques qu'il peut exploiter, il décide de se rendre dans une bibliothèque privée spécialisée dans son domaine. Le problème de cette bibliothèque est qu'elle ne propose aucun catalogue pour faciliter la recherche d'un livre. Il faut donc regarder directement dans les rayons pour effectuer une recherche, et pour se rendre compte de la pertinence d'un livre, il est nécessaire de le feuilleter. À la vue des centaines de livres dans la bibliothèque, notre étudiant ne peut pas se permettre de regarder chaque ouvrage pour emprunter celui qui lui convient le mieux.

Une telle manière de faire prendrait plusieurs jours. Utilisant le fait que les livres ont une disposition par rayon relativement cohérente, il va mettre en place un procédé de recherche lui permettant de trouver un ouvrage satisfaisant en un temps raisonnable. La stratégie de recherche consiste à piocher aléatoirement un livre et le feuilleter. Si l'ouvrage semble intéressant, il va regarder les ouvrages proches jusqu'au moment ou ils semblent ne plus convenir. À ce moment-là, il réitérera le processus en piochant au

hasard un autre livre, de préférence dans un rayon qu'il n'a pas encore exploré. Si au bout de plusieurs itérations il s'aperçoit que cette stratégie n'est pas efficace, alors il la modifiera. Lorsque notre étudiant souhaitera arrêter sa recherche, il empruntera le livre le plus pertinent parmi ceux qu'il a feuilletés. **[David MEIGNAN, 2008]**

Dans cet exemple, le problème auquel fait face l'étudiant est un problème d'optimisation pouvant s'exprimer sous la forme suivante : considérant les livres présents dans la bibliothèque, trouver le plus approprié au sujet de recherche de l'étudiant. Dans ce problème d'optimisation, chaque livre est une solution potentielle et la fonction objective à maximiser est la pertinence du livre.

Pour ce problème, l'étudiant ne dispose pas d'outils lui permettant de trouver en un temps raisonnable la meilleure solution. Il met donc en place une recherche de type heuristique. Le terme "heuristique" au sens originel qualifie "tous les outils intellectuels, tous les procédés et plus généralement toutes les démarches favorisant la découverte ou l'invention". Dans le cadre de la résolution de problèmes, les heuristiques sont définies comme étant des techniques de résolution incertaines dont l'usage n'est justifié que par le constat d'une efficacité antérieure dans des problèmes analogues. L'emploi de méthodes de recherche heuristiques se justifie par l'inexistence ou l'inefficacité de méthodes exactes, et la difficulté ou l'impossibilité d'explorer exhaustivement l'espace des solutions.

Dans l'exemple, l'heuristique consiste à choisir aléatoirement un livre puis à effectuer une recherche localement en examinant les livres proches. Ce processus est réitéré jusqu'à un critère d'arrêt.

L'heuristique présentée dans l'exemple est spécifique à la recherche d'un livre dans une bibliothèque. Si l'on dégage de cette heuristique un schéma général applicable à d'autres problèmes d'optimisation, alors ce schéma sera qualifie de méta heuristique. Le terme méta heuristique a été introduit pour distinguer les approches spécifiques à un problème des méthodes génériques applicables à différents problèmes (le méta heuristique).

a. Heuristique spécifique

Elles sont dédiées à la résolution d'un PMO (optimisation multi objectif) spécifique et ne fonctionnent que sur ce type de problèmes.

b. Meta heuristique

Elles sont fondées sur une idée générale. Et peuvent s'adapter à n'importe quel type de problème et donner de bons résultats. On trouve dans cette classe les algorithmes génétiques, la recherche Tabou, le recuit simulé,...

Recuit simulé

L'origine de la méthode du recuit simulé vient de l'analogie avec la métallurgie,où la méthode, pour atteindre des états de basse énergie d'un solide, consiste à monter la température du solide à des valeurs élevées, puis à le laisser refroidir lentement. Ce processus est appelé "recuit".

Le recuit simulé est une méthode issue de la mécanique statistique. L'application à l'optimisation combinatoire a été proposée par Kirkpatrick et al. 1983.

S.Kirkpatrick et ses collègues étaient des spécialistes de physique statistique (qui s'intéressaient aux configurations de base énergie de matériaux magnétiques désordonnés, regroupés sous le terme de verres de spin). **[Johann Dréo et al. 2003]**

En métallurgie, l'obtention d'un cristal parfait se fait grâce à la méthode du recuit. On porte un métal à une température suffisamment élevée pour qu'il soit à l'état liquide. Puis, à partir de cet état, on abaisse la température, et de ce fait les atomes se réorganisent en une nouvelle structure. Une même structure initiale peut donner différentes structures finales selon la façon dont on baisse la température.

Pour faire disparaître d'éventuels défauts dans la structure du cristal, on réchauffe doucement le métal, afin que les atomes aient plus de liberté de mouvement. En suffisamment, on donne assez d'énergie pour qu'ils sortent de l'optimum local.

En abaissant à nouveau la température régulièrement, ils pourront atteindre l'optimum global. C'est en s'inspirant de ce procédé que le recuit simulé a été développé. Il s'agit d'un algorithme stochastique permettant une détérioration de la valeur courante selon une certaine probabilité. Cette probabilité dépend de l'importance de la détérioration et du degré d'avancement de la recherche. Cette méthode n'exclut donc pas entièrement le cyclage , mais l'autorise selon une certaine probabilité. Une différence fondamentale apparaît: il n'y a plus ici la notion de température, elle deviendra simplement un paramètre de contrôle lorsqu'on simule un recuit (d'où le nom Recuit Simulé, ou Simulated annealing).

