Table des matières

CHAPITRE II : Optimisation multi-objectif évolutionnaire

CHAPITRE III : Conception et Implémentation du prototype

Listes des figures

Résumé

Les services Web sont des technologies émergentes et prometteuses pour le développement, le déploiement et l'intégration d'applications sur l'Internet. Ils constituent la technologie de base pour le développement d'architectures orientées services. Ces architectures sont de plus en plus répandues sur le Web. Le principe essentiel de l'approche service Web est de transformer le Web en un dispositif distribué d'échange et de calcul, où les services Web peuvent interagir d'une manière intelligente.

Actuellement, de nombreux services Web, avec des fonctionnalités similaires sont fournis par des fournisseurs concurrents, un problème qui apparaît à la mise en œuvre de cette technologie est que le processus de sélection devient assez complexe.

Notre travail consiste à développer une application basée sur une approche d'optimisation multi-objective évolutionnaires pour la sélection des meilleurs services Web en fonction de QoWS.

Le système interroge un espace de recherche sous la forme de collection de services Web et donne comme résultat une liste de meilleurs services possibles (population), dont les paramètres sont optimisés par l'algorithme génétique NSGA-II. Les éléments de la population résultat sont des services qui ont été créés par le biais de la composition de services existants.

Mots Clés : Services Web, Architectures Orientées Services, Sélection, Optimisation multi-objective évolutionnaires, QoWS, NSGA-II.

Introduction Générale

Introduction Générale

Plusieurs innovations technologiques ont eu lieu ces dernières années pour répondre aux besoins de partage des ressources et de communication entre les entreprises. Ceci a conduit à remettre en cause la rechercher de meilleures façons de développer des solutions informatiques [**Chouchani 2010**].

Apparus dès la fin des années 1990, à l'aube du 21ème siècle, les services Web ont provoqué une forte évolution dans le monde de l'informatique distribuée, et un bouleversement majeur dans la façon de concevoir des architectures. Un des intérêts du service Web est de faciliter l'interconnexion entre les différentes applications distantes, indépendamment des plateformes et des langages de programmation utilisés. Les services Web semblent être la solution de l'avenir pour implémenter les systèmes distribués, aujourd'hui, ces services sont distribués à large échelle sur Internet. Le développement d'Internet, la structuration des données via XML, et la recherche d'interopérabilité sont autant de facteurs qui ont favorisé l'essor des services Web. Les services Web constituent la technologie de base pour le développement d'architectures orientées services. Ces derniers sont des modèles qui définissent un système par un ensemble de services logiciels distribués, qui fonctionnent indépendamment les uns des autres afin de réaliser une fonctionnalité globale.

Les architectures orientées services, et en particulier les services Web, permettent l'accessibilité, la découverte et l'utilisation universelle de n'importe quelle application logicielle sur le Web en utilisant des normes ouvertes. Ils constituent un paradigme permettant d'organiser et d'utiliser des services Web distribués. Ainsi, les logiciels codés dans divers langages de programmation et sur diverses plateformes d'exécution peuvent employer des services Web, pour échanger des données à travers le Web. Bien que, les architectures orientées service soient récemment proposées, elles connaissent un engouement et focalisent l'attention de bon nombre de professionnels dans le domaine des technologies de l'information et de la communication, des chercheurs et des industriels [**Boukhadra 2011**].

Contexte du travail :

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. Ils semblent être la solution la plus adaptée pour assurer l'interopérabilité, qui permet de transmettre les données entre les différentes applications d'une organisation. Cette technologie permet de réaliser le traitement de ces données, et gérer les liaisons entre les différentes applications.

La particularité d'une plateforme de services Web réside dans le fait qu'elle utilise la technologie Internet comme infrastructure pour la communication entre les services Web, et ceci en mettant en place un cadre de travail basé sur un ensemble de standards. La technologie services Web est la technologie clé permettant l'intégration des applications via le Web. Les services Web peuvent être définis comme des programmes modulaires, généralement indépendants et auto descriptifs, qui peuvent être découverts et invoqués via l'Internet ou l'Intranet.

La popularité des services Web et leur utilisation dans ces contextes variés s'expliquent par le fait que les services Web peuvent opérer dans des environnements distribués particulièrement hétérogènes, aussi bien, du point de vue des plateformes logicielles déployées que des modèles de programmation utilisés. Les services Web sont généralement fournis par des organisations différentes et indépendamment de tout contexte d'exécution.

Dans la communauté des services Web, des efforts importants sont maintenant consacrés à ce problème, qui consiste a proposé une architecture orienté service qui regroupe la découverte, la composition et la sélection de services Web.

Problématique

Aujourd'hui le besoin d'un client est devenu de nature multi objectif (minimiser les coûts, minimiser les délais, augmenter le taux de service…). De plus, comme les services Web avec des fonctionnalités similaires sont censés être fournis par des fournisseurs concurrents, Il sera donc nécessaire, de fournir un outil d'aide à la décision.

De ce fait, découvrir un service web qui nous intéresse est une chose, alors que découvrir le service web le plus adéquat en est une autre. La qualité de service dans le cas des services web se mesure à l'aide de plusieurs métriques dont les métriques de performance, de disponibilité et de fiabilité. Une recherche sur internet nous permet certainement de trouver plusieurs services web qui remplissent ces critères. Mais lequel sera le plus adéquat, le moins cher, le plus luxueux, le plus rapide, etc., bref, le *meilleur?* Lequel de ces services sera le plus disponible, le plus fiable? Lequel aura un temps réponse *acceptable?* Il devient ainsi nécessaire de choisir les services web pertinents parmi ceux trouvés et de fixer des critères pour choisir les meilleurs.

Tout au long de cette étude, nous considérons que l'étape de composition a été effectuée.

Contribution

Dans le cadre de ce mémoire, nous proposons une approche qui se base sur les algorithmes génétiques multi-objectifs. Ces derniers sont répandus dans divers domaines pour résoudre des problèmes d'optimisation et de recherche. Nous nous inspirons de ces techniques pour décrire une approche de sélection multi-objective des meilleurs services Web.

Notre projet consiste à développer une architecture basée sur les techniques génétiques et vise à aider le concepteur à trouver le bon compromis à partir d'un ensemble de services concurrent. Les algorithmes génétiques multi-objectifs, avec un bon réglage de leurs paramètres, constituent une approche intéressante pour la résolution des problèmes d'optimisation multi-objectifs. De plus, ce domaine est très dynamique et ne cesse de se développer.

Plan du mémoire

Le mémoire est organisé comme suit:

Le premier chapitre est consacré à un état de l'art sur l'architecture orientée service et plus particulièrement la technologie des services Web. Nous donnons la définition et l'historique de l'architecture SOA, ainsi que les caractéristiques et les avantages des services Web et les principaux standards qu'elle supporte.

Le deuxième chapitre porte sur l'optimisation par algorithmes évolutionnaires multicritères. Nous illustrons d'abord les principes fondamentaux de l'approche multicritère. Un état de l'art des techniques évolutionnaires est ensuite proposé en insistant particulièrement sur un algorithme multicritères élitistes, le NSGA-II (Non dominated Sorting Genetic Algorithm-II) qui est utilisé dans ce mémoire.

Le dernier chapitre est consacré à l'illustration des résultats. La première partie, explique une application à partir d'un scénario et mentionne les paramètres, les fonctions et la base de données utilisé, et la seconde partie présente la conception de notre système et l'implémentation pour évaluer la performance de l'approche proposée

La conclusion générale résume les résultats de notre travail, et présente les perspectives que nous souhaitons réaliser dans le futur.

CHAPITRE I

L'architecture SOA et Les services Web

I. Introduction

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et tout particulièrement par la diffusion de l'accès à Internet. Cette évolution du monde informatique a entraîné le développement de nouveaux paradigmes d'interaction entre applications tels que la SOA. Cette dernière été mise en avant afin de permettre des interactions entre applications distantes.

L'architecture SOA est une méthode de conception basée sur des standards, permettant de créer une infrastructure informatique intégrée capable de répondre rapidement aux nouveaux besoins d'un utilisateur. Elle fournit les principes et directives permettant de transformer un réseau existant de ressources informatiques hétérogènes, distribuées, complexes et rigides en ressources intégrées, simplifiées et particulièrement souples pouvant être modifiées et combinées afin de mieux satisfaire les objectifs de l'utilisateur [**Boukhadra 2011**].

L'architecture SOA est un modèle abstrait, qui définit un système par un ensemble d'agents logiciels distribués, qui fonctionnent de concert afin de réaliser une fonctionnalité globale préalablement établie. SOA se présente comme un style architectural. Elle fournit un ensemble de méthodes pour le développement et l'intégration de systèmes dont les fonctionnalités sont développées sous forme de services. L'approche ultime de cette vision consiste, donc, à créer des applications constituées uniquement de services qui interagissent entre eux. Dans ce cas, peu importe où est déployé le service, ce qui importe est que le service remplisse un rôle bien précis [**Schreiner 2005**].

II. Définition de l'architecture SOA

Plusieurs définitions sont utilisées pour définir et expliquer l'architecture SOA. La plupart de ces définitions insiste sur les aspects techniques de l'architecture, alors que, d'autres s'intéressent aux caractéristiques métiers. Les définitions suivantes illustrent différentes vues de la SOA. Cependant, elles convergent toutes vers un seul sens :

II.1 Définition Métier :

« *L'architecture orientée service est un ensemble de méthodes techniques, métiers, procédurales, organisationnelles et gouvernementales pour réduire ou éliminer les frustrations avec les technologies d'information, et pour mesurer quantitativement la valeur métier des technologies d'information, pendant la création d'un environnement métier agile pour un intérêt concurrentiel.* **»** [**Margolisand 2007**].

II.2 Définition Technique :

« *Une architecture SOA est une structure d'intégration de processus métier qui supporte une infrastructure des technologies d'information comme étant des composants et services sécurisés, standardisés et qui peuvent être combinés pour s'adresser aux priorités de changements métiers.* » [**Jennings 2007**].

II.3 Définition Technique plus Large :

« *L'architecture SOA est un paradigme permettant d'organiser et d'utiliser des savoir faires distribués pouvant être de domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoirs faires pour produire le résultat désiré avec des pré-conditions et des buts mesurables.* » [**Josuttis 2007**].

Du point de vue des applications, l'architecture orientée services permet le développement d'une nouvelle génération d'applications dynamiques ou composites. Ces applications permettent aux utilisateurs d'accéder à des informations et à des processus hétérogènes, et de les utiliser de différentes manières, notamment via le Web. Du point de vue de l'infrastructure, l'architecture orientée services permet au service Web de simplifier l'intégration des applications et des systèmes, de recombiner et de réutiliser les fonctionnalités des applications, et d'organiser les différentes phases du processus de développement, dans un cadre cohérent et unifié. En réalité, la philosophie des SOA décompose une application monolithique en une suite de services assurant la modularité dans leurs fonctionnalités [**Devaux 2008**].

II.4 Historique de l'architecture SOA

Dans les années 80, l'architecture orientée objet est apparu comme une nouvelle philosophie de développement basée sur des concepts intéressants : encapsulation, héritage, . . . etc. Le problème de l'approche objet est le fait d'offrir juste la réutilisation technique sans aucune vue métier. Cette architecture rend difficile l'intégration d'applications résidentes dans plusieurs plateformes. Enfin, l'architecture objets est fragile, car un service de ce système est offert comme une méthode d'une classe implémentée par un objet [**Bieberstein 2008**].

Pour pallier les défauts de l'approche objet, l'approche composant est apparue. Le modèle de composants le plus commun dans l'environnement Windows est le COM. Les composants COM dans chacune de ses couches peuvent être réutilisés par d'autres composants et applications. Dans le monde Java, CORBA est le plus utilisé. Il rend possible la distribution

des tâches de développement à travers plusieurs programmeurs, et fait que, le système soit plus robuste, scalable, et maintenable [**Bieberstein 2008**].

Les composants sont des entités logicielles indépendantes. Ils interagissent à l'aide d'une infrastructure qui permet de gérer la communication entre des composants au sein d'un même système ou à travers un réseau, via une décomposition du logique métier en composants distribués. L'architecture de composants distribués a engendré un développement rapide et évolutif d'applications distribuées et complexes. L'écriture des composants pouvant être partagés doit prendre en considération que les différents langages de programmation sont incompatibles. Un composant écrit en C++ peut avoir des anomalies de fonctionnement, s'il est utilisé dans un environnement où le Visual Basic est le langage principal. L'interopérabilité des plateformes n'était plus facile avec les modèles de composants utilisés ces dernières années. Cette interopérabilité s'avère encore plus difficile si le composant à invoquer est situé au-delà d'un pare-feu [**Erl 2008**].

La mise en œuvre de ces deux architectures soulève des difficultés dans le cadre d'une infrastructure ouverte telle qu'Internet. En effet, ces architectures, bien qu'utilisant un modèle objet distribué, proposent, chacune, sa propre infrastructure. Ce qui impose une forte liaison (un fort couplage) entre les services offerts par les composants et leurs clients. Ainsi, on ne peut assembler que des objets CORBA (ou COM) entre eux. Le résultat est que les systèmes construits à base de ces architectures sont homogènes.

Dans les années 2000, sont apparus les services Web pour pallier à tous ces problèmes, et en ne s'intéressant qu'à la manière d'interagir avec ce service Web sans connaître sa structure ou sa technologie. Une application orientée services Web est simplement l'agrégation de services en une logique simple et en une application unifiée. Cependant, la clé de l'interopérabilité entre les services Web est l'utilisation des protocoles standards, des messages, et des contrats.

En outre, après l'avènement du B2C, où les entreprises mettaient en ligne leurs services pour leurs consommateurs à travers des applications Web, celles-ci souhaitaient accroître leurs productivités à l'aide du paradigme B2B. Le B2B repose sur l'échange de produits, d'informations et de services entre entreprises. Ceci implique l'utilisation de services et la collaboration avec des systèmes proposés par d'autres concepteurs, et par conséquent, une maîtrise de l'hétérogénéité [**Papazoglou 2003**].

L'interopérabilité est ainsi devenue une nécessité pour l'entreprise dans le monde du B2B. C'est justement ce que les services Web apportent par rapport aux solutions dites homogènes. D'une certaine façon, le modèle des services Web n'est pas une révolution, mais une

évolution du modèle des composants distribués. Cette évolution est rendue nécessaire par l'utilisation intensive d'Internet [**Papazoglou 2003**].

II.5 Caractéristiques de l'architecture SOA

L'approche SOA confère aux entreprises plus de flexibilité dans leur activité en améliorant les applications et l'infrastructure informatique. Elle contribue à réduire les dépenses informatiques. Elle fournit un accès rapide à des informations plus précises et permet à l'entreprise d'identifier et de résoudre plus efficacement les problèmes de flux. Parmi ces caractéristiques, on peut citer les plus primordiales suivantes [**Bejaoui 2008**] [**Kaabi 2008**] [**Albertella 2009**] :

II.5.1 Autonomie : Le service est indépendant de son environnement tout en gardant des relations de collaborations. Cette notion d'indépendance se traduit, aujourd'hui, par le fait qu'un service est indépendant du serveur car il a son propre conteneur qui peut être une simple application. Le service est doté d'une sécurité autonome, et il doit protéger ses fonctions et ses messages envoyés sans avoir besoin de connaitre le degré de sécurité des clients.

II.5.2 Frontières explicites : Les messages d'interaction avec un service sont sous forme de contrat en XML pour assurer l'interopérabilité. Les messages permettent de créer des systèmes faiblement couplés qui recouvrent plusieurs systèmes d'exploitation.

II.5.3 La plateforme : La plateforme utilisée pour supporter une implémentation d'un service ne doit pas être pertinente aux consommateurs. Ceci inclut les couches intermédiaires du système d'exploitation, du protocole de communication et même les couches de l'application. Cependant, l'architecture liant les demandeurs et fournisseurs de services est sous un contrôle direct. On peut gagner des avantages significatifs dans les performances et dans la gestion du système en évitant la séparation de cet aspect de l'architecture.

II.5.4 Architecture décentralisée : Parmi les grands avantages des SOA, est la décentralisation de l'architecture avec orchestration automatique. L'emplacement de différentes instances d'un service dans différentes localisations est l'un des facteurs primordiaux, pour avoir une architecture moins couplée. La Scalabilité d'une livraison d'un service est facile à atteindre en mettant plusieurs instances de celui-ci dans des terminaux différents. L'identité d'un service fournisseur peut être négociée à travers un composant, jouant le rôle d'intermédiaire. La négociation peut être selon le critère géographique de la localisation du service, le critère de l'identité du client, d'une information sur le plan de

l'adhésion des membres ou d'autres critères d'attachement des clients à leurs fournisseurs de services.

