# SELECTING WEB SERVICES FOR OPTIMAL COMPOSITION

Daniela Barreiro Claro<sup>1,2</sup>, Patrick Albers<sup>1</sup>, Jin-Kao Hao<sup>2</sup>

<sup>1</sup> ESEO 4, rue Merlet de la Boulaye, BP30926 49009 Angers cedex01 – France ; <sup>2</sup>LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01 - France

Abstract: Web services are modular applications that can be described, located and invoked on the Internet. A user request may not only correspond to one specific service, but also to a set of web services. Thus, it is necessary that a composition of services be done in order to obtain the expected result. Nonetheless, many services with the same goal but different characteristics can be discovered. Indeed, it is necessary to find non-functional criteria to distinguish them, in order for a user to be able to choose an optimal solution. In this paper, we propose service quality variables as non-functional criteria in order to make an optimal service composition for a goal. We propose then using multiobjective optimisation techniques to find a set of optimal Pareto solutions from which a user can choose the most interesting tradeoff. Finally, we introduce the environment which allows that this composition process be automatic.

Key words: Service Composition, QoS Model, Multiobjectives Approach.

# **1. INTRODUCTION**

Nowadays many enterprises publish their applications functionalities on the Internet. This new generation of applications allows greater efficiency and availability for business. In fact, more and more applications make functionalities available using a service format.

However there are many services around the web, each one, taken alone, has a limited utility. For example, if a user wants to travel, it is not sufficient to book a flight, but also to care about reserving a hotel, renting a car, getting entertained, and so on. The user needs to execute all these services manually and these tasks can be time and effort consuming.

For that reason, the notion of composite services is starting to be used as a collection of services combined to achieve a particular goal. In other words, from a user perspective, this composition will continue to be considered as a simple service, even though it is composed of several web services.

Before starting to compose web services, they must first be discovered in order to be used by a composition mechanism. One of today's problems is that many functionally similar services are available increasing the number of services discovered by search mechanisms. Indeed, the discovery process may retrieve many services when, in fact, only one of them will be used in the composition. After discovering, there is a set of candidate services. Between these services, a selection must be done. Usually this selection process chooses the most appropriate service to execute a task. In fact, discovery is a prerequisite for selection, but selection is the main problem (Sreenath and Singh, 2004).

In order to differentiate services that will be selected, it is necessary to use a nonfunctional mechanism able to make this distinction. Hence, a quality of service (QoS) model is incorporated into each service and service selection will be also based on these quality criteria.

Evaluating the quality model, it will be composed of more than one criterion, a large quantity of candidate services and a lot of tasks to be executed by services. Thus, this makes difficult all possible explorations. As there is more than one criterion to differentiate services and no preference (weight) is given to them, more than one tradeoff solution can be found. As a result, this kind of problem should be treated using a multiobjective approach.

This paper proposes an analysis of quality criteria in order to select from a set of services those that belong to the composition. It is organized as follows: the next section describes the related works. The third section describes web service composition. The fourth section explains the problem model. The fifth section shows the resolution method. The last section we position this work in our research and present our conclusion.

# 2. RELATED WORKS

Many authors have proposed quality of service models for selecting web services. In general, QoS selection can be organized in categories, for example, authors that use QoS with agents (Sreenath and Singh, 2004), authors that apply QoS workflow into web services domain (Cardoso et al., 2004), those who insert QoS models into centralized registries (Ran 2003), and finally, authors that propose and execute QoS model for selecting web services (Zeng et al., 2003; Liu, Ngu and Zeng, 2004).

In (Ran 2003) the main idea is to include the QoS model into UDDI registries so that they centralize all QoS attributes for each service into an information provider. In (Sreenath and Singh, 2004) the authors propose a mutual evaluation process between agents to select a web service. It selects the best services based on rates given to providers by agents. The authors in (Cardoso et al., 2004) retrieve the main idea from Workflow Quality of Services and transpose it to web services technologies. They propose some QoS constraints that were implemented into METEOR workflow management systems for Genomic Projects. Ideas in (Zeng et al., 2003; Liu, Ngu and Zeng, 2004; Cardoso et al., 2004) are closest to our proposition, because we do not use agents, nor consider a QoS model in a centralized approach.

The main idea of Zeng's paper is to compose web services by finding the best plan for the composition. For that, each constraint in the QoS model is measured and aggregated into one main objective function. To resolve this proposition linear programming is used to select the best plan. In (Liu, Ngu and Zeng, 2004), in order

#### Selecting Web Services for Optimal Composition

to improve the last work, the authors propose specific domain criteria for each service that will be selected.

In our paper, we follow the quality model proposed in (Zeng et al., 2003) and introduce several improvements. First of all, we reinforced the concepts of reputation, because the original concept did not measure the pertinence of the rank given to a service by a user. Thus, in our model, ranking from users with good knowledge of the service domain are considered more accurate. As a result, we use fuzzy numbers to measure this criterion. Additionally, we consider that service composition problem using quality of services is intrinsically a multiobjective problem. However, in (Zeng et al., 2003) they resolve this problem by aggregating their objectives into a single objective function. In our work, we resolve this problem using multi-objective evolutionary algorithms (MOEA), without giving any weight to any quality criterion. We use a multi-objective genetic algorithm called NSGA-II to execute this model.