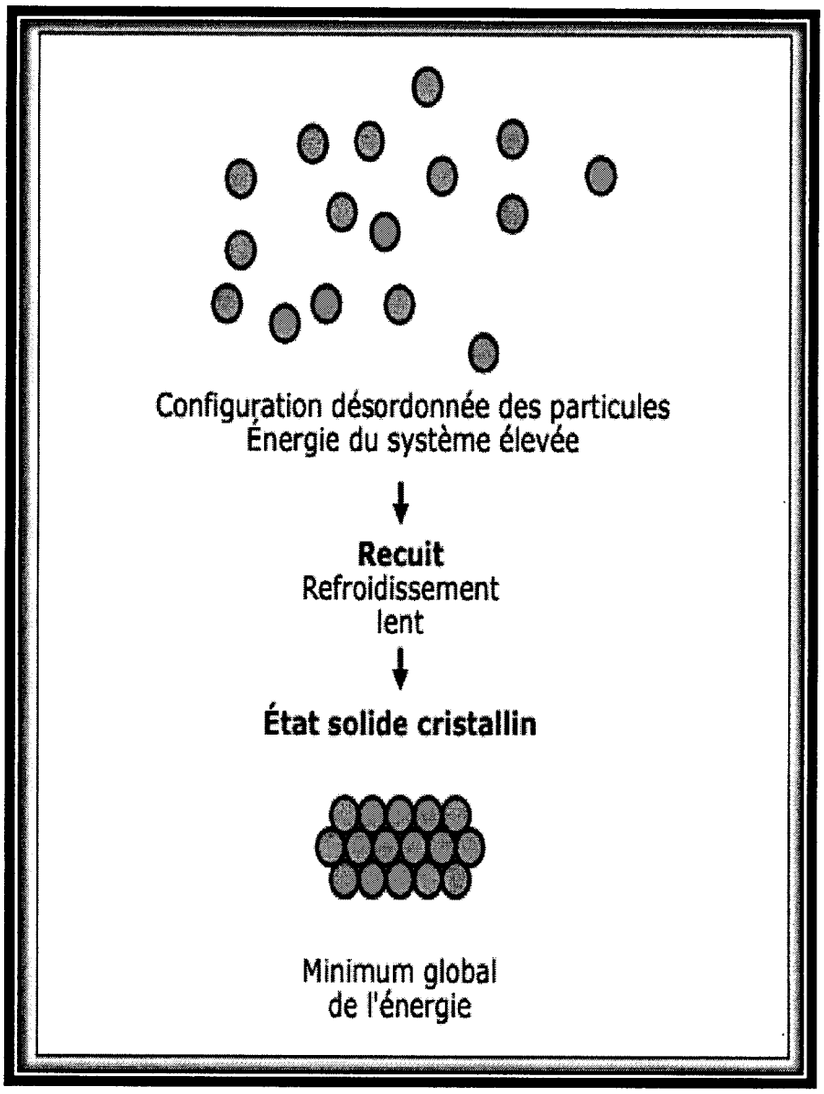


Figure II.3 Technique de recuit

Le principe du recuit simulé, présenté dans l'Algorithme suivante, consiste à démarre l' algorithme avec une température initiale élevée et à contrôler la décroissance de la température ,au sein de l'algorithme de Metropolis.

Plusieurs lois de décroissance de la température peuvent être utilisées en pratique.

Voici l'algorithme de recuit simulé :

1 **Définir** la fonction objectif (f)
2 **Choix** des mécanismes de perturbation d'une configuration ΔS
3 **Tirer** une configuration aléatoire E
4 **Calculer** l'énergie associée à cette configuration E
5 **Initialiser** la température (T_0)
6 **Tant** que condition d'arrêts pas satisfaite **faire**
  6.1 **Tant** que l'équilibre thermodynamique pas atteint **faire**
    6.1.1 **Tirer** une nouvelle configuration S'
    6.1.2 **Appliquer** la règle de metropolis
    6.1.3 **Si** f(S') < f(S)
      f_{min} = f(S')
      S_{opt} = S'
      Fin de Si
  Fin tant que
  6.2 **Décroître** de la température
Fin tant que
7 **Afficher** la solution optimale

En 1953, Metropolis avait proposé un algorithme itératif qui permet d'atteindre l'état d'équilibre thermodynamique d'un système simulé à une température T .Son principe consiste à itérer les deux étapes suivantes :

* évaluer la variation d'énergie associée à une transition élémentaire aléatoire de l'état courant i, d'énergie Ei, vers un nouvel état j, d'énergie Ej : $\Delta E_{ij} = E_j - E_i$;

* accepter la transition vers le nouvel état avec une probabilité Pij où

$$\begin{cases} P_{ij}(T) = 1 & \text{si} \quad \Delta E_{ij} \le 0 \\ P_{ij}(T) = \exp\left(-\dfrac{\Delta E}{T}\right) & \text{si} \quad \Delta E_{ij} > 0 \end{cases}$$

La méthode de recuit simulé, appliquée aux problèmes d'optimisation mono objectif, renvoi une solution unique.

Recherche Tabou

La méthode de recherche Tabou, ou simplement méthode Tabou, a été formalisée en 1986 par F. Glover. Sa principale particularité tient dans la mise en ouvre de mécanismes inspirés de la mémoire humaine. La méthode tabou prend, sur ce plan, le contre-pied du recuit simulé, totalement dépourvu de mémoire, et donc incapable de tirer les leçons du passé. La méthode Tabou est une méthode itérative qui, partant d'une solution initiale, tente d'atteindre la solution optimale. Elle exécute à chaque pas des mouvements dans un graphe d'espace d'états. En acceptant de détériorer la valeur de la solution courante, elle permet de s'éloigner d'un optimum local. Le risque de cycler est évité en utilisant une liste tabou qui garde en mémoire les derniers mouvements et ceci pendant un nombre limité d'itérations.

Le principe est simple : comme le recuit simulé, Tabou fonctionne avec une seule *configuration courante* à la fois (au départ, une solution quelconque), qui est actualisée au cours des *itérations* successives. À chaque itération, le mécanisme de passage d.une configuration, soit *s*, à la suivante, soit *t*, comporte deux étapes :

* On construit l'ensemble des *voisins* de *s*, c'est-à- dire l'ensemble des configurations accessibles en un seul *mouvement* élémentaire à partir de *s* (si cet ensemble est trop vaste, on en extrait aléatoirement un sous-ensemble de taille fixée) ; soit $V(s)$ l'ensemble (ou le sous-ensemble) de ces voisins ;

* On évalue la fonction objectif f du problème en chacune des configurations appartenant à $V(s)$. La configuration *t*, qui succède à *s* dans la suite de solutions construite par Tabou, est la configuration de $V(s)$ en laquelle f prend la valeur minimale. Notons que cette configuration *t* est adoptée même si elle est moins bonne que *s* (si $f(t) > f(s)$) : c.est grâce à cette particularité que Tabou permet d'éviter le piégeage dans les minimums locaux de f.

Telle quelle, la procédure précédente est inopérante, car il y a un risque important de retourner à une configuration déjà retenue lors d.une itération précédente, ce qui engendre un cycle. Pour éviter ce phénomène, on tient à jour et on exploite, à chaque itération, une *liste tabou* de mouvements interdits : cette liste circulaire, qui a donné son nom à la Méthode, contient *m* mouvements ($t \rightarrow s$), qui sont les inverses des *m*

derniers mouvements ($s \rightarrow t$) effectués. L'organigramme de cet algorithme dit « Tabou simple » est représenté sur la figure suivante :

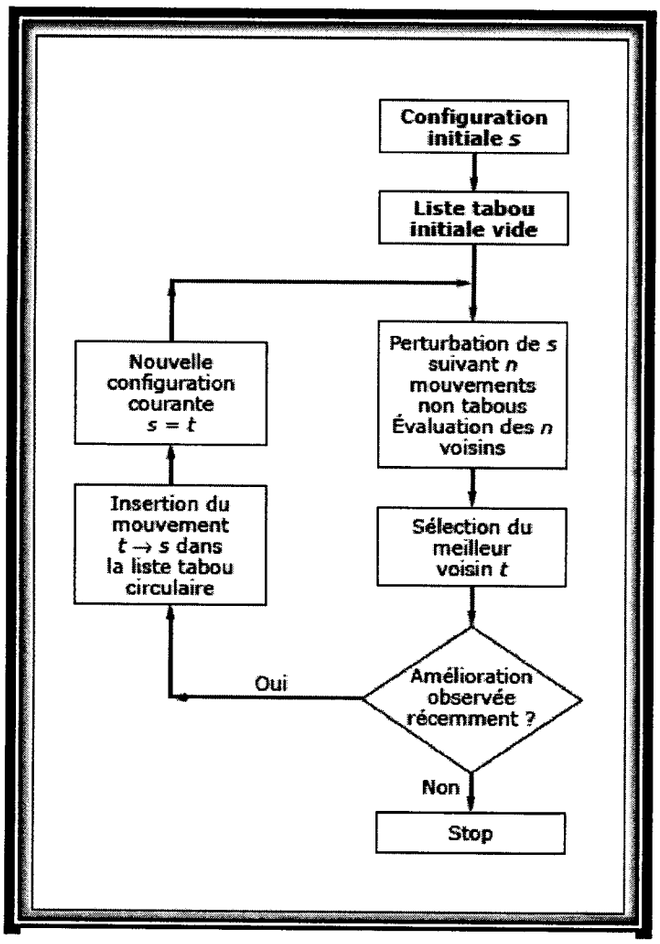


Figure II.4 Organigramme de l'algorithme Tabou

Algorithme Génétique

Les algorithmes génétiques (AG) sont des techniques de recherche inspirées par l'évolution biologique des espèces. Ce type d'algorithme a été proposé par Holland. Il examine à chaque pas une population de solutions. Chaque population (enfants) est dérivée de celle qui la précède (parents) en combinant ses meilleurs éléments et en mettant de côté les moins bons. Des perturbations aléatoires sont