II.5.5 Les protocoles : De même que les considérations de la plateforme de déploiement, l'indépendance des protocoles peut ou ne pas être nécessaire en fonction du modèle métier et du contexte. Comme l'exemple d'une entreprise de vente en détail, où les protocoles entre l'entreprise et ses branches sont totalement contrôlés par l'entreprise. Ainsi, le choix des protocoles peut être unifié, bien que ceci ne limite pas l'exposition d'un service pour être utilisé par d'autres protocoles en même temps. En pratique, les protocoles de communications sont définis d'habitude par un fichier de configuration dans l'interface d'un service SOA. Ceci permet de modifier cette configuration au besoin sans être obligé de modifier le code du service.

II.5.6 Le langage de programmation : Une SOA doit être implémentée indépendamment des spécifications des langages de programmation. Cependant, la pratique a révélé quelques problèmes d'interopérabilité entre les demandeurs et les fournisseurs de services en matière de représentation de types de données très complexes (tableaux, pointeurs nuls, . . . etc.). Ils sont implémentés différemment dans différents systèmes et qui ont différents comportements en performances.

II.5.7 Les modèles d'invocation : Un modèle d'invocation est la circulation générale des flux d'interactions entre un demandeur et son fournisseur de service. Un service peut avoir besoin d'être synchronisé avec un autre service, il peut avoir aussi besoin de fonctionner d'une façon asynchrone si le modèle le permet.

II.5.8 Le modèle d'un service : On peut décrire les différents services et leurs relations chacun avec l'autre en termes d'un modèle, qui peut être sauvegardé, échangé et utilisé pour automatiser les interactions et faciliter la création d'une orchestration des services. Ceci nous permet d'utiliser dans plusieurs cas les mêmes services, pour créer des modèles d'interaction différents, et pour la construction de nouveaux domaines ou de nouvelles solutions.

Le paradigme orienté services représente une nouvelle tendance d'ingénierie logicielle. Il assure un développement d'applications plus rapide et à moindre coûts. SOA est une architecture fournissant une infrastructure nécessaire, pour intégrer les applications isolées afin de les utiliser an tant que services dans un réseau. La SOA est entrée depuis peu de temps dans le domaine du réel, grâce à un ensemble de normes appelées collectivement services

Web. Les services Web, réalisation concrète des architectures SOA, sont la déclinaison du paradigme des architectures orientées service, sur le Web.

Ces services Web constituent un moyen de plus en plus normalisé, étendu et puissant pour mettre en œuvre une architecture SOA. Cette architecture est construite autour de la notion de service Web, qui est matérialisée par un composant logiciel assurant une fonctionnalité particulière et accessible via son interface. Un service Web est toujours accompagné d'une description fournissant aux applications les informations nécessaires à son utilisation. En réalité, la philosophie des SOA décompose une application homogène en une suite de services assurant la modularité dans leurs fonctionnalités [**Brown 2008**].

III. Définition des services Web

De nos jours, les entreprises expriment un grand besoin pour échanger des informations et des services. Ceci nécessite des langages communs de communication. Les efforts d'élaboration de ces langages ont donne lieu a une technologie émergeante qui a permis de tracer quelques pistes intéressantes pour la communication entre entreprises. Cette technologie est celle de services web [**Fakhfakh 2006**].

Les services Web constituent le développement ultime dans ce domaine. Le terme service Web est souvent utilisé de nos jours, mais pas toujours avec la même signification, car la signification est tributaire de l'application de cette technologie.

On peut dire qu'un service Web est souvent vu comme une application accessible à d'autres applications sur le Web, mais il existe plusieurs définitions pour les services Web :

III.1 Définition : Le consortium W3C définit un service Web comme étant : **«** *une application, ou un composant logiciel qui vérifie les propriétés suivantes* **»** [**Cerami 2002**] :

- Il est identifié par un URI.
- Ses interfaces et ses liens peuvent être décrits en XML.
- Sa définition peut être découverte par d'autres services Web.
- Il peut interagir directement avec d'autres services Web à travers le langage XML en utilisant des protocoles Internet standards.

III.2 Définition : « *Un service Web est une application accessible à partir du Web. Il utilise les protocoles Internet pour communiquer, et utilise un langage standard pour décrire son interface.* **»** [**Melliti 2004**].

III.3 Définition : « *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 2008**].

III.4 Caractéristiques des services Web

Plusieurs acteurs définissent les services Web par des caractéristiques technologiques distinctives, qui sont [**Cerami 2002**] :

III.4.1 Un service Web est une application logicielle qui est reconnue par un URI : URI est la façon d'identifier un point de contenu sur le Web comme un document tel qu'un texte, audio ou vidéo. L'URI le plus connue est l'adresse d'une page Web. Le service Web est donc accessible en spécifiant son URI, c'est-à-dire que le service Web est caractérisé par un seul objet et une seule fonctionnalité. A partir de cela, on peut faire la construction d'une application logicielle très large comportant plusieurs fonctionnalités, afin de sélectionner les fonctionnalités qui sont recherchées par les URI spécifiques.

III.4.2 Capacité des interfaces et liaisons d'être publiées, localisées et invoquées via le langage XML : Les principales tâches d'un service Web sont : La publication dans un registre, la localisation en interrogeant le registre qui l'héberge et l'invocation par un ou plusieurs services Web après sa localisation. Ces tâches sont réalisées via l'utilisation d'XML.

III.4.3 Capacité d'interagir avec les composantes des logiciels via des éléments XML avec l'utilisation des protocoles Internet standards : Un service Web est créé pour être utilisé et interagir avec d'autres logiciels contrairement à une page Web, ou à une autre application qui n'utilise pas les services Web. C'est l'interopérabilité basée sur l'utilisation de l'XML et les protocoles Internet standards, par exemple, HTTP, SMTP, FTP, . . . etc.

III.4.4 Composante logicielle légèrement couplée à interaction dynamique : Un service Web ayant un programme qui permet de l'invoquer est appelé consommateur de service Web. Le service Web et son consommateur sont indépendants l'un de l'autre. Si une modification est à faire sur le consommateur, on n'a pas besoin de connaitre la machine, le langage de programmation, le système d'exploitation ou autre paramètre, afin d'établir à nouveau une communication entre le service Web et son consommateur. Le consommateur possède une

fonctionnalité qui correspond à faire une localisation et une invocation sur le service Web, au moment de l'exécution du programme de service Web de manière automatique.

III.5 Avantages des services Web

L'idée essentielle derrière les services Web est de partager les applications et les programmes en un ensemble d'éléments réutilisables appelés service, de sorte que, chacun de ces éléments effectuent une tâche principale et efficace, afin de faciliter l'interopérabilité entre tous ces services Web. D'autre part les services Web [**Chappell 2002**] :

- Permettent l'interopérabilité dans des environnements applicatifs, cela veut dire, que les logiciels et les applications écrits dans différents langages de programmation, et évoluant sur différents systèmes d'exploitation peuvent communiquer et/ou échanger des données entre eux facilement.
- Permettent de profiter de différents environnements et langages de développement par une publication, localisation, description et une invocation via XML. Les services Web sont très flexibles, indépendants des langages de programmation et des systèmes d'exploitation.
- Permettent d'accéder aux applications à travers les pare-feux à l'aide d'utilisation via le langage XML et les protocoles Internet standards comme HTTP sur le port 80, qui est généralement ouvert. Cela permet d'assurer une transmission des données transactionnelles et sécurisé.
- Utilisables à distance via n'importe quel type de plateforme, et sont accessibles depuis n'importe quel type de clients.
- Peuvent servir au développement d'applications distribuées. Ils appartiennent à des applications capables de collaborer entre elles de manière transparente pour l'utilisateur, et permettent d'avoir un partage des fonctionnalités et facilitent grandement le développement.

III.6 Architecture des services Web

Techniquement, un service Web peut donc être perçu comme étant une interface décrivant une collection d'opérations accessibles via le réseau, à travers des messages XML standardisés. D'un point de vue technique, la description d'un service Web inclut tous les détails nécessaires à l'interaction avec le service constituant, ce qu'on appelle l'architecture des services Web, par exemples, le format des messages, les signatures des opérations, le protocole de transport et la localisation du service Web [**Kreger 2001**].

L'interopérabilité est l'objectif premier des services Web. Pour permettre cet échange d'information entre des applications distantes, les services Web sont composés des couches standards. Deux types d'architecture existent pour les services Web : La première dite de référence, elle contient trois couches principales. La seconde architecture est plus complète, elle utilise les couches standards de la première architecture en ajoutant au-dessus d'autres couches plus spécifiques. Elle est appelé architecture étendue ou encore en Pile [**Kreger 2001**].

III.6.1 Architecture de référence

Cette architecture vise trois objectifs importants [**Kreger 2001**] :

- Identification des composants fonctionnels.
- Définition des relations entre ces composants.
- Etablissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

L'architecture de référence s'articule autour des trois rôles suivants (Figure I.1) :

Le fournisseur de service : Correspond au propriétaire du service Web. D'un point de vue technique, il est constitué par la plateforme d'accueil du service Web.

Le client : Correspond au demandeur de service Web. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service Web. L'application cliente peut être elle-même un service Web.

L'annuaire des services : Correspond à un registre de descriptions des services Web offrant des facilités de publication des services Web à l'intention des fournisseurs, ainsi que des facilités de recherche des services Web à l'intention des clients.

Les interactions de base entre ces trois rôles incluent les opérations de publication, de recherche et de liens d'opérations. Nous citons, notamment les standards émergents suivants [**Kreger 2001**]:

SOAP : standard supporté par le W3C, il définit un protocole de transmission de messages basé sur XML.

WSDL : soumise en tant que note auprès du W3C en mars 2001, il introduit une grammaire pour la description des services Web.

UDDI : standard supporté par le consortium OASIS, il fournit l'infrastructure de base pour la publication, et la découverte des services Web.

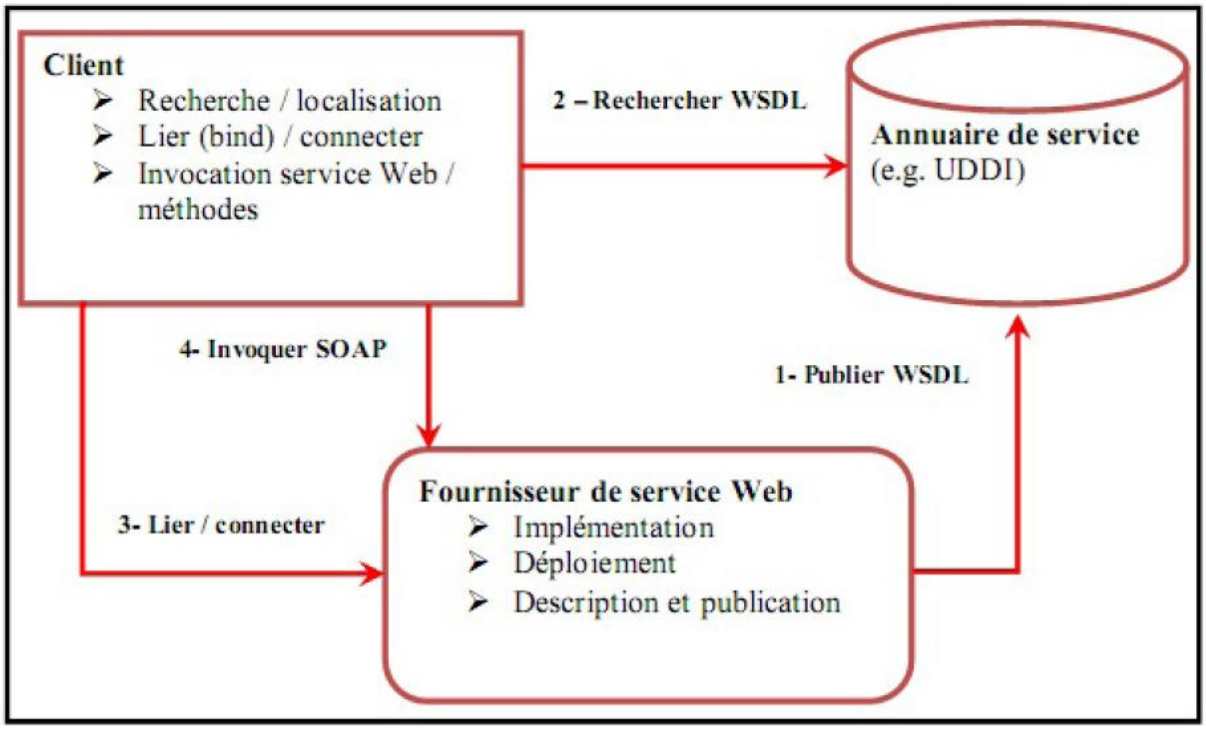


Figure I.1 : Architecture de référence des services Web.

Cependant, cette infrastructure n'est pas suffisante pour permettre une utilisation effective des services Web, dans les domaines dont les exigences vont au-delà de la capacité d'interactions simples via des protocoles standards. Par exemple, dans le domaine du e-business, une nouvelle architecture suffisante et complète est donc nécessaire d'être introduite.

III.6.2 Architecture étendue

Une architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres. La pile est constituée de plusieurs couches, chaque couche s'appuyant sur un standard particulier. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment.

Nous apportons une explication de la mise en relief des trois types de couches (Figure I.2) :

L'infrastructure de base (Discovery, Discription, Exchange) : Ce sont les fondements techniques établis par l'architecture de référence. Nous distinguons les échanges des messages établis par SOAP (W3C), la description de service par WSDL (W3C) et la recherche de services Web que les organisations souhaitent utiliser via le registre UDDI (OASIS).

Couches transversales (Security, Transactions, Administration, QoS) : Ce sont des couches qui rendent viable l'utilisation effective des services Web dans le monde industriel.

La couche Business Processus (BusinessProcess) : Cette couche supérieure permet l'intégration de services Web, elle établit la représentation d'un **«** *BusinessProcess* **»** comme

un ensemble de service Web. De plus, la description de l'utilisation de différents services composant ce service est disponible par l'intermédiaire de cette couche.

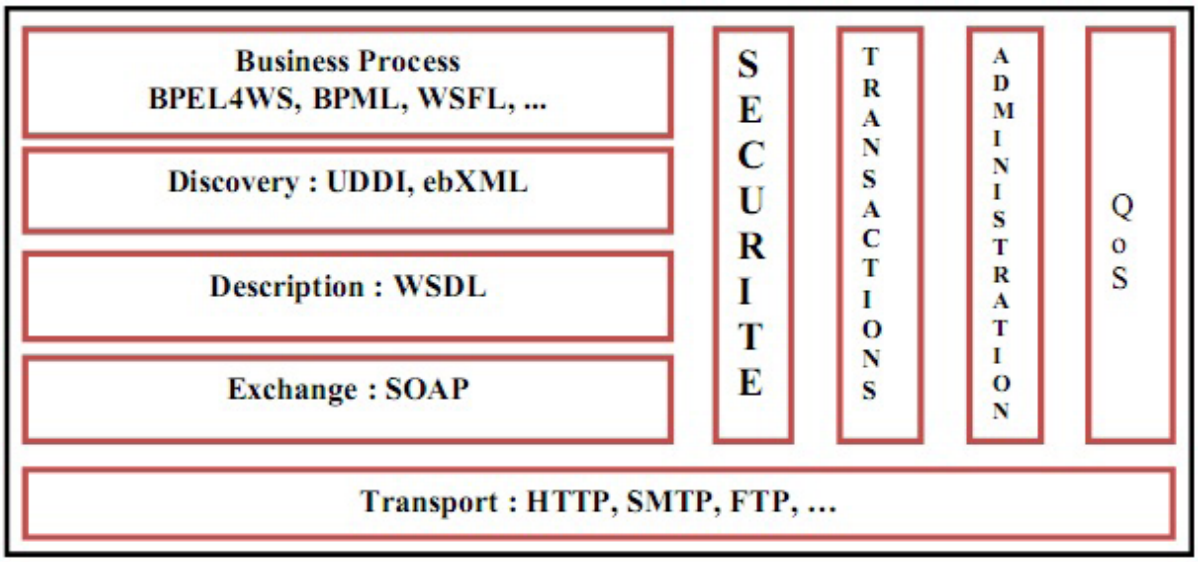


Figure I.2 : Architecture en Pile des services Web.