appliquées aux enfants de façon à assurer une grande diversité de solutions dans une population: ce sont les mutations.

Une description de l'algorithme génétique est présentée si dessue

1 **Initialisation** de P(t)
2 **Evaluer** chaque individu de P(t)
3 **Tant** que le critère d'arrêts n'est pas satisfait **faire**
  3.1 t= t+1
  3.2 **Sélectionner** P(t+1) de P(t)
  3.3 **Croisement** P(t+1)
  3.4 **Muter** P(t+1)
  3.5 **Evaluer** P(t+1)
Fin tant que
4 **Afficher** la meilleur état rencontré au cour de la recherche

Le principe d'un AG se décrit simplement, dans le cas de la minimisation d'une fonction f. Un ensemble de N points, qui peuvent être choisis au hasard, constitue la population initiale ; chaque individu x de la population possède une certaine compétence, qui mesure son degré d'adaptation à l'objectif visé .Un AG consiste à faire évoluer progressivement, par générations successives, la composition de cette population, en maintenant sa taille constante. D'une génération à la suivante, la compétence de la population doit globalement s'améliorer ; un tel résultat est obtenu en mimant les deux principaux mécanismes qui régissent l'évolution des êtres vivants: la sélection naturelle (qui détermine quels membres d'une population survivent et se reproduisent) et la reproduction (qui assure le brassage et la recombinaison des gènes parentaux, pour former des descendants aux potentialités nouvelles).

En pratique, chaque individu est codé par une chaine de caractères de longueur donnée (de même qu'un chromosome est formé d'une chaine de gènes). Le passage d'une génération à la suivante se déroule en deux phases: une phase de reproduction et une phase de remplacement. La phase de reproduction consiste à appliquer des opérateurs, dits génétiques, sur les individus de la population courante, pour engendrer de nouveaux individus ; les opérateurs les plus utilisés sont le croisement, qui produit deux descendants à partir de deux parents, et la mutation, qui produit un

nouvel individu à partir d'un seul individu. Les individus de la population prenant part à la reproduction sont préalablement sélectionnés, en respectant le principe suivant : plus un individu est compétent, plus sa probabilité de sélection est élevée.
La phase de remplacement consiste ensuite à choisir les membres de la nouvelle génération : on peut, par exemple, remplacer les plus ′ mauvais individus (au sens de la fonction objectif) de la population courante par les meilleurs individus produits (en nombre égal). L'algorithme est interrompu après un nombre donné de générations.

II.6 Le front de Pareto

II.6.1 définition

La notion de front de Pareto est liée à celle d'optimum parétien, qui tirent toutes deux leurs noms des travaux de Vilfredo Pareto (1848-1923), économiste italien. On lui doit notamment le diagramme de Pareto, qui permet de représenter par ordre d'importance les causes d'un phénomène.
Le front de Pareto est un outil d'aide aux choix de solutions en fonction de plusieurs critères que l'on cherche à minimiser ou maximiser. Il se présente sous la forme d'une courbe sur laquelle, on ne peut voir quelle solution est la meilleure.
Lorsqu'une solution est associée à plusieurs objectifs que l'on cherche à minimiser, on recherche un ensemble de solutions non dominées (le « front de Pareto »). C'est à dire un ensemble pour lequel on ne peut voir quelle solution est la plus adaptée, toutes les solutions étant meilleures que les autres sur au moins un objectif.

II.6.2 Domination et front de Pareto

Soit u = (u1, u2, . . . , um) et v = (v1, v2, . . . , vm) deux vecteurs de décision. v domine u ($u \prec v$) pour un problème de minimisation si et seulement si

$$\begin{cases} \forall i \in 1, \ldots, n \; fi(v) \leq fi(u) \\ \exists i \in 1, \ldots, n \; fi(v) < fi(u) \end{cases}$$

Le front de Pareto FP* est l'ensemble des vecteurs de décision qui ne sont pas dominés.

$FP* = \{x \in X \mid @x0 \in X \; x \prec x0\}$

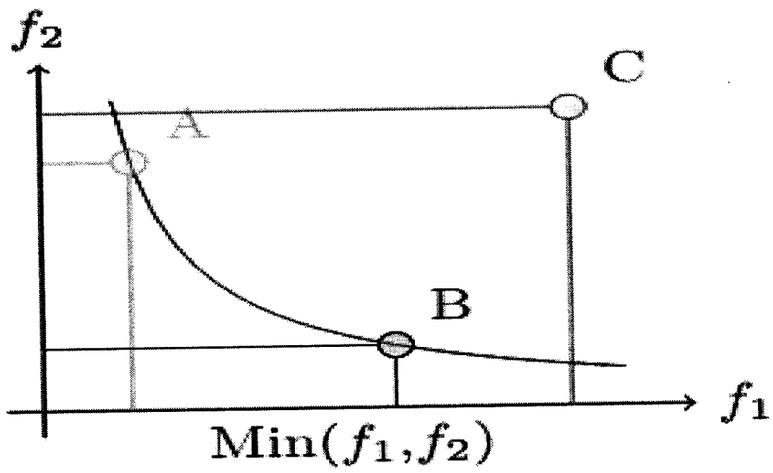


Figure II.5 Le front de Pareto

Le front de Pareto est l'ensemble des solutions de compromis. Sur la figure précédente, les points A et B sont deux points du front de Pareto : A ne domine pas B, B ne domine pas A, mais tous les deux dominent le point C. Le but de l'optimisation multi objectif est de déterminer le front de Pareto pour un problème donné.

CHAPITRE III : CONCEPTION ET IMPLÉMENTATION DU PROTOTYPE

III.1 Introduction

se chapitre est reserver pour l'application où on va resoudre le probleme d'optimisation multi objectif des services web a l'aide de l'algorithme de recuit simulé.

Le principe de la méthode de recuit simulé est simple. A partir d'une solution courante S_c , une solution voisine S_n est générée. Si cette dernière améliore la fonction objective (f), elle devient la solution courante. Sinon, S_n n'est retenu que selon la probabilité donnée par la formule suivante : $P(S_n, T) = \exp\left[\frac{f(S_n) - f(S_c)}{T}\right]$

Le processus de l'algorithme de recuit simulé est comme le suivant :

I=0 ; T= Température initiale ; S_c = Solution initial

Répéter

Répéter

Générer une autre solution S_n à partir de S_c

Si $f(S_c) < f(S_n)$ alors $S_c = S_n$

Sinon r = nombre aléatoire compris entre [0,1]

$P(S_n, T) = \exp\left[\frac{f(S_n) - f(S_c)}{T}\right]$

Si r < p alors $S_c = S_n$

Finsi

Jusqu'à un nombre d'itération prédéfini

Diminuer la température de T

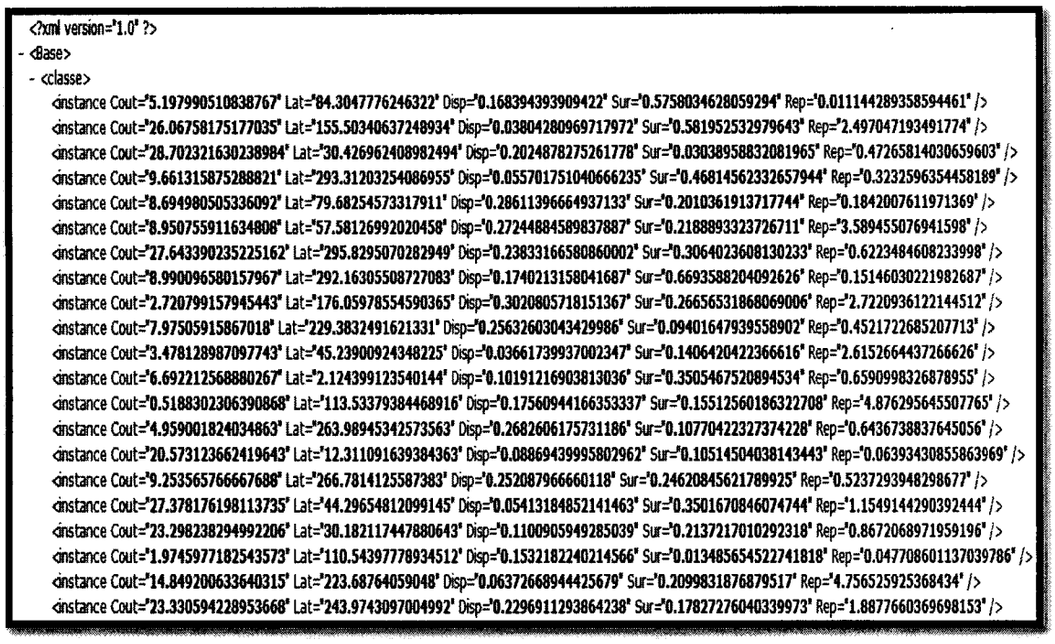
I=i+1

Jusqu'à critère d'arrêt

Fin

III.1.1 Présentation de la base et de la requête

Notre base (corpus) est sous forme de fichier XML, elle contient 10 services et chaque service et proposer par 40 compagnie de tel sorte a que chaque une de ces services est caractériser par cinq critères (Cout, Latence, Disponibilité, Surcharge, Réputation)



```
<?xml version="1.0" ?>
- <Base>
  - <classe>
      <instance Cout="5.197990510838767" Lat="84.3047776246322" Disp="0.168394393909422" Sur="0.5758034628059294" Rep="0.011144289358594461" />
      <instance Cout="26.067581751770 35" Lat="155.50340637248934" Disp="0.03804280969717972" Sur="0.581952532979643" Rep="2.497047193491774" />
      <instance Cout="28.702321630238984" Lat="30.426962408982494" Disp="0.2024878275261778" Sur="0.030389588320819 65" Rep="0.4726581403065 9603" />
      <instance Cout="9.661315875288821" Lat="293.31203254086955" Disp="0.055701751040666235" Sur="0.46814562332657944" Rep="0.3232596354458189" />
      <instance Cout="8.694980505336092" Lat="79.68254573317911" Disp="0.28611396664937133" Sur="0.2010361913717744" Rep="0.1842007611971369" />
      <instance Cout="8.950755911634808" Lat="57.58126992020458" Disp="0.27244884589837887" Sur="0.2188893323726711" Rep="3.589455076941598" />
      <instance Cout="27.643390235225162" Lat="295.8295070282949" Disp="0.23833166580860002" Sur="0.3064023608130233" Rep="0.6223484608233998" />
      <instance Cout="8.990096580157967" Lat="292.16305508727083" Disp="0.1740213158041687" Sur="0.6693588204092626" Rep="0.15146030221982687" />
      <instance Cout="2.720799157945443" Lat="176.059785545903 65" Disp="0.3020805718151367" Sur="0.266565318680690 06" Rep="2.7220936122144512" />
      <instance Cout="7.97505915867018" Lat="229.3832491621331" Disp="0.25632603043429986" Sur="0.09401647939558902" Rep="0.4521722685207713" />
      <instance Cout="3.478128987097743" Lat="45.23900924348225" Disp="0.03661739937002347" Sur="0.1406420422366616" Rep="2.6152664437266626" />
      <instance Cout="6.692212568880267" Lat="2.124399123540144" Disp="0.10191216903813036" Sur="0.3505467520894534" Rep="0.6590998326878955" />
      <instance Cout="0.5188302306390868" Lat="113.53379384468916" Disp="0.17560944166353337" Sur="0.1551256018632270 8" Rep="4.876295645507765" />
      <instance Cout="4.959001824034863" Lat="263.98945342573563" Disp="0.2682606175731186" Sur="0.107704223273742 28" Rep="0.6436738837645056" />
      <instance Cout="20.573123662419643" Lat="12.311091639384363" Disp="0.08869439995802962" Sur="0.1051450403814344 3" Rep="0.06393430855863969" />
      <instance Cout="9.253565766667688" Lat="266.7814125587383" Disp="0.252087966660118" Sur="0.24620845621789925" Rep="0.5237293948298677" />
      <instance Cout="27.378176198113735" Lat="44.29654812099145" Disp="0.05413184852141463" Sur="0.3501670846074744" Rep="1.1549144290392444" />
      <instance Cout="23.298238294992206" Lat="30.182117447880643" Disp="0.1100905949285039" Sur="0.2137217010292318" Rep="0.8672068971959196" />
      <instance Cout="1.9745977182543573" Lat="110.54397778934512" Disp="0.15321822402145 66" Sur="0.013485654522741818" Rep="0.04770860113703978 6" />
      <instance Cout="14.849200633640315" Lat="223.68764059048" Disp="0.06372668944425679" Sur="0.2099831876879517" Rep="4.756525925368434" />
      <instance Cout="23.330594228953668" Lat="243.9743097004992" Disp="0.2296911293864238" Sur="0.178272760403399 73" Rep="1.8877660369698153" />
```

Figure III.1 La Base

Les donnes du fichier peuvent êtres générer aléatoirement de tel façon a que : Couts ∈ [0 - 30] ; Latence ∈ [0 - 300] ; Disponibilité ∈ [O.7 - 1] ; Surcharge ∈ [0.5 - 1] ; Réputation ∈ [0 - 5].

Et pour la requête du client elle se compose de cinq valeurs souhaite pour les cinq critères.