La technologie des services Web offre de fortes potentialités pour surmonter les problèmes d'interopérabilité des systèmes. Elle constitue un cadre prometteur pour l'intégration des applications, et pour la gestion des interactions entre divers partenaires dans un environnement distribués, hétérogènes, ouvert et versatile qui est le Web [**Bertino 2004**].

III.7 Les principales technologies standards autour de service Web :

L'infrastructure des services web s'est concrétisée autour de trois spécifications considérées comme des standards, à savoir SOAP, WSDL (W3C) et UDDI (OASIS). Nous les détaillons dans la section suivante :

III.7.1 Le protocol SOAP

SOAP (W3C), veut dire « Simple Object Access Protocol » et la traduction de cette définition en français donnerait « Protocole Simple d'Accès aux Objets ». En effet, le protocole SOAP consiste à faire circuler du XML via du HTTP sur le port 80. Cela facilite grandement les communications, car le XML est un langage standard et le port utilisé est le port 80 qui ne pose pas de problèmes pour les firewalls. SOAP peut donc être utilisé dans tous les styles de communication : synchrone ou asynchrone, point à point ou multipoint, Intranet ou Internet [**Kulchenko 2001**].

Le SOAP est un protocole à la fois simple et facile à implémenter dans les serveurs Web, destiné à l'échange d'informations dans un environnement distribué et décentralisé [**Gardien 2002**].

Le SOAP fait partie de la couche de communication des services Web. La force de ce protocole réside dans son universalité et sa flexibilité. Il définit la structure des messages XML utilisés par les applications pour dialoguer entre elles. Par exemple, un client SOAP Java s'exécutant sur Linux, ou un client SOAP PERL s'exécutant sur Solaris peut se connecter à un serveur SOAP Microsoft s'exécutant sur Windows 2000. L'invocation d'une méthode d'un service Web depuis un client suppose la coopération de plusieurs couches logicielles.