III.2 Conception

En a réalisé notre conception avec UML (Unified Modeling Language)

III.2.1 Diagramme de cas d'utilisation

Le diagramme de classe concernons l'utilisateur est présenter dans la figure suivante :

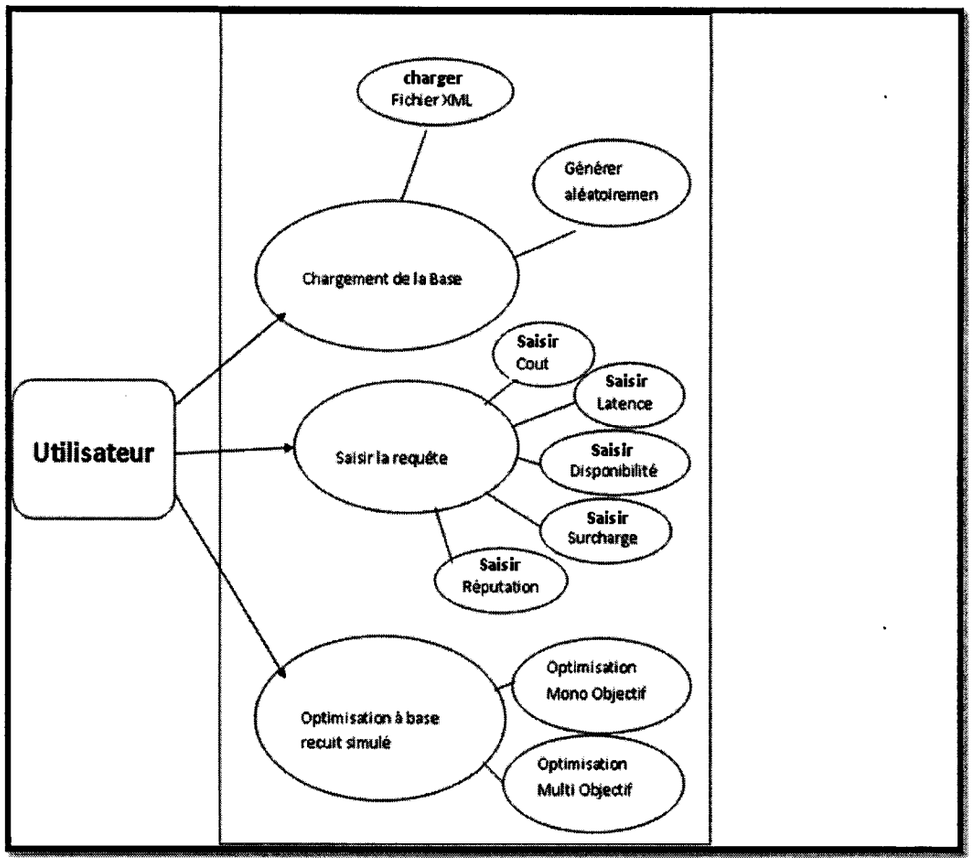


Figure III.2 Diagramme de cas d'utilisation

Pour la fonction mono objectif

$$F(x) = \sum_{Q_i \in Neg} W_i \frac{Q_i^{\max} + Q_i(x)}{Q_i^{\max} - Q_i^{\min}} + \sum_{Q_i \in Pos} W_i \frac{Q_i(x) - Q_i^{\min}}{Q_i^{\max} - Q_i^{\min}}$$

Et pour les fonctions multi objectifs

$$Cout_i(x) = \sum_{j=1}^{n} Cout_i(op_j)$$

$$Lat_i(x) = \sum_{j=1}^{n} Lat_i(op_j)$$

$$Disp_i(x) = \ln \prod_{i=1}^{n} Disp_i(op_j)$$

$$Sur_i(x) = \ln \prod_{j=1}^{n} Sur_i(op_j)$$

$$\mathrm{Re}\, p_i(x) = \sum_{j=1}^{n} \mathrm{Re}\, p_i(op_j)$$

En doit maximiser la disponibilité et la réputation d'une part et minimiser le cout, latence et la surcharge.

III.2.2 Diagramme de classe

Pour le diagramme de classe il est présenté dans la figure suivante :

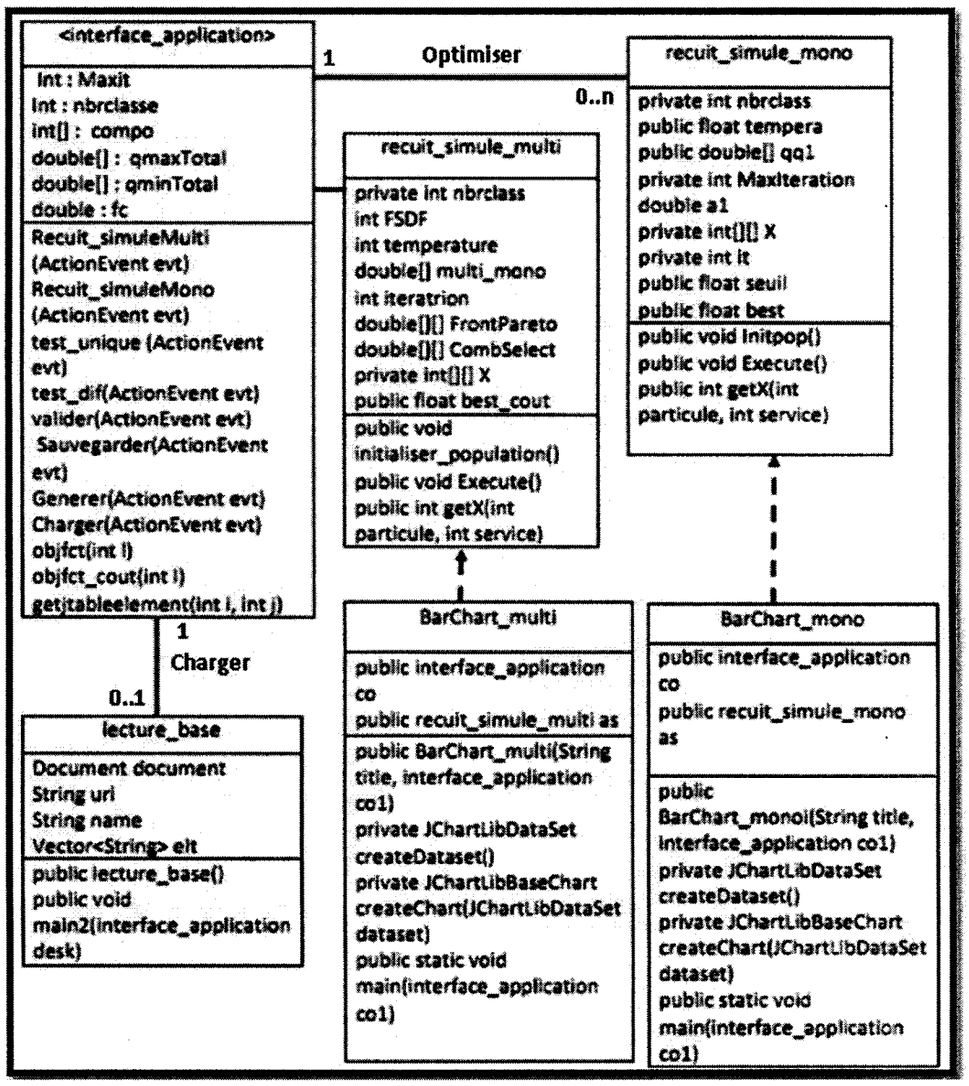


Figure III.3 Diagramme de classe

III.3 Implémentation

Les figures suivantes montres le mode de fonctionnement de notre application

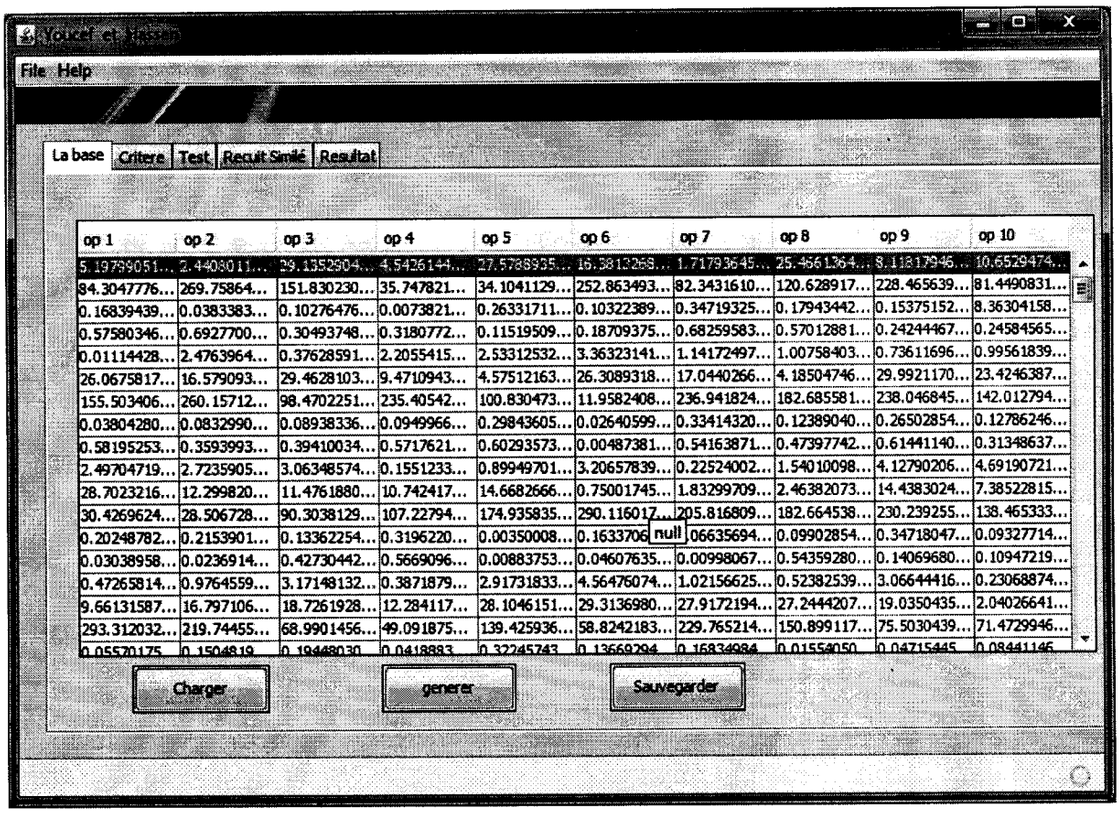


Figure III.4 Chargement de la base

Les cinq premières lignes représentent respectivement le cout, latence, disponibilité, surcharge et la réputation de la première compagnie et les cinq suivants sels de la deuxième compagnie et un de se suite jusqu'à la compagnie 40

Vous pouvez charger une base existant comme vous pouvez générer une nouvelle base aléatoirement et la sauvegarder

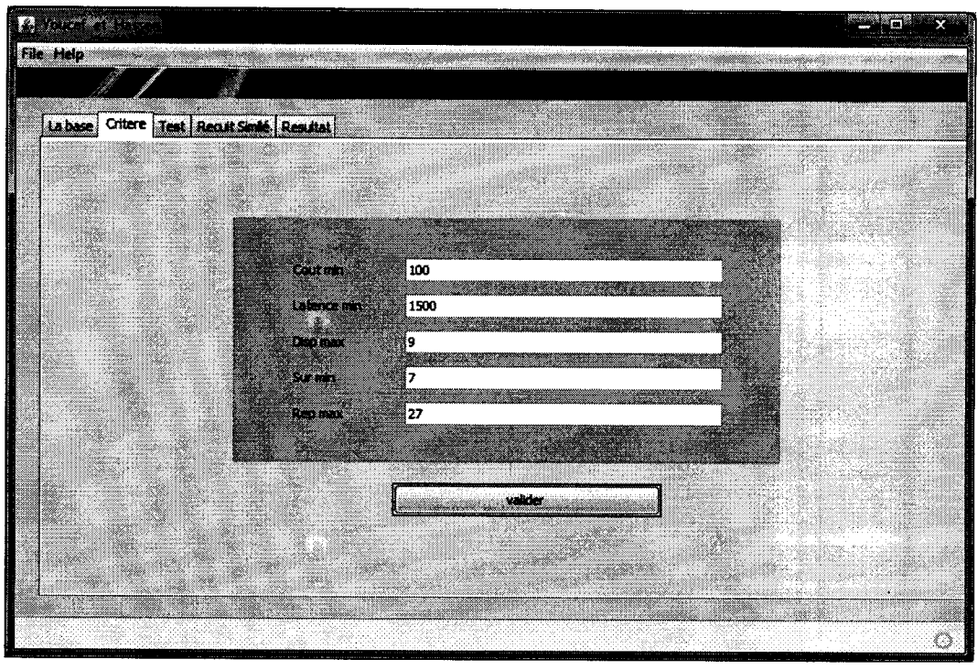


Figure III.5 Requête

Dans cette fenêtre l'utilisateur va saisir les valeurs des cinq critères en suite valider son choix

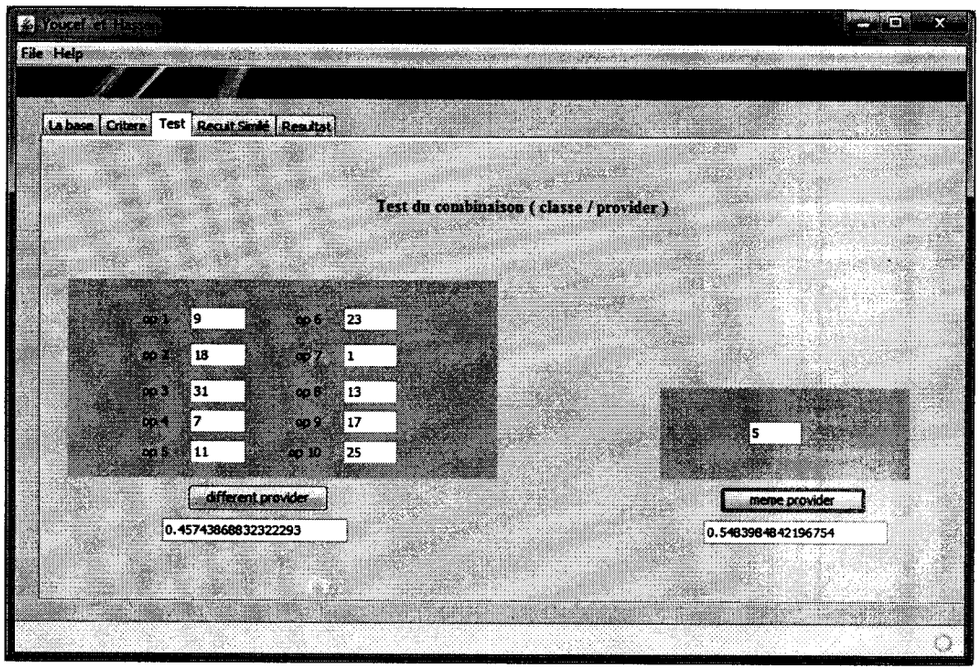


Figure III.6 Test

Dans cette fenêtre vous pouvez tester ou vérifier le bon fonctionnement d'application car vous pouvez choisir un ensemble de compagnie (ou la même compagnie) et ensuite calculer la fonction objectif