La figure I.3 montre un exemple de message SOAP requête d'un service web qui additionne deux entiers.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv: Header/>
  <soapenv:Body>
   <Addition xmlns="http://doc">
       <a>4</a>
       <b>7</b>
   </Addition>
  </soapenv:Body>
  </soapenv:Envelope>
```

Figure I.3 : Le formatage visuel d'un message SOAP

Le message est englobé dans une enveloppe et divisés en 2 parties : l'entête et le corps.

- L'entête (*Header*) : offre des mécanismes flexibles pour étendre un message SOAP sans aucune préalable connaissance des parties communicantes. Les extensions peuvent contenir des informations concernant l'authentification, la gestion des transactions, le payement, etc.
- Le corps (*Body*) : offre un mécanisme simple d'échange des informations mandataires destinées au receveur du message SOAP. Cette partie contient les paramètres fonctionnels tels que le nom de l'opération à invoquer, les paramètres d'entrés et de sortis ou des rapports d'erreur [**Ben Halima 2009**].

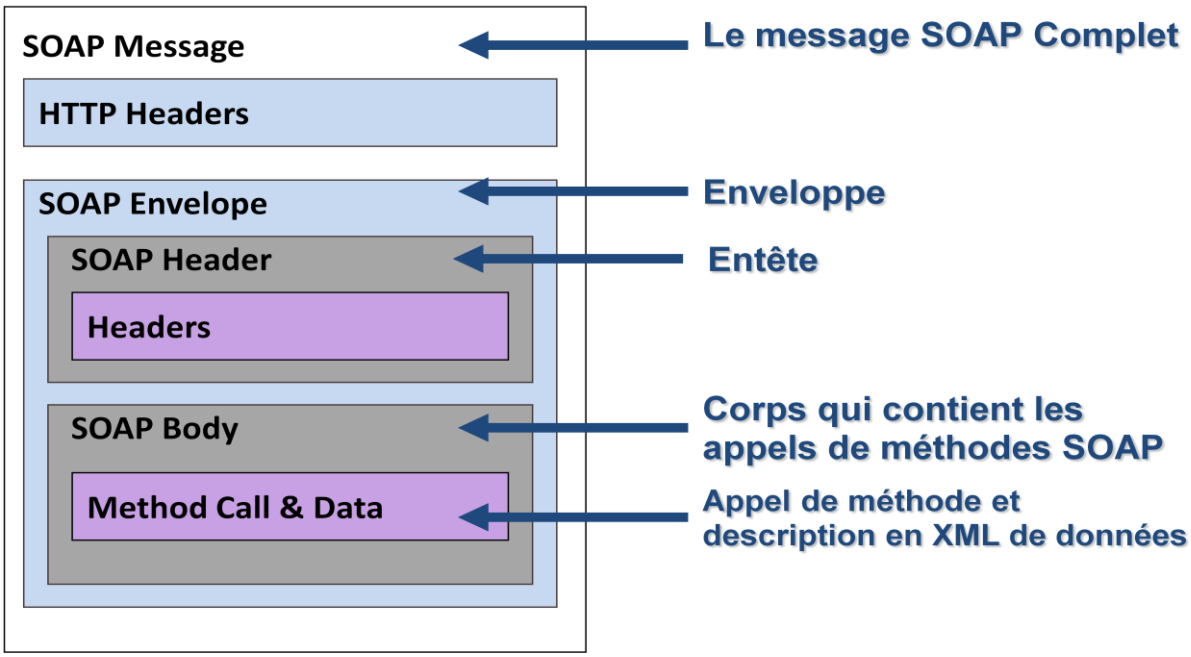


Figure I.4 : les composants d'un message SOAP

III.7.2 Le langage WSDL

Le langage WSDL (W3C), est l'acronyme de « Web Service Description Language »est un langage basé sur XML permettant de décrire et de publier les interfaces et protocoles des services Web d'une manière standard. L'interface d'un service Web décrit en fait tout le fonctionnement d'un service Web, et cache tout le détail de l'implémentation du service Web, donc elle est indispensable pour pouvoir invoquer un service Web par une application cliente, ou un autre service Web permettant une utilisation indépendante de la plateforme utilisée ainsi que du langage utilisé [**Scott 2002**].

Le langage WSDL présente un format commun pour la description et la publication des interfaces et protocoles relatifs aux services Web. Une description WSDL d'un service Web est faite sur deux niveaux, niveau abstrait et niveau concret. Au niveau abstrait, la description du service Web consiste à définir les éléments de l'interface du service Web tel que : [**Chinnici 2004**] [**Gardien 2002**].

Les types de données : « Data types » est l'élément qui définit les types de données utilisées dans les messages échangés par le service Web. Une fois définie, les « Data types », ou type peuvent être référencés dans n'importe quel message.

Les messages : L'élément « Message »spécifie les types d'opérations supportées par le service Web, il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier

les caractéristiques du flux de données, par exemple, un message Input et un message Output corrélés sont mis en correspondance dans une seule opération de type « Request/Response ».

Les Opérations : l'élément « Operation »spécifie les types d'opérations supportées par le service Web, il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier les caractéristiques du flux de données.

Les PortType : le « PortType »est un groupement logique ou une collection d'opérations supportées par un ou plusieurs protocoles de transport, il est analogique à une définition d'un objet contenant un ensemble de méthodes.

Les Liaisons : décrit la façon dont un type de port est mis en œuvre pour un protocole particulier (HTTP par exemple), et un mode d'invocation (SOAP par exemple). Cette description est faite par un ensemble donné d'opérations abstraites. Pour un type de port, on peut avoir plusieurs liaisons, pour différencier les modes d'invocation ou de transport de différentes opérations.

Au niveau concret, le service Web est défini grâce aux deux éléments : Port et Service. Ces deux dernières décrivent des informations liées à un usage contextuel du service Web. On y trouve : l'adresse du fournisseur implémentant le service, et le service qui est représenté par les adresses des fournisseurs.

L'élément Port : l'élément « Port », dans la partie concrète, spécifie une adresse URL qui correspond à l'implémentation du service Web par un fournisseur, et identifie un ou plusieurs « Bindings » (ou liaisons) aux protocoles de transports (HTTP, SMTP, FTP, . . .) pour un « Port-Type » donné. La séparation du protocole de transport de la définition du « PortType» permet à un service Web d'être valable à travers plusieurs protocoles de transports, sans avoir à redéfinir l'ensemble du fichier WSDL.

L'élément Service : spécifie l'adresse complète du service Web, et permet à un point d'accès d'une application distante de choisir à exposer de multiples catégories d'opérations pour divers types d'interactions.

III.7.3 L'annuaire UDDI

UDDI (OASIS), a été conçu en 2000 à l'initiative d'un ensemble d'industriels (Ariba, IBM, Microsoft), en vue de devenir le registre standard de la technologie des services Web. Pour convenir à la technologie des services Web, les services référencés dans UDDI sont accessibles par l'intermédiaire du protocole de communication SOAP, et la publication des informations concernant les fournisseurs et les services doit être spécifiée en XML afin que la recherche et l'utilisation soient faites de manière dynamique et automatique. UDDI constitue

un méta-service possédant des fonctions de publication et de recherche [**Lopez-Velasco 2008**].

En clair, l'UDDI gère l'information relative à la publication, la découverte et l'utilisation d'un service Web. Ce standard définit donc, un registre des services Web sous un format XML. Les organisations publient les informations décrivant leurs services Web dans l'annuaire, et l'application client ayant besoin d'un certain service, consulte cet annuaire pour la recherche des informations concernant le service Web qui fourni le service désiré, pour une éventuelle interaction, ainsi l'UDDI a été créé pour faciliter la découverte de services Web en plus de leurs publications. Par une API SOAP, on peut interagir avec l'UDDI au moment de la conception et l'exécution des applications afin de découvrir des données techniques, et administratives sur les entreprises et leurs services Web. L'annuaire UDDI repose sur le protocole SOAP, les requêtes et les réponses sont des messages SOAP [**Scott 2002**] [**Gardien 2002**].

L'UDDI est subdivisé en deux parties principales : partie publication ou inscription, et partie découverte.

La partie publication regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement. La partie découverte facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP [**Newcomere 2004**].

L'UDDI peut être vu comme un annuaire contenant les parties suivantes :

Les pages blanches (*White Paper*). Ce composant permet de connaître les informations à propos de l'organisation proposant le service. Cette description contient toutes les informations jugées pertinentes pour identifier l'organisation (telles que son nom, son adresse physique). Le futur client du service retrouve dans les pages blanches les informations que le fournisseur a renseignées dans l'élément *Business Entity* lors de la publication.

Les pages jaunes (*Yellow Paper*). Les pages jaunes d'UDDI détaillent la description de l'organisation faite dans les pages blanches en répertoriant les services proposés. Dans cette section, sont décrits : la catégorie de l'entreprise, le secteur d'activité dans lequel exerce l'entreprise, les services offerts par cette organisation, le type de services et les conventions d'utilisation (prix, qualité de service,…etc.). La description des services contenue dans les pages jaunes est non technique et est renseignée par les fournisseurs eux mêmes.

Les pages vertes (*Green Paper*). Les pages vertes comportent les informations techniques liées aux services Web et basées sur leur description WSDL. À l'origine, il existait des registres UDDI dits publics (tels que ceux de Microsoft ou IBM) pour lesquels n'importe qui

pouvait devenir, soit fournisseur, soit client de services Web. L'universalité de ces registres devait amener UDDI à devenir le standard de publication des services Web. En 2006, le nombre de services Web publiés a atteint le nombre de 50000. Malgré ce nombre, UDDI n'a jamais atteint son but : devenir le registre standard des services Web [**Lopez-Velasco 2008**].
Cependant, d'après [**Dovey et al. 2005**], les spécifications UDDI souffrent de certaines limitations : il n'existe pas de plate-forme d'édition ; les API de UDDI sont insuffisantes pour développer efficacement des méthodes de publication et de recherche, le modèle de recherche de UDDI est pauvre (recherche portant sur l'identifiant, le nom du service, ou sur les éléments du document WSDL).

IV. Conclusion

De plus en plus, avec l'essor d'Internet, le développement tend vers les technologies du Web. Les Services Web sont des composants logiciels représentant une fonction applicative, ils représentent un mécanisme de communication entre applications distantes à travers le Web. Les services Web permettent le changement de la nature du Web, du Web du document utilisé par les organisations pour la publication des informations, au Web orienté service, qui permet aux serveurs d'applications la communication entre eux.
Les services Web sont suffisamment développés pour que les développeurs les utilisent maintenant dans tous les domaines de l'informatique, afin de récolter les divers bénéfices de la technologie. Ils peuvent être vus comme des ressources actives, dynamiques, relativement à des ressources statiques, celles contenues dans les bases de données disponibles sur le Web. Si un service Web est vu comme une ressource, alors les mêmes problèmes vont se poser : comment les décrire, les invoquer, les composer et les sélectionner.

CHAPITRE II

Optimisation multi-objectif évolutionnaire

I. Introduction

Les ingénieurs, les économistes, les décideurs se heurtent quotidiennement à des problèmes d'optimisation complexes de grande dimension (traitement d'images, conception de systèmes, conception d'emplois du temps...) pour lesquels les décisions doivent être prises de façon optimale. Ces problèmes sont rarement uni-objectifs : il y a généralement plusieurs objectifs contradictoires à satisfaire simultanément. Contrairement à l'optimisation mono-objective, la solution d'un problème multi-objectif n'est pas une solution unique, mais un ensemble de solution compromis, connu comme surface de Pareto. L'optimisation multi-objective s'intéresse à la résolution de ce type de problèmes. Elle possède ses racines dans le 19ème siècle dans les travaux en économie d'Edgeworth et Pareto [**Ben Abdallah 2005**].

Qu'ils comportent un seul ou plusieurs objectifs, les problèmes d'optimisation sont en général difficiles à résoudre. De plus, le temps de calcul nécessaire à leur résolution peut devenir si important que l'algorithme développé devient inutilisable en pratique. Plusieurs méthodes ont été développées pour tenter d'apporter une réponse satisfaisante à ces problèmes. Parmi celles-ci, nous distinguons les méthodes approchées souvent appelées méta heuristiques. Le but de telles méthodes est de produire des solutions de meilleure qualité possible avec un temps de calcul raisonnable [**J. Dréo et al, 2003**]. Parmi ces approches, nous nous intéressons dans ce chapitre aux méthodes d'optimisation multi-objectif évolutionnaires, qui ont reçu un intérêt croissant ces dernières années.