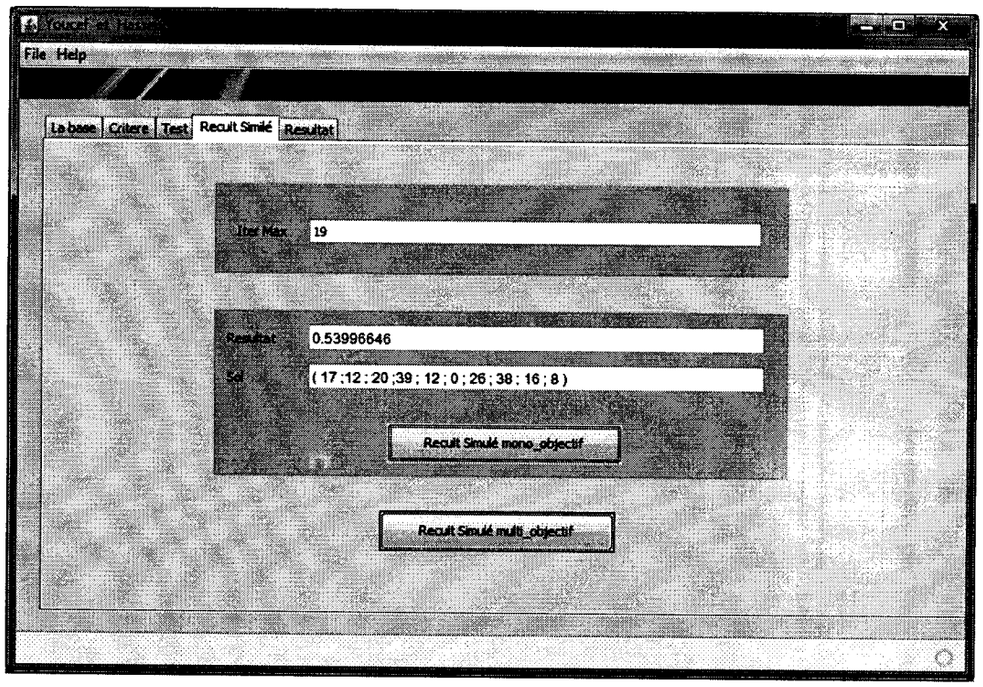


Figure III.7 Optimisation

L'utilisateur choisie le nombre d'itération et sélectionne la solution ou les solutions optimal a base de recuit simule pour l'optimisation mono objectif et l'optimisation multi objectif

III.3.1 Résultats

La première résultat c'est sel du score obtenu avec l'optimisation mono objectif car en peut avoir la valeur retenu ainsi que les compagnies qui en était choisi pour atteindre ce résultat comme le montre la figure suivante

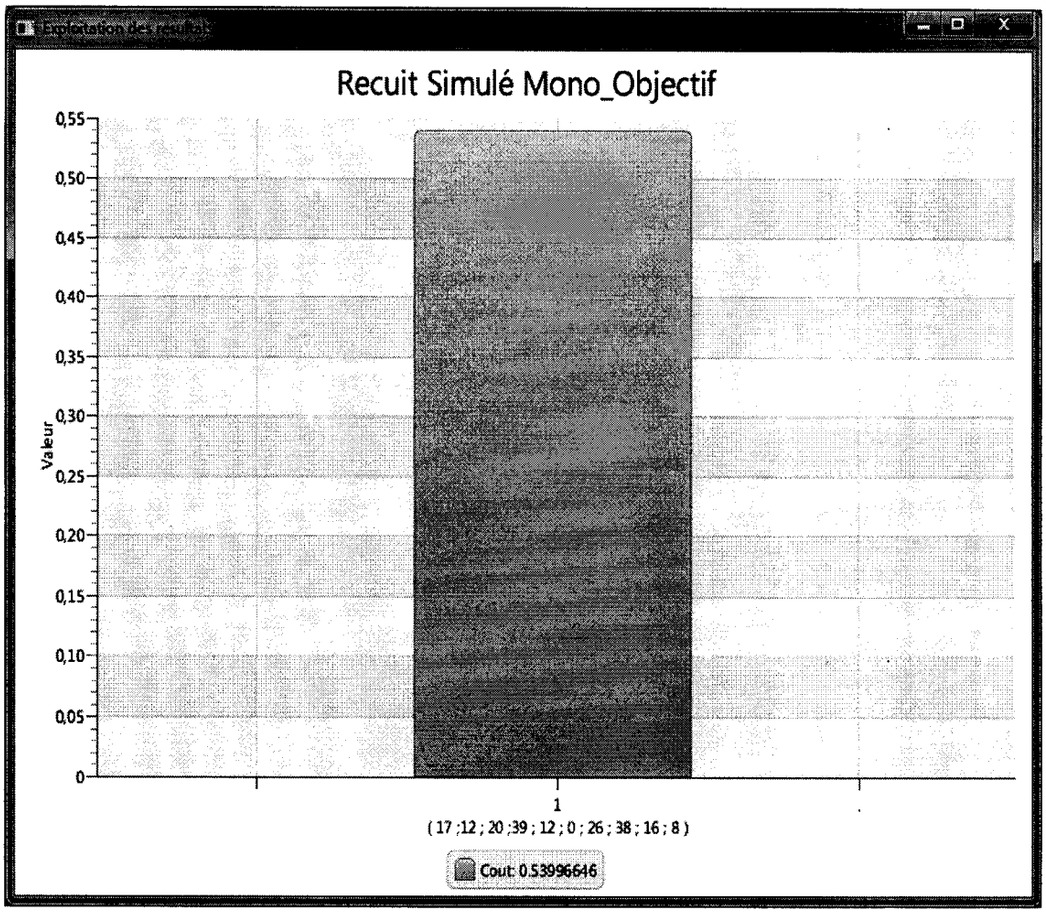
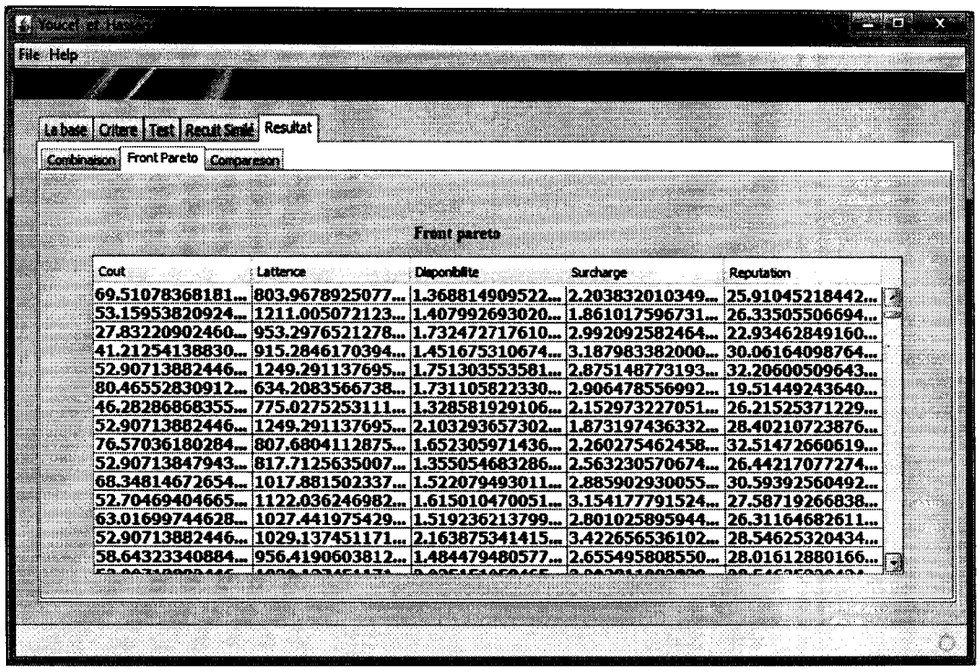


Figure III.8 Optimisation mono objectif

Les autres résultats concernent l'optimisation multi objectif car vous pouvez avoir un ensemble d'information sur le nombre d'itération, le nombre des combinaisons sélectionner et le nombre de combinaisons tirer dans le premier front Pareto comme dans la figure suivante :



Cout	Lattence	Disponibilite	Surcharge	Reputation
69.51078368181...	803.9678925077...	1.368814909522...	2.203832010349...	25.91045218442...
53.15953820924...	1211.005072123...	1.407992693020...	1.861017596731...	26.33505506694...
27.83220902460...	953.2976521278...	1.732472717610...	2.992092582464...	22.93462849160...
41.21254138830...	915.2846170394...	1.451675310674...	3.187983382000...	30.06164098764...
52.90713882446...	1249.291137695...	1.751303553581...	2.875148773193...	32.20600509643...
80.46552830912...	634.2083566738...	1.731105822330...	2.906478556992...	19.51449243640...
46.28286868355...	775.0275253111...	1.328581929106...	2.152973227051...	26.21525371229...
52.90713882446...	1249.291137695...	2.103293657302...	1.873197436332...	28.40210723876...
76.57036180284...	807.6804112875...	1.652305971436...	2.260275462458...	32.51472660619...
52.90713847943...	817.7125635007...	1.355054683286...	2.563230570674...	26.44217077274...
68.34814672654...	1017.881502337...	1.522079493011...	2.885902930055...	30.59392560492...
52.70469404665...	1122.036246982...	1.615010470051...	3.154177791524...	27.58719266838...
63.01699744628...	1027.441975429...	1.519236213799...	2.801025895944...	26.31164682611...
52.90713882446...	1029.137451171...	2.163875341415...	3.422656536102...	28.54625320434...
58.64323340884...	956.4190603812...	1.484479480577...	2.655495808550...	28.01612880166...

Figure III.10 Front Pareto

Et ces composant de front Pareto son même représenter aussi avec un graphe qui comme le suivante :

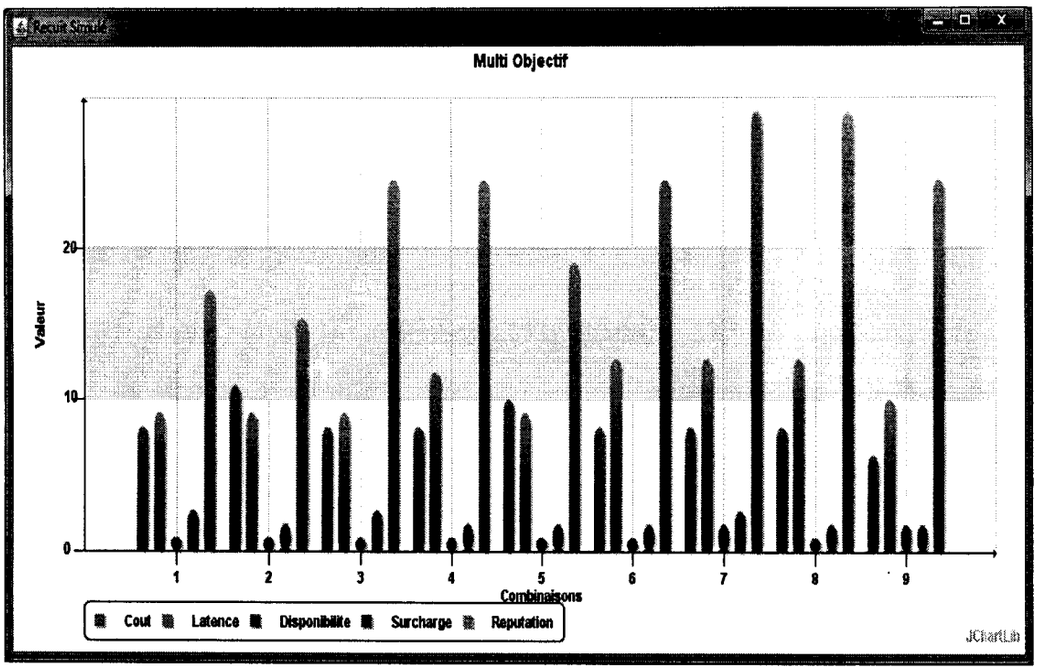


Figure III.11 multi objectif

Référence

[Afef JMAL, 2009] Afef JMAL, 2009, Évaluation de Politiques d'Auto-Adaptabilité basées sur la Mobilité des Services Web Orchestrés, Master, Université de Sfax, République Tunisienne, 82 pages

[Christian Bernard et al. ,2003] Christian Bernard et al. ,2003, Services web avec J2EE et .NET conception et implémentation , ÉDITIONS EYROLLES, ISBN : 2-212-11067-7

[Collette et al. 2002] collette, et Siarry, P. Optimisation multiobjectif _ Eyrolles, 2002

[David MEIGNAN, 2008] David MEIGNAN, 2008, UNE APPROCHE ORGANISATIONNELLE ET MULTI-AGENT POUR LA MODÉLISATION ET L'IMPLANTATION DE MÉTAHEURISTIQUES Application aux problèmes d'optimisation de réseaux de transports, doctorat, Université de Technologie de Belfort-Montbéliard, 161 pages

[Johann Dréo et al. 2003] Johann Dréo et al., 2003, Métaheuristiques pour l'optimisation difficile, édition EYROLLES, ISBN : 2-212-11368-4

[MOUELHI Ouael, 2010] MOUELHI Ouael, 2010, Contribution à l'optimisation multiobjectif en conception multidisciplinaire, doctorat, l'institut national des sciences appliquées de Lyon, France, 170 pages

[Ricardo DE LA ROSA-ROSERO, 2004] Ricardo DE LA ROSA-ROSERO, 2004, Découverte et Sélection de Services Web pour une application Mélusine, Master, université JOSEPH FOURIER, Suisse, 60 pages

[W3C] W3C World Wide Web Consortium; "Web Services Architecture"; W3C Working Group Note 11; February 2004; http://www.w3.org/TR/ws-arch