De ce fait, ce chapitre est divisé en trois grandes parties :

La première partie expose les principes de l'optimisation multi-objectifs. Nous effectuons un bref rappel dans lequel nous évoquons les terminologies les plus usuelles en optimisation.

La deuxième partie dresse les algorithmes évolutionnaires en rappelant les principes fondamentaux sur lesquels ils s'appuient.

La troisième partie développe plus particulièrement l'algorithme élitiste de référence (à savoir le NSGA-II) qui sera utilisé dans la suite de ce travail.

II. Optimisation multi-objectifs

II.1 Définition

L'optimisation multi-objective [**Ejday 2011**] peut être définie comme la recherche de la meilleure ou des meilleures solutions possibles d'un problème donné. Souvent, les problèmes d'optimisation sont des problèmes multi-objectifs. Si les objectifs sont antagonistes, alors on n'a pas une seule solution optimale mais un ensemble de solutions de compromis.

Un problème d'optimisation multi-objectifs consiste à rechercher les meilleures solutions qui minimisent un nombre *M* de fonctions, appelées fonctions coût, $f_m(X)$, $m = 1,..., M$, par rapport à un vecteur *X*, qui est le vecteur des *n* variables de contrôle (ou paramètres) : $X = [x_1\ x_2 \dots x_n]^T$, en satisfaisant un certain nombre de contraintes, explicites comme les contraintes de borne $(x_i^1 \le x_i \le x_i^s, i = 1, \dots, n)$ ou implicites, d'égalité $h_k(X) = 0,\ k = 1,..,K$ ou d'inégalité $g_j(X) \geq 0,\ j = 1,..,j$.

Généralement, on se limite aux problèmes de minimisation, puisque la maximisation d'une fonction $f(X)$ peut facilement être transformée en un problème de minimisation :

$$\max(f(X)) = -\min(-f(X)) \quad \{18\}$$

Un problème d'optimisation s'écrie sous la forme mathématique suivante : [**Ejday 2011**]

Minimiser $f_m(X),\ m = 1,..,M$

Satisfaisant $\begin{cases} g_j(x) \geq 0, j = 1, \dots, J \\ h_k(x) = 0,\ k = 1,..,K \\ (x_i^1 \le x_i \le x_i^s, i = 1 \dots n) \end{cases}$

Quand le problème d'optimisation [**Deb, 2001**] contient une seule fonction coût (*M=1*), il est appelé mono-objectif. Ce problème, utilise un seul espace de recherche, connu aussi sous le nom d'espace de décision S. Il s'agit de l'ensemble des combinaisons possibles des paramètres qui satisfont toutes les contraintes.

En revanche, si le problème d'optimisation contient plusieurs fonctions coût (*M>1)*, et si les fonctions coût sont antagonistes, il est appelé multi-objectifs [**Deb, 2001**]. Un problème d'optimisation multi-objectif porte sur deux espaces, que sont l'espace de décision *S* et l'espace fonctionnel (l'espace des fonctions coût) *F*.

II.2 Vocabulaire et Terminologie

Nous donnons ici quelques définitions des principaux termes fréquemment utilisés dans ce mémoire :

II.2.1 Vecteur de décision : c'est le nom donné au vecteur $\vec{x}$. Il correspond à l'ensemble des variables du problème [**Barichard 2003**].

II.2.2 Fonction objectif : c'est le nom donné à la fonction f (appelé aussi fonction de coût ou fonction fitness) [**Barichard 2003**].

II.2.3 Notion de dominance au sens de Pareto : V. Pareto, formule le concept suivant [**Abdelli 2007**] : dans un problème multi-objectif, il existe un équilibre tel que l'on ne peut pas améliorer un objectif sans détériorer au moins un des autres. Cet équilibre a été appelé optimum de Pareto. Un point est dit Pareto – optimal s'il n'est dominé par aucun autre point appartenant à l'espace des solutions. Ces points sont également appelés solutions *non inférieures* ou *non dominées*.

Un point $X \in E$ *domine* $Y \in E$ si est seulement si :

$\forall\ i \in E,\ f_i(x) \leq f_i(y)$

et, $\exists\ j,$ tel que $f_j(x) < f_j(y)$ [**Abdelli 2007**].

La figure II.1 illustre un exemple graphique des définitions mentionnées ci-dessus, le point noir :

- domine chacun des carrés,
- est dominé par chacun des triangles, et
- est équivalent aux anneaux au sens de la dominance [**Ben Abdallah 2005**].

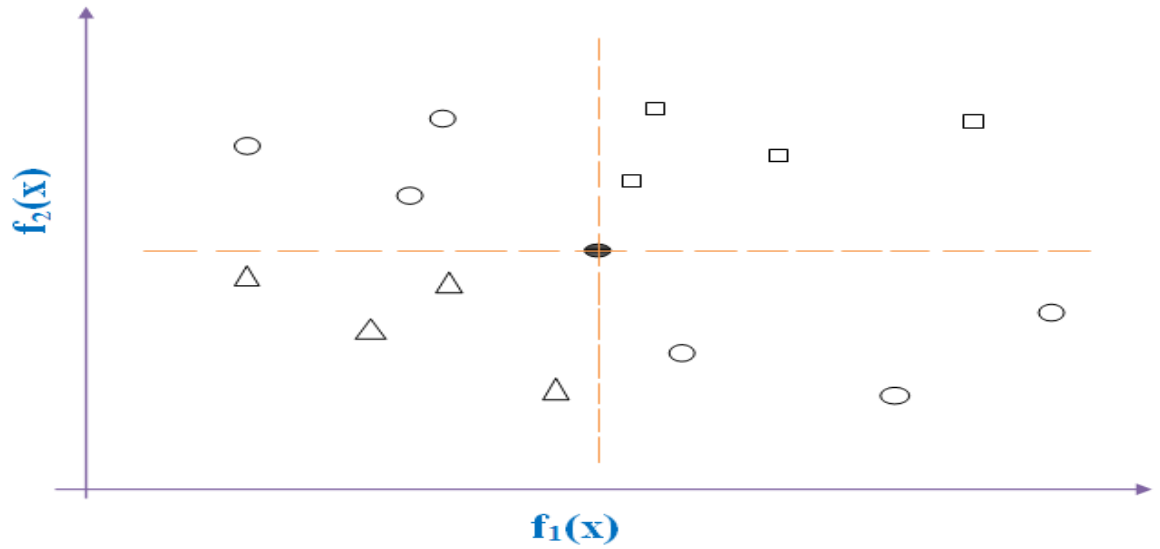


Figure II.1 Exemple de dominance

II.2.4 Optimum au sens de Pareto : dans le cas le plus général, les solutions optimales d'un problème multi-objectif (qualifiées aussi de solutions Pareto-optimales) sont constituées par l'ensemble des solutions équivalentes de l'espace des objectifs qui ne sont jamais dominées au sens strict. Nous définissons les notions d'optimum local et d'optimum global au sens de Pareto comme suit :

Optimum global au sens de Pareto : une solution **X** est considérée comme un optimum global au sens de Pareto s'il n'existe aucune solution **Y** qui domine **X** dans l'espace des objectifs F. La figure II.2 illustre une représentation graphique de l'optimalité globale au sens de Pareto dans le cas d'un espace à deux objectifs [**Regnier 2003**].

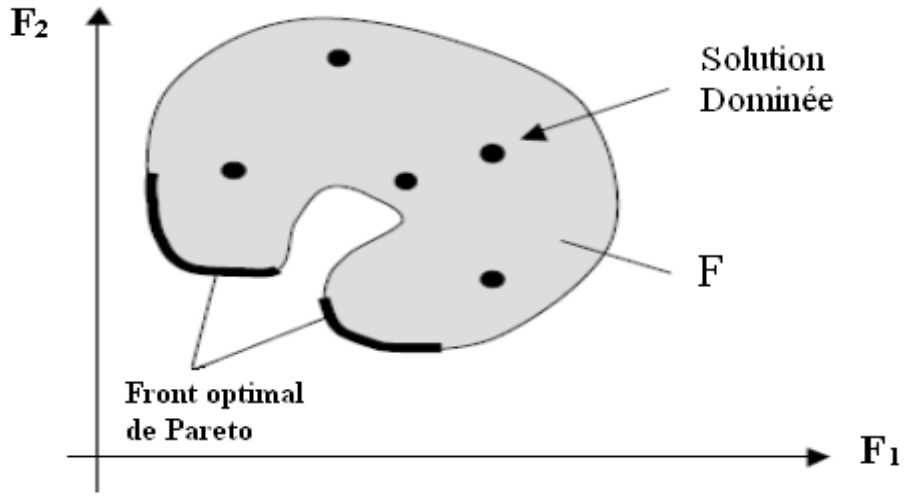


Figure II.2 : Optimalité globale au sens de Pareto

L'ensemble P des solutions Pareto-optimales globales regroupe tous les meilleurs compromis possibles au problème multi-objectif. Comme signalé précédemment, cet ensemble est aussi qualifié de front optimal de Pareto, ou plus généralement de surface optimale de Pareto, pour des problèmes de dimension plus élevée [**Regnier 2003**].

Optimum local au sens de Pareto : une solution **X** est dite localement optimale au sens de Pareto s'il n'existe, dans le voisinage de **X**, aucune solution **Y** dominant **X**. Le voisinage d'une solution peut être défini comme une restriction F_0 de l'espace complet F des objectifs. La figure II.3 illustre ce type d'optimalité [**Regnier 2003**].

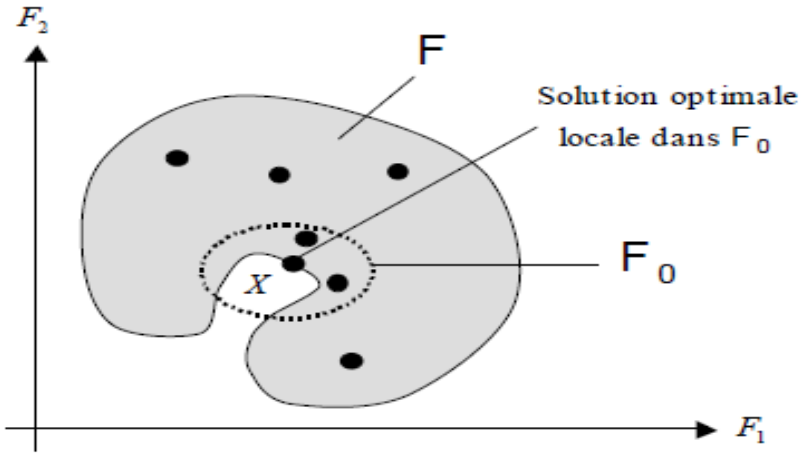


Figure II.3 : Optimalité locale au sens de Pareto

II.3 Approches de résolution d'un problème multi-objectif

Plusieurs méthodes ont été proposées pour le traitement des problèmes multi-objectifs. Ces méthodes peuvent être classées principalement en deux catégories: La première catégorie enveloppe les méthodes "agrégatives", qui transforment le problème en un problème uni-objectif utilisant une fonction objectif équivalente. La deuxième famille comporte les méthodes dites "non agrégatives", dans ces méthodes il n'y a pas fusion d'objectifs pour se ramener à un problème d'optimisation mono-objectif [**Guenounou 2009**].

II.3.1 Méthode agrégative

C'est l'une des premières méthodes utilisée pour résoudre les problèmes d'optimisation multi-objectifs (MO). Elle consiste à transformer le problème MO en un problème mono-objectif en combinant les composantes *fi* du vecteur objectif du problème en une seule fonction scalaire *f*. Il existe dans la pratique, différentes façons de construire la fonction *f*. La plus classique et la plus utilisée se ramène à une simple somme pondérée des objectifs *fi* (agrégation additive):

$$\sum_{i=1}^{n} \mathrm{Wi} \times f\mathrm{i}$$

Où les paramètres Wi sont les poids de pondération. La figure II.4 illustre l'interprétation géométrique de la méthode de pondération dans le cas d'un problème à deux objectifs. La solution Pareto optimale est le point où l'hyper-plan possède une tangente commune avec l'espace réalisable. La méthode de pondération est relativement simple à utiliser mais il est assez difficile de fixer a priori les valeurs des poids associés à chaque objectif. Il est par ailleurs impératif de normaliser les objectifs lorsque ceux-ci sont non-commensurables, ce qui est souvent le cas. Ce problème de mise à l'échelle s'avère complexe car les valeurs de normalisation ne sont pas faciles à déterminer [**Guenounou 2009**].

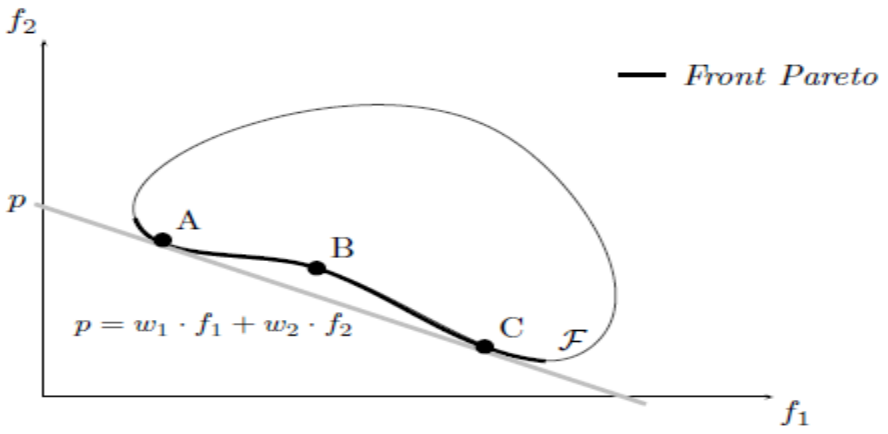


Figure II.4 Interprétation graphique de l'approche par pondération.

II.3.2 Méthode non agrégative

Ces méthodes utilisent directement la notion d'optimalité au sens de Pareto. Elles visent à atteindre deux buts, d'une part converger vers le front de Pareto optimal et d'autre part, obtenir des solutions diversifiées, c'est-à-dire réparties uniformément sur le front.

La difficulté provient du fait qu'un algorithme d'optimisation multi-objectif a lui-même plusieurs objectifs à atteindre.

Comme nous l'avons signalé précédemment, la méthode agrégative peut être utilisée de façon séquentielle pour obtenir le front de Pareto pour un problème d'optimisations multi-objectif. Toutefois, cette approche n'est généralement pas satisfaisante car le nombre d'exécutions successives nécessaires pour déterminer les différents compromis conduit alors à un nombre d'évaluations de critères prohibitif. Ainsi, pour surmonter cette difficulté, on préfère utiliser des méthodes permettant d'une part de trouver l'ensemble de solutions Pareto optimales en une seule exécution et d'autre part de s'affranchir des problèmes de mise à l'échelle des objectifs. Parmi ces méthodes on peut citer :

- Vector Evaluated Genetic Algorithm (VEGA).
- Multiple Objective Genetic Algorithm (MOGA).
- Niched Pareto genetic algorithm (NPGA).
- Nondominated sorting genetic algorithm II (NSGAII) [**Guenounou 2009**].

III. Evolution artificielle

Les algorithmes évolutionnaires (AEs) sont des méthodes stochastiques qui simulent le processus de l'évolution naturelle dans la résolution des problèmes d'optimisation [**Zitzler et al, 2003**]. Ils reposent sur l'analogie avec l'un des principes Darwiniens les plus connus : la survie de l'individu le mieux adapté [**Lepadatu 2006**]. Ils sont basés sur la notion de « population d'individus », dans laquelle chaque individu représente une solution potentielle de l'espace de recherche. Ces algorithmes permettent de surmonter certains problèmes rencontrés avec les méthodes classiques. Ce sont des méthodes d'optimisation globales qui assurent la convergence vers les optima globaux malgré la présence d'optima locaux, et indépendamment de la répartition de la population initiale. Ils peuvent également être utilisés pour des problèmes discrets et discontinus. En outre, en tant qu'algorithme à base de population, leur parallélisassions est aisée : il suffit de distribuer l'évaluation des fonctions coût sur autant de processeurs qu'il y a d'individus dans la population. Ils sont fréquemment utilisés à cause de

leur robustesse et de leur souplesse, qui leurs permettent d'aborder les problèmes les plus raides.

Historiquement, les algorithmes évolutionnaires sont utilisés depuis les années soixante. Ils se divisent en quatre catégories principales, voir la Figure II.5 [**Do 2006**].

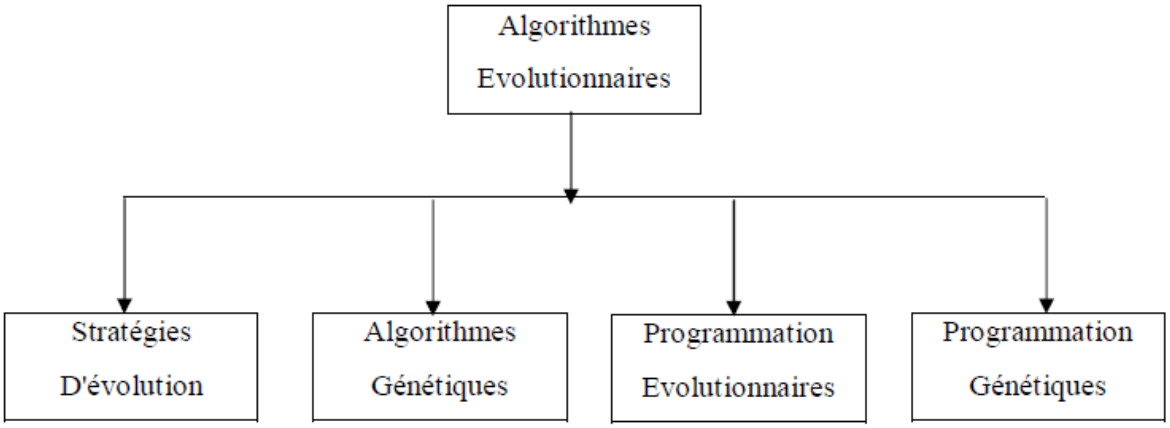


Figure II.5 – Principales catégories d'algorithmes évolutionnaires.

Dans le reste de cette section, on présente la description du principe de fonctionnement d'un algorithme génétiques et ces stratégies d'évolution.

III.1 Principe d'un algorithme génétique standard

Les algorithmes génétiques (AGs) [**Ejday 2011**] ont été mis au point par John Holland de l'université du Michigan aux États-Unis dans les années 60. Ils ont ensuite été raffinés par De Jong et popularisés par Goldberg et Goldberg et Holland. C'est les plus connus des algorithmes évolutionnaires; ils favorisent l'utilisation du croisement comme principal opérateur de recherche. Il utilise cependant la mutation avec un faible pourcentage de probabilité, et une méthode de sélection de type probabiliste dans laquelle la probabilité de sélection est proportionnelle à la fonction d'adaptation de l'individu. Suivant, le type de codage utilisé, on distingue deux types de représentation des individus : codage binaire et le codage réal qui est le plus utilisé.

Le processus de l'évolution génétique est le suivant :

III.1.1 Initialisation de la population

Le choix d'une population de N individus $P(0) = \{X_1, \ldots X_n\}$ se fait, en général, par des tirages uniformes dans l'espace de recherche E en veillant éventuellement à ce que les individus produits respectent les contraintes. Par ailleurs, si on dispose d'informations a priori sur le problème indiquant une région où l'on est sur de trouver l'optimum, il parait bien

évidemment naturel d'ajouter manuellement de bonnes solutions dans la population initiale, tout en assurant une diversité suffisante de la population. La population initiale peut aussi être le résultat d'une évolution précédente [**Ben Abdallah 2005**].

III.1.2 Sélection

La sélection est un opérateur essentiel pour améliorer la qualité d'une population. Son objectif est de retenir et multiplier les meilleurs individus qui contribueront à l'opération de croisement, et d'éliminer les plus mauvais.

Indépendamment du codage utilisé, la littérature propose différents types de sélection, comme celle par tournoi, proportionnelle (roulette) ou par rang. Pour plus de détails, le lecteur peut consulter Deb [**Deb 2001**].

III.1.3 Croisement

Le croisement est l'opérateur principal des algorithmes génétiques (AGs). Son rôle consiste à choisir aléatoirement deux individus parents parmi ceux sélectionnés pour les combiner et créer deux nouveaux individus enfants. Il joue un rôle important sur la convergence de l'AG, en lui permettant de concentrer une partie de la population autour des meilleurs individus. À chaque type de codage correspondent différents types de croisement. Pour les AGs avec un codage binaire, on utilise le croisement à un point (chaque bit caractérisant le nouvel individu est celui de l'un de ses « parents » choisi aléatoirement), à multipoints et le croisement uniforme. Selon Deb [**Deb 2001**], les AGs avec un codage réel peuvent utiliser la méthode de croisement linéaire (la valeur de l'individu enfant est une combinaison linéaire de celle de ses parents), naïf, mélangé ou encore binaire simulé.

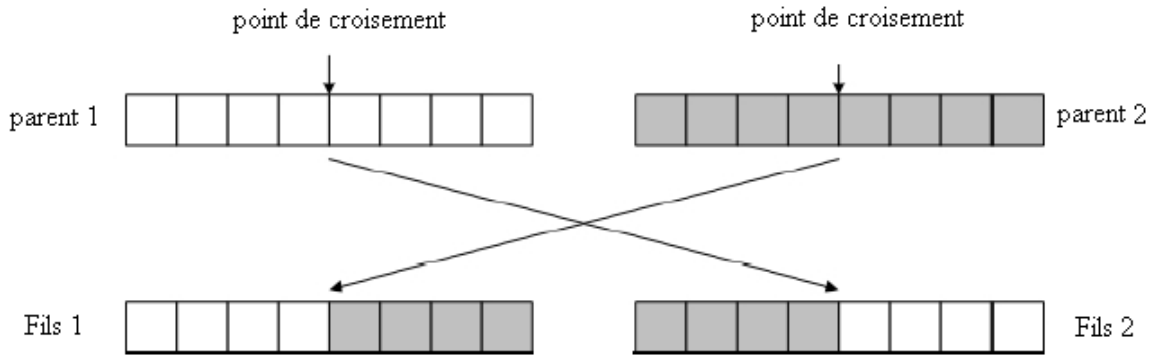


Figure II.6 : l'opération de Croisement

III.1.4 Mutation

Les AGs utilisent l'opérateur de mutation pour conserver la diversité de la population et évite en particulier à l'AG de converger vers un optimum local [**Gildemyn 2008**]. Pour les AGs avec un codage binaire, elle consiste à changer la valeur d'un ou plusieurs bits de l'individu de la population parent d'une valeur 1 à la valeur 0, ou réciproquement, avec une probabilité de mutation p_m fixée. Concernant les AGs avec un codage réel, il existe différentes méthodes de mutation telles que la mutation aléatoire, la mutation non uniforme, la mutation distribuée normalement et la mutation polynomiale [**Deb 2001**].

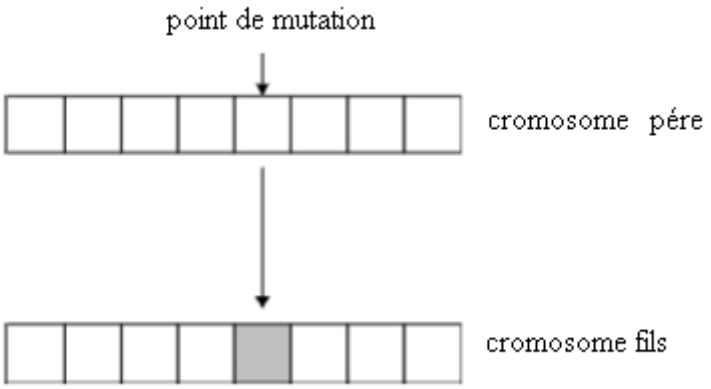


Figure II.7 : l'opération de mutation

III.1.5 Remplacement

Le remplacement consiste à déterminer quels sont les individus de la génération courante, constituée de parents et d'enfants, qui seront les parents de la génération suivante. Plusieurs stratégies de remplacement peuvent être utilisées la plus connus est :

Remplacement élitiste

L'opérateur d'élitisme [**Deb 2001**] permet de conserver les tout meilleurs individus de la population courante. Il est essentiel pour l'optimisation multi-objective. L'élitisme peut être utilisé de manière locale, lorsque deux enfants viennent d'être créés : ils sont comparés à leurs parents. Parmi les quatre individus parents et enfants, seuls les deux meilleurs sont conservés ; les parents sont en concurrence directe avec leurs enfants. Il peut également être utilisé d'une manière globale : une fois la population enfant créée, elle est combinée avec la population parent, et seuls les meilleurs individus sont retenus pour former la nouvelle population. Ainsi, une bonne solution trouvée très tôt dans l'optimisation n'est pas perdue même si une autre meilleure solution est trouvée. Donc l'élitisme accroît la probabilité de créer des meilleurs enfants et d'assurer une convergence rapide vers la solution optimale.

IV. Algorithme génétique élitiste de tri non-dominé (NSGA-II)

Dans cette partie, on présente l'Algorithme Génétique élitiste de Tri Non- Dominé (Elitist Non-Dominated Sorting Genetic Algorithm), NSGA-II, qui sera utilisé dans la suite de ce mémoire. NSGA-II, a été développé par Kalyanmoy Deb et ses étudiants à partir de 2000. Il est également décrit dans [**Deb 2001**] et [**Goal et al, 2007**], et apparaît comme l'un des algorithmes de référence pour trouver l'ensemble optimal de Pareto avec une excellente variété de solutions. Il est certainement le plus populaire algorithme génétique multi-objectif [**Justesen 2009**]. Il est basé sur les trois caractéristiques suivantes : il utilise le principe de l'élitisme, il favorise les solutions non dominées, et il utilise une variété explicite des solutions, grâce au critère de distance d'encombrement (Distance de crowding).

Les principales caractéristiques de cet algorithme sont évoquées par la suite. Nous développons notamment la méthode de classification des individus, la définition de la distance de *crowding*, la description de la méthode de sélection et la structure de l'algorithme.

IV.1 Classification des individus

Dans un premier temps [**Abdelli 2007**], avant de procéder à la sélection, on affecte à chaque individu de la population un rang correspondant au front auquel il appartient. Tous les individus de la population non-dominés appartenant au front optimal (1er front) reçoivent un rang égal à 1. Ces individus sont retirés de la population et l'opération est répétée ainsi de suite pour les fronts successifs suivants en incrémentant le rang.

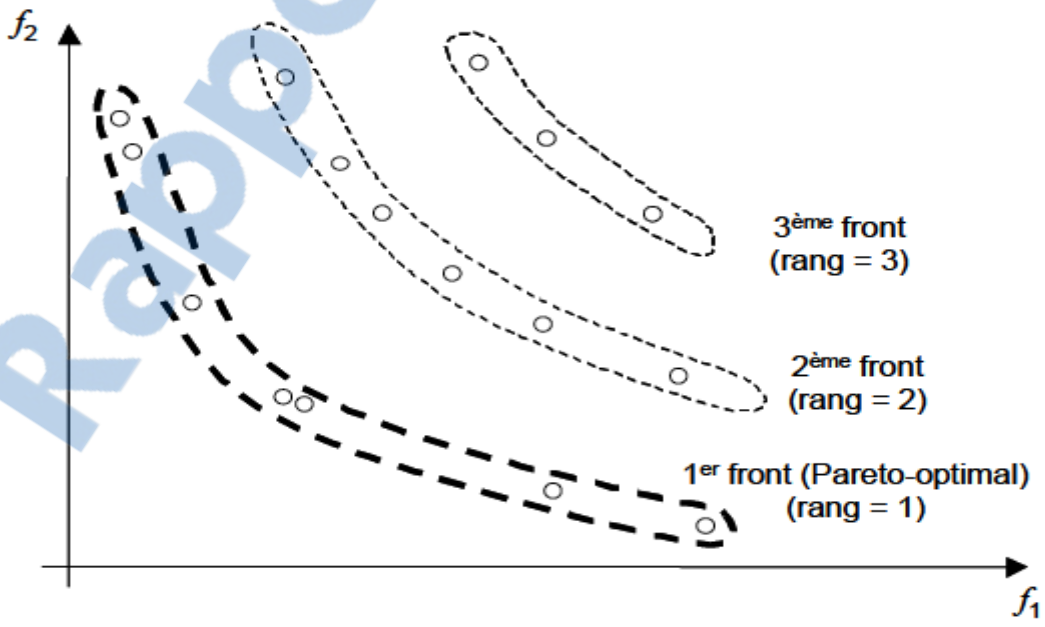


Figure II.8 : Classification des individus selon le rang de Pareto

IV.2 Distance de crowding

La distance de *crowding* d'une solution *i* (ou d'un individu) se calcule en fonction du périmètre formé par les solutions du même front les plus proches de *i* sur chaque objectif. La figure II.9 montre une présentation à deux dimensions associée à la solution *i*. Le calcul de la distance de *crowding* nécessite avant tout le tri des solutions selon chaque objectif, dans un ordre ascendant. Ensuite, pour chaque objectif, les individus possédant les valeurs limites (la plus petite et la plus grande valeur de fonction objective) se voient associés à une distance infinie (∞). Pour les autres solutions intermédiaires, on calcule une distance de *crowding* égale à la différence normalisée des valeurs de fonctions objectives de deux solutions adjacentes. Ce calcul est réalisé pour chaque fonction objective. La distance de *crowding* d'une solution est calculée en sommant les distances correspondantes à chaque objectif.
[**Abdelli 2007**]

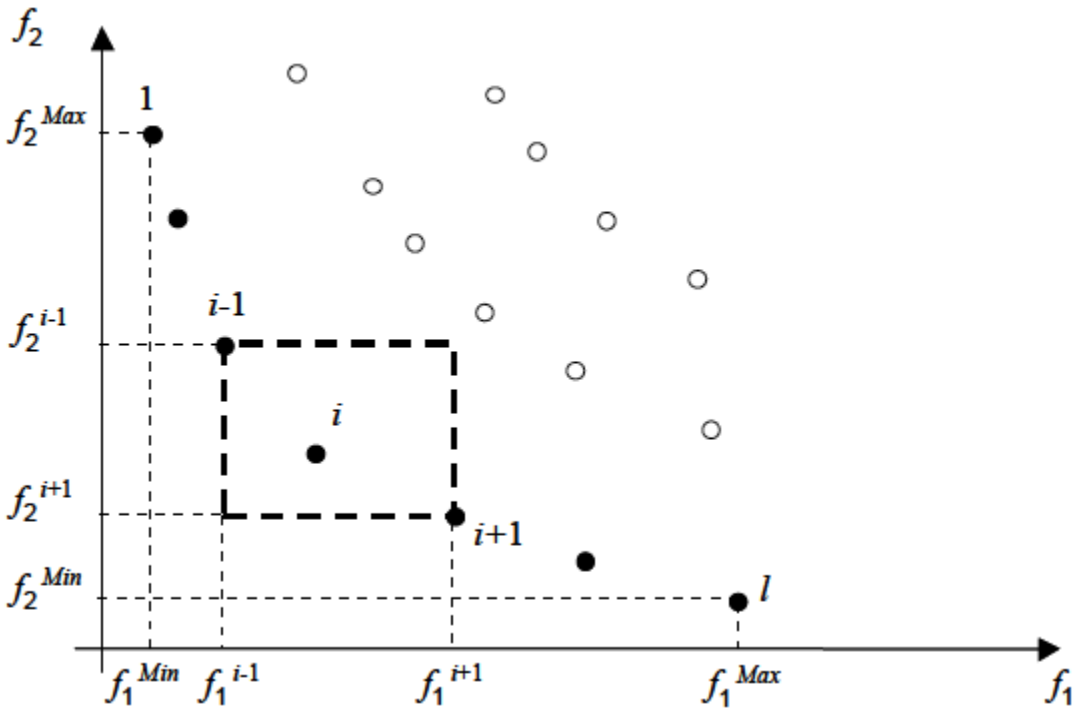


Figure II.9 : Distance de *crowding (Distance de surpeplement)*

IV.3Opérateur de sélection

La sélection est basée sur un opérateur de comparaison (crowded-comparison) qui exploite les notions de rang de Pareto et de distance de *crowding* présentées précédemment. Cet opérateur est utilisé pour guider le processus de sélection vers les solutions Pareto-optimales. Chaque solution (*i*) de la population est identifiée par son rang (i_{rang}) et par sa distance de *crowding*

($i_{distance}$). L'opérateur défini ci-dessous permet d'identifier un ordre de préférence entre deux solutions *i* et *j* :

Entre deux solutions de rangs différents, on préfère la solution de plus petit rang (c'est-à dire appartenant au meilleur front). Pour deux solutions qui appartiennent au même front, on préfère la solution qui est localisée dans la région où la densité de solutions est moindre, soit l'individu possédant la plus grande valeur de distance de *crowding*. Cet opérateur de sélection intensifie donc la recherche des solutions Pareto-optimales mais préserve aussi la diversité parmi des solutions équivalente.

IV.4 Etapes du NSGA-II

Dans cet algorithme [**Deb 2001**], une population de parents (P_t) de taille (N) et une population (Q_t) de taille (N) sont assemblées pour former une population $R = (P \cup Q)$, comme le montre la figure II.10. Cet assemblage permet d'assurer l'élitisme. La population de taille ($2N$) est ensuite triée selon un critère de non-dominance pour identifier les différents fronts F_1, F_2, etc. Les meilleurs individus vont se retrouver dans le ou les premiers fronts. Une nouvelle population parent (P_{t+1}) est formée en ajoutant les fronts (premier front F_1, etc.) tant que le nombre d'individus ajoutés ne dépasse pas la taille limite de l'archive N. Si le nombre d'individus présents dans (P_{t+1}) est inférieur à (N), un tri selon la distance de crowding est appliqué au premier front (F_i) non inclus dans (P_{t+1}) Ce tri a pour objectif d'insérer les (N-$|P_{t+1}|$) meilleurs individus qui manquent dans la population (P_{t+1}). L'insertion des individus du front F_i se fait par troncation, relativement au tri effectué pour maintenir un maximum de diversité dans la population.

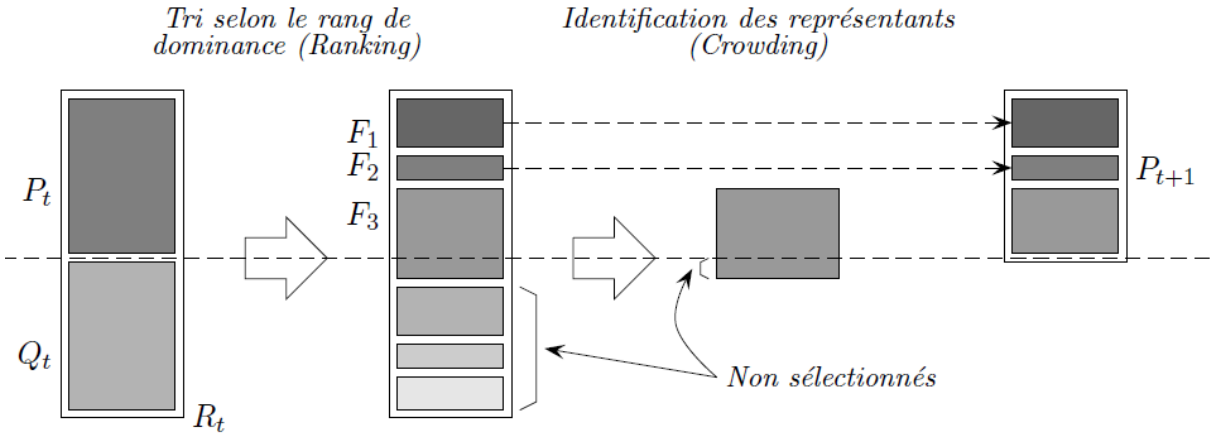


Figure II.10 Fonctionnement de NSGA-II [Deb, 2001].

Une fois que les individus appartenant à la population (P_{t+1}) sont identifiés, une nouvelle population enfant (Q_{t+1}) est crée par croisement et mutation. Le processus continu d'une génération à la suivante, jusqu'à ce que un critère d'arrêt soit vérifié.

Pseudo-code de l'algorithme NSGA-II

Initialiser les populations P0 et Q0 de taille N

Tant que critère d'arrêt non rencontré faire

- Création de Rt = Pt ∪ Qt

- Calcul des différents fronts Fi de la population Rt par un algorithme de « ranking »

- Mettre $P_{t+1} = \varnothing$; et i = 0,

- Tant que |Pt+1|+ |Fi| < N faire

Pt+1 = Pt ∪ Fi

i = i + 1

Fin Tant Que

- Inclure dans Pt+1 les (N - |P_{t+1}|) individus de Fi les mieux répartis au sens de la « distance de crowding »

- Sélection dans P_{t+1} et création de Q_{t+1} par application des opérateurs de croisement et mutation

Fin Tant Que

V. Conclusion

Dans ce chapitre, nous avons présenté les concepts de l'optimisation multi-objectifs, puis les algorithmes évolutionnaires (AEs) comme étant les meilleurs algorithmes de résolution de problèmes d'optimisation multi-objectifs. Ensuite, on a décrit l'algorithme génétique NSGA-II, qui est utilisé dans ce mémoire comme algorithme de référence pour les problèmes d'optimisation multi-objectifs.

CHAPITRE III

Conception et implémentation du prototype

I. Introduction

Avec la prolifération des services web [**Ben Halima 2009**], la notion de qualité de service (QoWS) émerge aujourd'hui et prend de plus en plus une grande importance pour les fournisseurs de service aussi bien que pour les clients de service. En outre, comme les services Web avec des fonctionnalités similaires devraient être fournis par des fournisseurs concurrents, un défi majeur est de concevoir des stratégies d'optimisation pour trouver les meilleurs services Web en fonction de qualité de services.
Etant donné qu'il y a plusieurs services Web publiés sur Internet, beaucoup d'entre eux, ne peuvent satisfaire les différents besoins d'un utilisateur. Il sera donc nécessaire, d'ordonner et de classer plusieurs services Web en se basent sur différents critères de qualité.

En effet, en disposant d'une bibliothèque de services Web similaires, nous pouvons faire une sélection à la main, en faisant l'écoute sur les services les plus populaire, ou guidée par un outil d'aide à la décision. La sélection des services Web a pour objectif de déterminer le service le plus adéquat.

II. La Qualité de Service

Jusqu'aujourd'hui, il n'existe pas de consensus sur la définition de la qualité de service (QoS). On peut la définit comme "*un ensemble d'exigences dans le comportement collectif d'un ou plusieurs objets*". Dans le contexte des technologies de l'information et multimédia, la QoS a été définie par [**Vogel et al, 1995**] comme "*l'ensemble des caractéristiques quantitatives et qualitatives d'un système multimédia, nécessaires pour atteindre la fonctionnalité requise par l'application*". Nous pouvons aussi dire que la qualité de service représente l'aptitude d'un service à répondre d'une manière adéquate à des exigences, exprimées ou implicites, qui visent à satisfaire ses usagers. Ces exigences peuvent être liées à plusieurs aspects d'un service, par exemple :

- Le débit : Le nombre de requêtes servies pendant un intervalle de temps.
- Le temps de réponse : Le temps requis pour compléter une requête du service web.
- La fiabilité : La capacité d'un service d'exécuter correctement ses fonctions.
- La scalabilité : La capacité du service de traiter le plus grand nombre d'opérations ou de transactions pendant une période donnée tout en gardant les mêmes performances.
- La robustesse : La probabilité qu'un service peut réagir proprement à des messages d'entrée invalides, incomplètes ou conflictuelles.

- La disponibilité : La probabilité d'accessibilité d'un service.
- Le prix d'exécution : C'est le prix qu'un client du service doit payer pour bénéficier du service.
- La réputation : C'est une mesure de la crédibilité du service. Elle dépend principalement des expériences d'utilisateurs finals.
- La sécurité : C'est un regroupement d'un ensemble de qualités à savoir : la confidentialité, le cryptage des messages et le contrôle d'accès.

III. Modèles de QoWS existants

Le groupe de travail *Architecture des Services Web* du W3C travaillant sur les architectures des services web, a identifié et décrit un ensemble de paramètres de QoWS pour les services web [**Kangchan Lee et al, 2003**] (QoWS) , à savoir : la performance (qui englobe le débit, le temps de réponse et le temps d'exécution), la fiabilité, la scalabilité ou l'adaptation au facteur d'échelle, la capacité, la robustesse, le traitement d'exception, l'exactitude, l'intégrité, l'accessibilité, la disponibilité, l'interopérabilité et la sécurité.

Il n'y a pas un consensus bien précis au sujet de l'ensemble des QoWS importantes pour les services web, mais la plupart des travaux de recherche qui ont essayé d'identifier et de classifier les paramètres de QoWS ont pris en considération les paramètres définis par le W3C aux quels sont associés dans certains travaux d'autres paramètres.

IV. QoWS considérées

Dans ce qui suit nous nous focalisons sur les attributs de QoWS que nous gérons dans notre travail : [**Menascé 2002**] [**El Saddik 2006**]

IV.1 Coût

- *Défini comme:* Le montant (selon une certaine devise) pour exécuter l'opération.

IV.2 Le Temps de latence :

- *Défini comme* : Le temps nécessaire pour traiter une requête dès l'instant de son envoi jusqu'au moment de la réception de la réponse.
- *Formule latence* = Le temps de réception de la réponse par le client - Le temps d'envoi de la requête par le client.
- *Quantification :* seconde.

IV.3 La Disponibilité :

- *Défini comme* Le pourcentage de requêtes réussites par le fournisseur. Les réponses échouées correspondent aux exceptions reçues du côté client.
- *Formule Disponibilité* = Nombre de requêtes réussites / Nombre total de requêtes
- *Quantification* : Pourcentage.

IV.4 Fiabilité

- *Défini comme:* La fiabilité des opérations est la capacité de l'opération doit être exécutée dans le délai prévu au maximum.
- *Formule fiabilité* Nsuccess(op)/Ninvoked(op),
  où Nsuccess(op) est le nombre de fois que op a été exécutée avec succès et Ninvoked(op) est le nombre total d'appels
- *Quantification* Pourcentage

IV.5 Réputation

- *Défini comme* La réputation de l'opération est une mesure de la fiabilité de l'opération. Elle dépend principalement du rapport à laquelle la fourniture effective du service est conforme à sa promise. Les qualités fraîches peuvent être obtenues sur la base sur la publicité des fournisseurs de services dans la description du service alors que la réputation est basée sur le classement des utilisateurs finaux.

V. Scénario :

Nous considérons une agence de voyages [**Bouguettaya et al, 2009**], nommé travelagency, fournissant l'organisation de voyages (par exemple, le transport, l'itinéraire, et l'hébergement) pour ses clients (figure III.1). Supposons qu'un professeur d'université, Fethallah, veut assister à une conférence internationale à Sydney, en Australie. Les services typiques nécessaires par Fethallah pourraient inclure les compagnies aériennes, au sol de transport (par exemple, de taxi et de location de voitures), services de divertissement d'hébergement (par exemple, des hôtels), et d'autres (par exemple, un restaurant et de l'opéra). Fethallah a besoin à la recherche d'abord pour les services qui fournissent des paquets de voyage. La recherche peut être effectuée dans certains registres de services bien connus.

Cet exemple exprime un scénario typique de sélection de service Web dans le domaine de tourisme

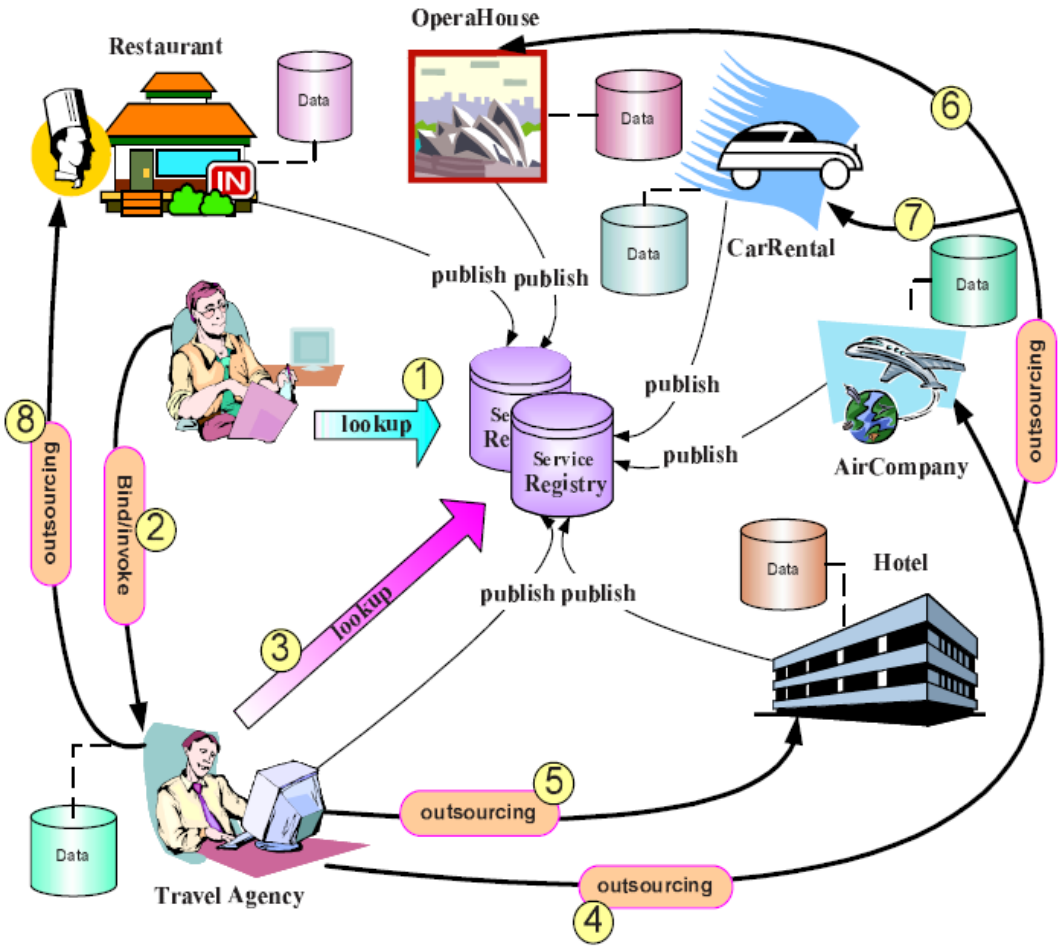


Figure III.1 Un scénario de préparation de voyages

VI. Présentation de la base

Notre base d'exemples comporte un fichier XML qui contient dix classes de services Web, et quarante provider. Chaque provider est caractérisé par cinq critères de QoWS.

Les données du fichier sont générées aléatoirement d'une façon où la latence est prise dans l'intervalle comprit entre 0 et 300 seconde, la fiabilité entre 0.5 et 1, la disponibilité entre 0.7 et 1, le coût entre 0 et 30$, et la réputation entre 0 et 5.

Notre objectif est de trouver une combinaison qui maximise la fiabilité, la réputation et la disponibilité ainsi qui minimise le coût et le temps de latence.

VII. Conception

VII.1 Diagramme de cas d'utilisation

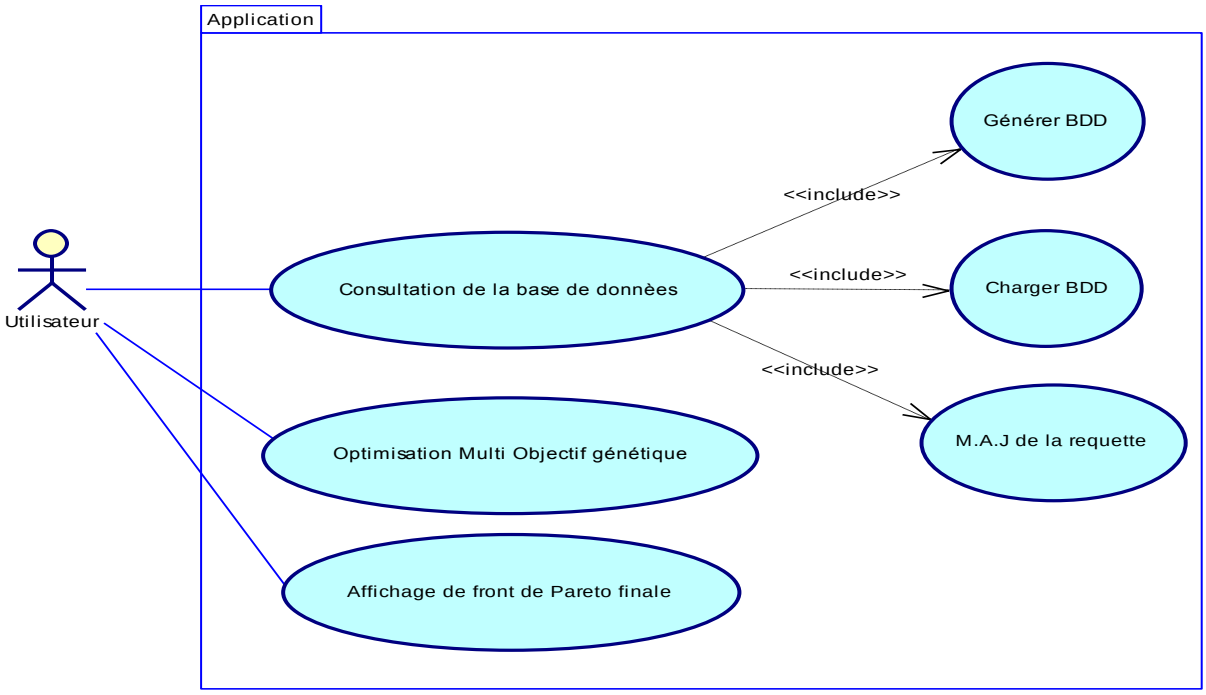


Figure III.2 Diagramme de cas d'utilisation

VII.2 Diagramme de classe

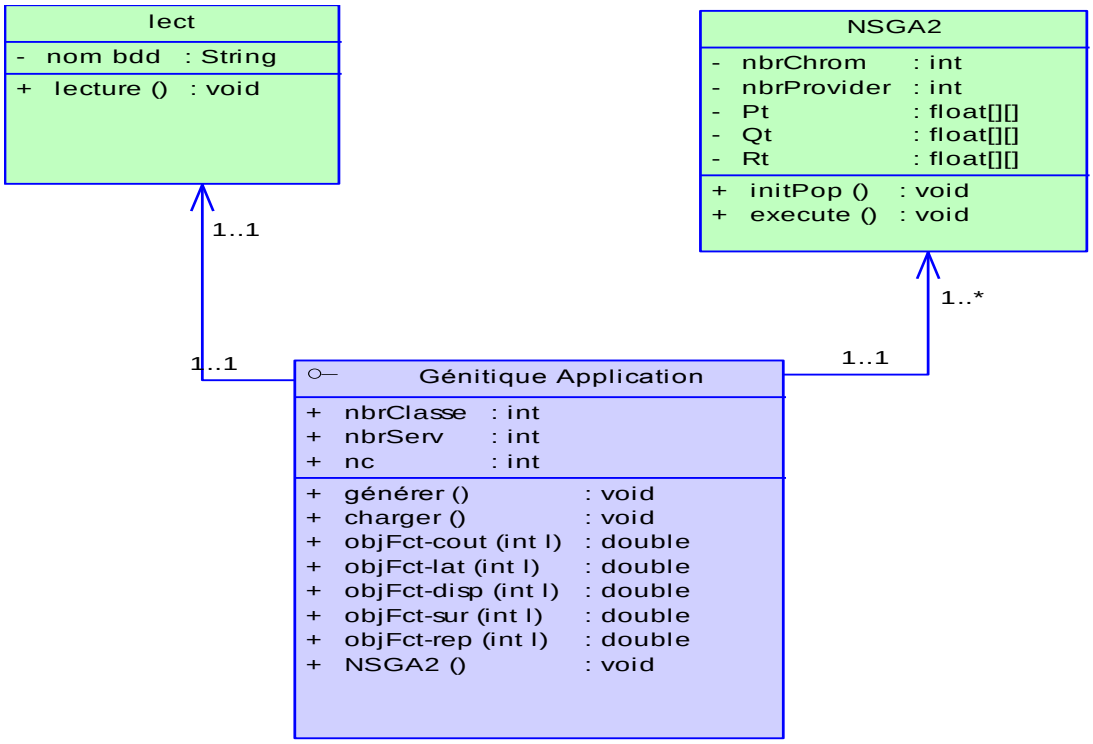


Figure III.3 Diagramme de classe

VIII. Interface homme machine

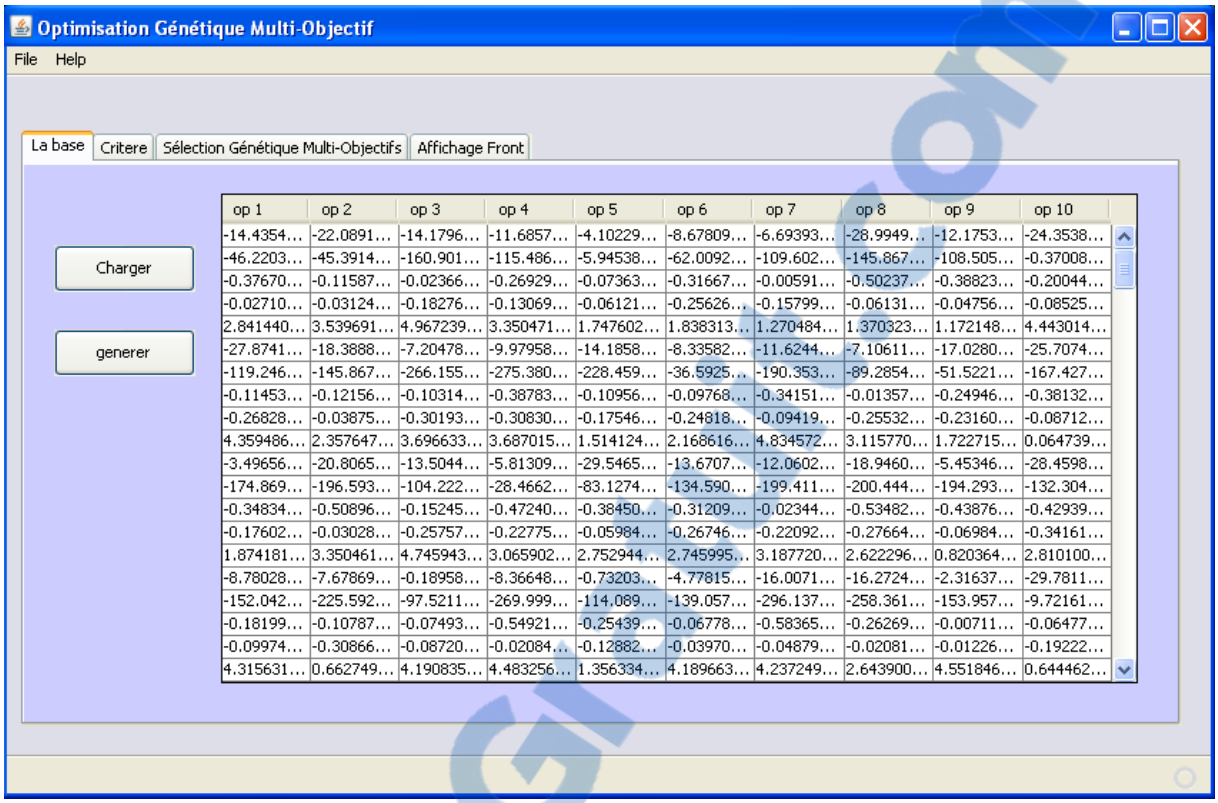


Figure III.4 chargement de la base d'exemples

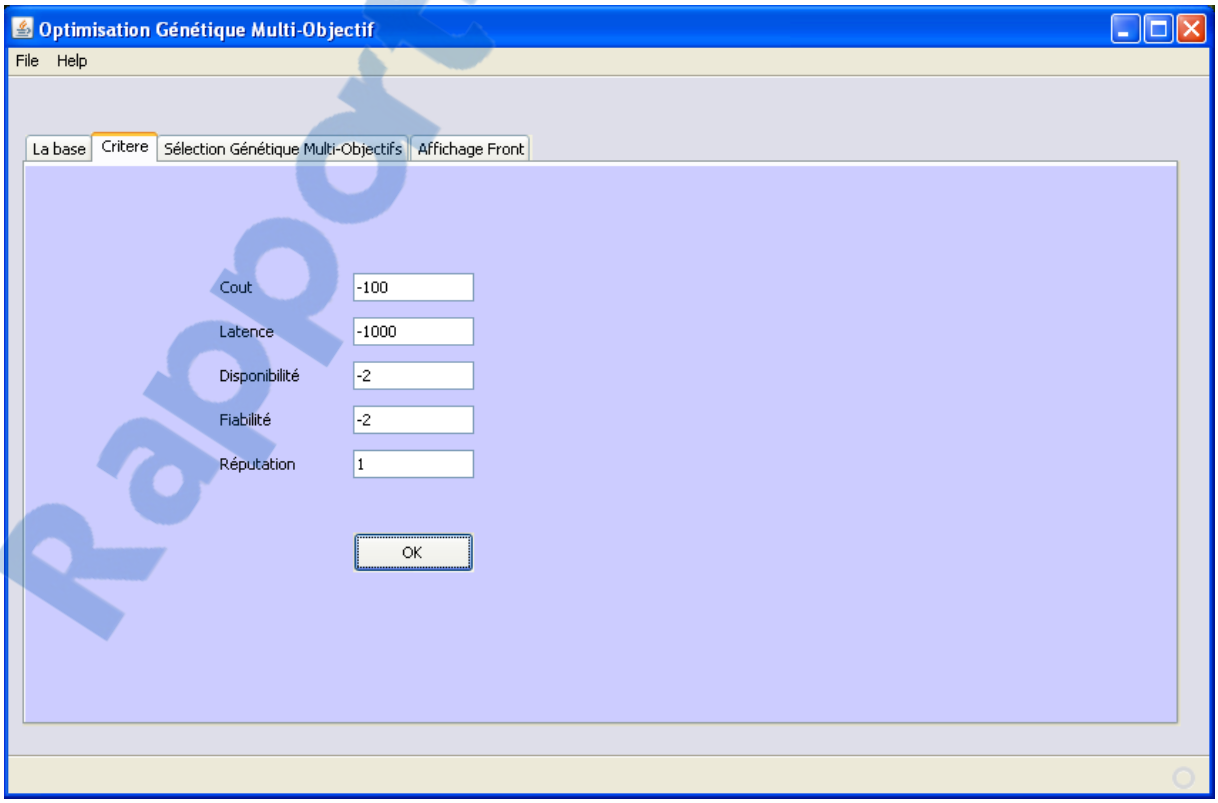


Figure III.5 validation des contraintes

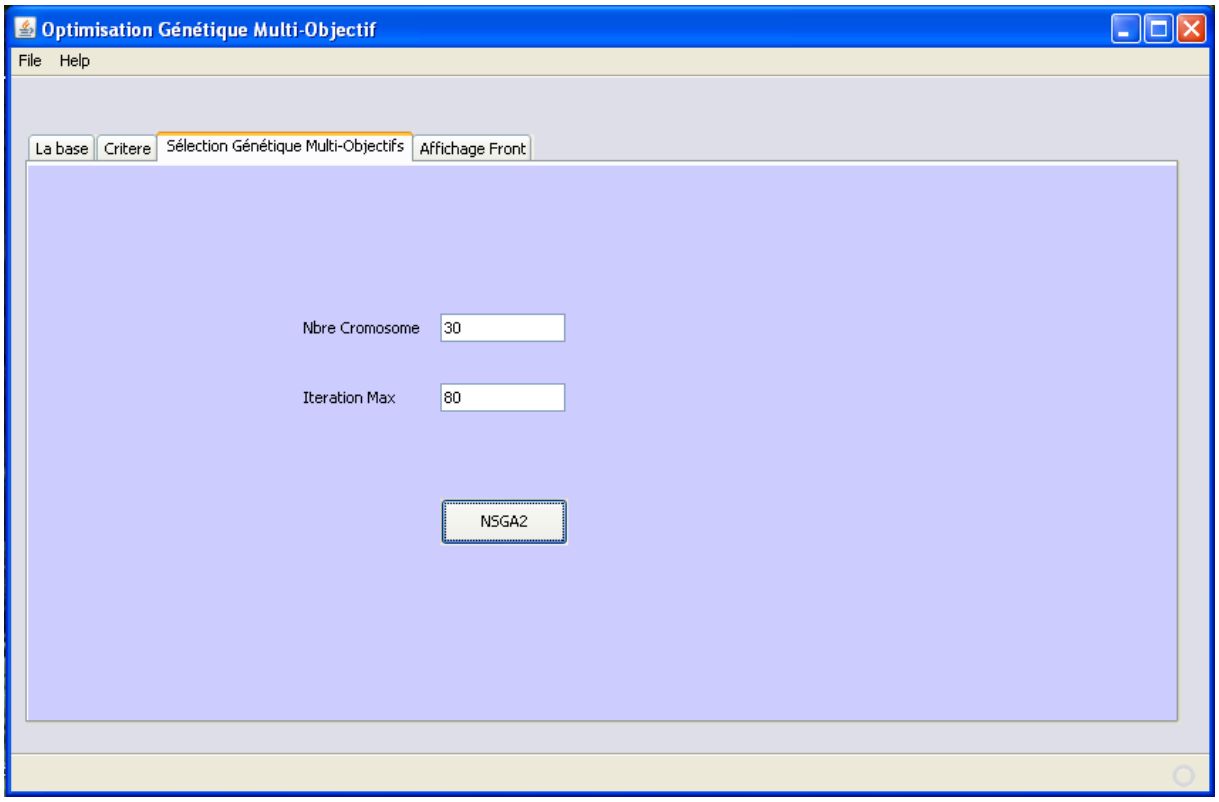
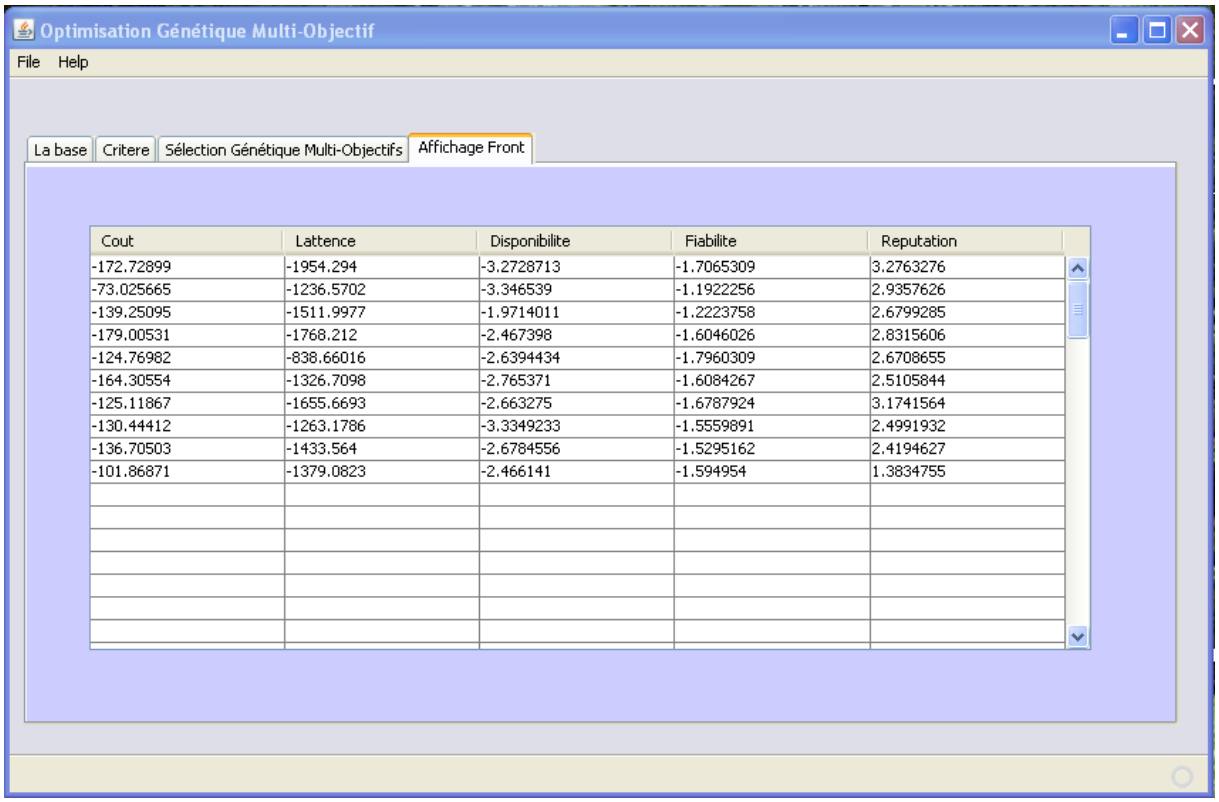


Figure III.6 sélection à base de NSGA II



Cout	Lattence	Disponibilite	Fiabilite	Reputation
-172.72899	-1954.294	-3.2728713	-1.7065309	3.2763276
-73.025665	-1236.5702	-3.346539	-1.1922256	2.9357626
-139.25095	-1511.9977	-1.9714011	-1.2223758	2.6799285
-179.00531	-1768.212	-2.467398	-1.6046026	2.8315606
-124.76982	-838.66016	-2.6394434	-1.7960309	2.6708655
-164.30554	-1326.7098	-2.765371	-1.6084267	2.5105844
-125.11867	-1655.6693	-2.663275	-1.6787924	3.1741564
-130.44412	-1263.1786	-3.3349233	-1.5559891	2.4991932
-136.70503	-1433.564	-2.6784556	-1.5295162	2.4194627
-101.86871	-1379.0823	-2.466141	-1.594954	1.3834755

Figure III.7 affichage du front de Pareto final

IX. Expérimentation

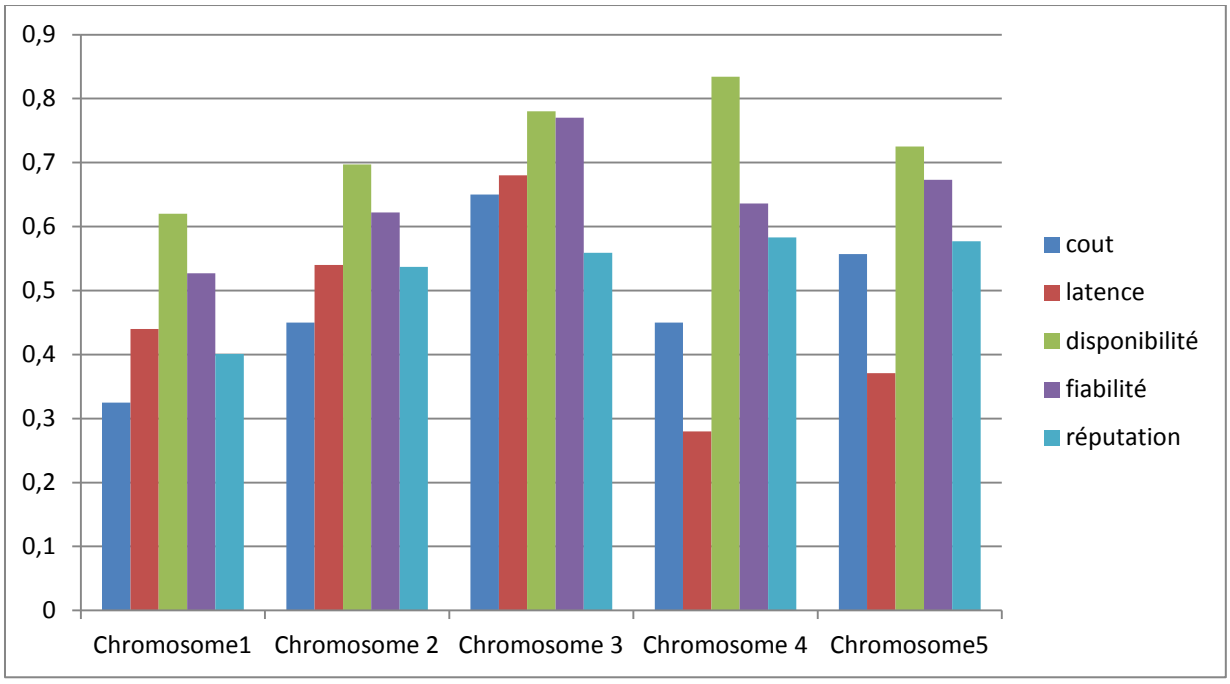


Figure III.8 Histogramme d'optimalité

- **Discussion**

On constate que l'algorithme NSGA II arrive toujours à donné un front de Pareto acceptable en termes d'optimalité, mais si la taille de la population dépasse un certain seuil alors le temps de réponse ne sera plus raisonnable.

X. Conclusion

Nous avons présenté dans ce chapitre un algorithme d'optimisation multi objectif pour la sélection de services web, les résultats obtenus confirment l'efficacité du NSGA II dans ce genre de problèmes. Et peuvent être améliorés avec de nouvelles heuristiques.

Conclusion Générale & Perspectives

Conclusion Générale et Perspectives

Nous avons présenté dans ce mémoire les technologies liées aux services Web. Nous avons proposé aussi un algorithme de sélection qui se base sur les algorithmes génétiques multi-objectifs pour l'optimisation des compositions.

Notre prototype adopte l'algorithme NSGA II pour sélectionner les compositions de services les plus satisfaisantes,

Les compositions sont caractérisées par un groupe de sur 05 critères de QOS : Latence, fiabilité, disponibilité, coût, réputation.

Comme perspective à ce travail, nous proposons d'implémenter d'autres algorithmes de sélection, pour cela, nous pouvons citer :

- Les colonies de fourmis.
- Les colonies d'abeilles.
- la programmation par contraintes.
- la recherche harmonique.

...

Références Bibliographiques

[**Abdelli 2007**] Abdenour Abdelli, Optimisation multicritère d'une chaîne éolienne passive, l'institut national polytechnique de TOULOUSE, 15 octobre 2007.

[**Albertella 2009**] D. Albertella, Les avantages compétitifs de SOA : aspects techniques, organisationnels et financiers est-ce que SOA remplit ses promesses ? , Université de Fribourg, Suisse, 2009.

[**Barichard 2003**] Vincent Barichard, Approches hybrides pour les problèmes multi-objectifs, école doctorale d'Angers, 24 Novembre 2003.

[**Bejaoui 2008**] R. Bejaoui, SOA : les impacts sur le cycle de vie du projet de développement logiciel. Maîtrise en informatique de gestion, Université de Québec à Montréal, 2008.

[**Ben Abdallah 2005**] Ahmed Chamseddine Ben Abdallah, Optimisation multi-objectif évolutionnaire, école polytechnique de Tunisie, 2005.

[**Ben Halima 2009**] Riadh Ben Halima, Conception, implantation et expérimentation d'une architecture en bus pour l'auto réparation des applications distribuées à base de services web, l'université de TOULOUSE**,** 14 Mai 2009.

[**Bertino 2004**] A. Bertino, M. Keidl, et E. Kemper, A Framework for Context-aware Adaptable Webservices, EDBT, LNCS 2992, Springer-Verlag Berlin Heidelberg, 2004.

[**Bieberstein 2008**] Norbert Bieberstein, Keith Jones robert, et Tilak Mitra, Executing SOA: a Practical Guide for the Service-Oriented Architect, édition IBM Press, Mai 2008.

[**Bouguettaya et al, 2009**] Athman Bouguettaya, Qi Yu, Foundations for Efficient Web Service Selection, Springer Science and Business Media, LLC 2009.

[**Boukhadra 2011**] Adel Boukhadra, La composition dynamique des services Web sémantiques a base d'alignement des ontologies owl-s, 2011.

[**Brown 2008**] Paul C . Brown, Implementing SOA : Total Architecture in Practice, edition Addison Wesley Professional, April 2008.

[Cerami 2002] Ethan Cerami, Web Services Essentials, édition O'Reilly, Février 2002.

[Chappell 2002] David Chappell, et Tyler JEWELL, Java Web Service, édition O'Reilly, Mars 2002.

[Chinnici 2004] Roberto Chinnici, et Martin Gudgin, Web Services Description Language, édition W3C, le 22 Aout 2004.

[Chouchani 2010] Imad Chouchani, Utilisation d'un algorithme génétique pour la composition de service web, UNIVERSITÉ DU QUÉBEC À MONTRÉAL, mai 2010.

[Deb, 2001] Deb, K., "Multi-Objective Optimisation using Evolutionary Algorithms", Edited by John Wiley & Sons, Chichester, 2001.

[Devaux 2008] C. Devaux, Urbanisation et SOA : Quelques bonnes pratiques pour leur mise en œuvre, Livre Blanc, Aubay, France, 2008.

[Do, 2006] Do, T T., "Optimisation de forme en forgeage 3D", thèse de doctorat, Mines ParisTech, Sophia-Antipolis, 2006.

[Dovey et al., 2005] Dovey, M., Kostadinov, I., Giddy, J., Green, P., Berry, D., Chonan, D., Wang, X. UK Engineering Tasks Force Evaluation of UDDI for UK e-Science. UK Technical Report tel-00388991, version 1 - 27 May 2009

[Ejday 2011] Mohsen Ejday, Optimisation Multi-Objectifs à base de Méta modèle pour les Procédés de Mise en Forme, école nationale supérieure des mines de Paris, 17 mars 2011.

[El Saddik 2006] Abdulmotaleb El Saddik. Performance measurements of web services based applications. IEEE Transactions on Instrumentation and Measurement, October 2006.

[Erl 2008] Thomas Erl, SOA Principles of Service Design, édition prentice hall, juillet 2008.

[Fakhfakh 2006] Kaouthar Fakhfakh, Composition de protocoles métier de services web pour le passage à l'échelle en utilisant les automates, 22 juin 2006.

[Gardien 2002] Georges Gardien, XML des bases de données aux Services Web, édition Dunod, 2002.

[Gildemyn, 2008] Gildemyn, E., "Caractérisation des procédés de fabrication de pièces de sécurité automobile. Optimisation multi-objectifs de la mise en forme", thèse de doctorat, Ecole Nationale Supérieure d'Arts et Métiers, 2008.

[Goal et al, 2007] Goal, T., R. Vaidyanathan, R. T. Haftka, W. Shyy, N. V. Queipo, and K. Tucker, "Response surface approximation of Pareto optimal front in multi-objective optimization", 2007.

[Guenounou 2009] Ouahib Guenounou, Méthodologie de conception de contrôleurs intelligents par l'approche génétique- application à un bioprocédé, l'université de TOULOUSE, 22 avril 2009.

[J. Dréo et al, 2003] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. Méta heuristiques pour l'optimisation difficile. Eyrolles, 2003.

[Jennings 2007] Frank Jennings, Ramesh Loganathan, et Poornachandra Sarang, Approach to Integration; XML, Web services, ESB, and BPEL in real-world SOA projects, édition Packt Publishing Ltd, November 2007.

[Josuttis 2007] Nicolai Josuttis, SOA in Practice, édition O'Reilly, Septembre 2007.

[Justesen, 2009] Justesen, P D., "Multi-objective Optimization using Evolutionary Algorithms", University of Aarhus, Department of Computer Science, Denmark, 2009.

[Kaabi 2008] R. S. Kaabi, Une approche méthodologique pour la modélisation intentionnelle des services et leur opérationnalisation, Thèse de Doctorat de l'Université Paris 1, Sorbonne, 2008.

[Kangchan Lee et al, 2003] Kangchan Lee, Jonghong Jeon, Wonseok Lee, Seong-Ho Jeong, and Sang-Won Park. Qos for web services : Requirements and possible approaches. Technical report, W3C, Web Services Architecture Working Group, November 2003.

[Kreger 2001] Heather Kreger, Web Service Conceptual Architecture, édition IBM Software Group, Mai 2001.

[Kulchenko 2001] Pavel Kulchenko, James Snell, et Doug Tidwell, Programming Web Services with SOAP, édition O'Reilly, Décembre 2001.

[Lepadatu, 2006] Lepadatu, D., "Optimisation Des Procédés De Mise En Forme Par Approche Couplée Plans D'expériences, Éléments Finis Et Surface De Réponse", thèse de doctorat, Institut des Sciences et Techniques de l'Ingénieur d'Angers, 2006.

[Lopez-Velasco 2008] Céline Lopez-Velasco, Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation, pour obtenir le grade de Docteur de l'univesité JOSEPH FOURIER, 18 novembre 2008.

[Margolisand 2007] Ben Margolisand, et Joseph Sharpe, SOA for the Business Developer: Concepts, BPEL, and SCA, édition MC Press, October 2007.

[Melliti 2004] Tarek Melliti, Interopérabilité des Services Web complexes, Thèse de Doctorat, Université Paris IX Dauphine, le 8 Décembre 2004.

[Menascé 2002] Daniel A. Menascé. Qos issues in web services. IEEE Internet Computing, 6(6) :72–75, 2002.

[Newcomere 2004] Eric Newcomere, Understanding Web Services Xml WSDL SOAP and UDDI, edition O'Reilly, 2004.

[Papazoglou 2003] M.P. Papazoglou, Web Services and Business Transactions, World Wide Web Internet and Web Information Systems, edition Kluwer Academic, 2003.

[Ponge 2008] Julien Ponge. Model Based Analysis of Time-aware Web Services Interactions. Thèse de Doctorat de l'Université Blaise Pascal - Clermont-Ferrand II, dans le cadre de l'Ecole Doctorale des Sciences pour l'Ingénieur, France, 2008.

[Regnier 2003] Jérémi Regnier, Conception de systèmes hétérogènes en Génie Électrique par optimisation évolutionnaire multicritère, l'institut national polytechnique de TOULOUSE, 18 Décembre 2003.

[Schreiner 2005] Wolfgang Schreiner, et Schahram tdar, A survey on Web services composition, Int. J. Web and Grid Services, 2005.

[Scott 2002] Scott Short, Construire des Services Web XML, édition Dunod, 2002.

[**Vogel et al, 1995**] Andreas Vogel, Brigitte Kerhervé, Gregor von Bochmann, and Jan Gecsei. Distributed multimedia and qos : A survey. IEEE MultiMedia, 2(2) :10–19, 1995.

[**Zitzler et al, 2003**] Zitzler, E., M. Laumanns, and S. Bleuler, "A Tutorial on Evolutionary Multiobjective Optimization. Metaheuristics for Multiobjective Optimisation", Workshop on Multiple Objective Metaheuristics (MOMH 2002), Springer-Verlag, Berlin, Germany, pp.3-38, 2003.

Annexe

XML : est une recommandation du W3C. C'est un langage de balisage définissant un format universel de représentation des données, permettant de décrire la structure hiérarchique d'un document. Ainsi, un document XML contient à la fois les données et les indications sur le rôle que jouent ces données. Ces indications permettent de déterminer la structure du document. XML est défini pour fonctionner indépendamment des plateformes et des systèmes d'exploitation.

B2B (*Business-to-Business*) : Qualificatif signifiant une interaction entre deux entités d'affaires. Par exemple, nous pouvons utiliser l'expression marketing B2B, qui signifie le marketing d'entreprise à entreprise.

B2C : Le Business to Consumer aussi appelé Business to Customer est le nom donné à l'ensemble d'architectures techniques et logiciels informatiques permettant de mettre en relation des entreprises directement avec les consommateurs : en français, « des entreprises aux particuliers ».

COM *Component Object Model* : est une technique de composants logiciel créée par Microsoft. *COM* est utilisé en programmation pour permettre le dialogue entre programmes. Bien qu'il ait été mis en place sur de nombreuses plates-formes, il est toujours majoritairement utilisé sur Microsoft Windows.

CORBA (*Common Object Request Broker Architecture)* : Architecture et spécifications visant à permettre de créer, publier et gérer des objets, sur un réseau, dans un contexte de programmation distribuée. Cette architecture a déjà été largement utilisée sauf par Microsoft qui avait développé sa propre architecture DCOM.

DCOM (*Distributed Component Object Model*) : Ensemble d'interfaces programmes et de concepts développé par Microsoft qui est en fait l'équivalent de CORBA. Il est utilisé entre autre pour faire des appels RPC (Remote Procedure Call).

W3C (*World Wide Web Consortium)* : Consortium industriel visant à favoriser l'interopérabilité des produits informatique et l'évolution du World Wide Web, par le développement de standards et de spécifications techniques.

OASIS (*Organization for the Advancement of Structured Information Standards)*: OSBL technologique international visant la promotion et l'adoption de standards indépendants des manufacturiers, dans le domaine des technologies de l'information. OASIS tente de rapprocher les acteurs industriels et les groupes de standardisation afin qu'ils s'entendent sur l'utilisation universelle de XML.

Protocole : Ensemble de règles qui sont partagée entre deux ordinateurs afin de communiquer l'un avec l'autre.

HTTP (*Hypertext Transfer Protocol)* : Protocole de communication permettant l'échange de fichiers sur le World Wide Web. HTTP est un protocole d'application développé par le W3C.

SMTP (*Simple Mail Transfer Protocol* (littéralement) : est un protocole de communication utilisé pour transférer le courrier électronique vers les serveurs de messagerie électronique.

FTP File Transfer Protocol : est un protocole de communication destiné à l'échange informatique de fichiers sur un réseau TCP/IP. Il permet, depuis un ordinateur, de copier des fichiers vers un autre ordinateur du réseau, ou encore de supprimer ou de modifier des fichiers sur cet ordinateur. Ce mécanisme de copie est souvent utilisé pour alimenter un site web hébergé chez un tiers.

HTML (*Hypertext Markup Language*) : Ensemble de balises et de symboles insérés dans un fichier informatique visant à être affiché sur un navigateur World Wide Web. Les balises HTML définissent comment un navigateur Web devra afficher les informations contenues dans une page Web. HTML a été développé par le W3C.

API (*Application Programming Interface*) : est une interface fournie par un programme informatique. Elle permet l'interaction des programmes les uns avec les autres, de manière analogue à une interface homme-machine, qui rend possible l'interaction entre un homme et une machine.

Résumé

La sélection du service Web est un élément clé dans l'architecture orientée services, beaucoup de travaux ont été proposées pour résoudre ce problème.

Nous proposons dans ce travail une approche multi-objective basée sur des algorithmes génétiques et plus précisément nous adoptons le NSGA II (Algorithme Génétique de tri non-dominés II) pour l'optimisation de la sélection.

Les résultats expérimentaux, que semer la NSGA II peut obtenir le front de Pareto de manière très efficace.

Abstract

The web service selection is a key element in the service oriented architecture; a lot of works have been proposed to solve this problem.

We propose in this work a multi objective frame-work based on genetic algorithms and more specifically we adopt NSGA II (Non-Dominated Sorting Genetic Algorithm II) for optimizing the selection.

Experimental results, show that NSGA II can get the Pareto front very efficiently.

ملخص

انتقاء خدمات الواب على شبكة الإنترنت أصبح من بين أهم الميادين في عصرنا هذا نظرا لكثرة الخدمات و تطورها، وقد تم اقتراح الكثير من الأعمال من أجل حل هذه المشكلة.

نقترح في هذا العمل مقاربة انتقاء تعتمد على الخوارزميات الوراثية متعددة الأهداف وبشكل أكثر تحديدا نعتمد على الخوارزمية الوراثية للتصنيف المهيمن المسماة (NSGA II).