## **3. WEB SERVICES COMPOSITION**

Web service composition originated from the necessity to achieve a predetermined goal that cannot be realized by a standalone service. From a user's perspective, composite services, or more objectively, the set of services should seem like a single service. Internally in a composition, services can interact with each other to exchange parameters.

## **3.1 Problem Description**

In order to illustrate our approach, we use a Travel Service as an application example. Indeed, this scenario is a typical web services composition problems (Narayanan and McIlraith, 2002;OWL-S Coalition, 2005).

As far as creating the Travel service, we use three atomic services (which are not composed) that will internally execute the travel; each one independently executes a task. A task can be described as an activity that applies to a specific domain. In this work, we treat activity and task as the same things. We choose to use atomic services to compose the Travel service. However, this is not important since a composed service is viewed by a user as a single service. There are 3 tasks that will be executed by 3 services called Airplane service, Hotel service and CarRental service.

A planner, as explained in section 6, will determine the execution order of these three tasks. Many services are candidate to execute a given task. These services are the result of the discovery process.

## **3.2** Composition Model

The problem of composing web services can be characterized as a combinatory problem. In the composition we have a set of candidate services  $s_i$ ,  $i \in [1..n]$  that could execute a set of tasks  $t_j$ ,  $j \in [1..m]$ . However, it is necessary to consider that one service can be dependent of other services. The main goal is to find the optimal service composition, considering that there is a set of candidate services for each task.

Each service *i* is allocated to one task *j*, this association can be represented by a matrix  $(x_{ij})$  where  $s_i$  represents the candidate services and  $t_j$  represents the tasks. This matrix can represent the service's allocation to a composition. In our scenario the number of tasks *m* is limited to 3: the Airplane, Hotel and CarRental web service. Thus we can generate the matrix X as shown in Eq. (1).

$$X = \begin{cases} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ \vdots & \vdots & \vdots \end{cases}$$
(1)

For each possible composition, there is a set of candidate services but only one will be part of the composition. The Eq. (2) will determine which service will belong to the composition or not.

$$x_{ij} \equiv \left\{ \frac{1, \text{ if service } i \text{ is allocated } to \text{ task } j}{0, \text{ otherwise}} \right\}$$
(2)  
$$\forall i \in [1..n], \forall j \in [1..m]$$

Thus, the matrix X' (Eq.3) represents one of the possible combinations in which service 3 will execute task 1, service 1 will execute task 2 and task 3 will be executed by service 2. As a result, this composition will be formed by services 3, 1 and 2 respectively.

$$X' = \begin{cases} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{cases}$$
(3)

A non-determined number of tasks m can be used to compose a service and an unlimited number of candidate services n for each task j can be found. In fact, these possible combinations are considered for a pre-defined plan, which determines exactly in which order the tasks should be composed. However, concerning this environment, the plan can also be changed, and so other possible combinations might be overseen. Moreover, if it is considered that p plans using m tasks can be created, the problem becomes even harder.

## 4. **PROBLEM MODELING**

In order to differentiate candidate services with identical functionalities, nonfunctional (QoS) criteria must be used. The QoS model will be incorporated directly into services. The quality of service measure belongs to each service and each value refers to an activity that the service executes.

### 4.1 Qos Model

We propose four criteria as parameters for the quality model: cost, time, availability and reputation. Each one is presented below and a difference is made between the QoS of single service and the composition.

#### 4.1.1 Cost

The cost quality (Zeng et al., 2003)  $c_{ij}$  is the amount that a service requester needs to pay to execute this service *i* using task *j*. In reference to composition, this can be expressed in Eq. 4.

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} \qquad (4)$$

#### 4.1.2 Time

The time quality (Zeng et al. 2003)  $t_{ij}$  measures the execution time between the moment the request is sent and the moment the results are received. The time can be expressed in reference to composition (Eq. 5).

$$\sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} \qquad (5)$$

#### 4.1.3 Availability

The quality availability  $a_{ij}$  is the probability that the service can be accessed and used. It means that this quality is obtained by the number of times the service answers a request divided by the number of total requests. We can express this using the formula seen in Eq.6.

$$a_{ij} = \frac{req_{ij}}{tot_{ij}}, tot_{ij} \neq 0$$
 (6)

req is the number of successful requests to service *i* using task *j*, and *tot* is the total number of invocations. We decided to measure the probability based on

discrete values, as success times by total times. If we consider a composition, the availability can be expressed as the average value for each service in the composition, as shown in Eq. 7.

$$\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}}{n}, n > 0$$

$$(7)$$

#### 4.1.4 Reputation

The reputation quality  $r_{ij}$  is the measure of its trustworthiness. It depends on the user's experience using the service. Different end users can have different opinions about the same service. For many authors (Zeng et al., 2003; Liu, Ngu and Zeng, 2004), reputation can be defined as the average ranking given to the service by end users as shown in Eq. 8.

$$q_{rep}(s) = \frac{\sum_{i=1}^{n} k(s_i)}{n}$$
(8)

 $k(s_i)$  is the ranking applied to a service  $(s_i)$  and *n* is the number of times the service is ranked.

However, in real world, when something is judged for example, a paper in a conference, the reviewers have to give their knowledge domain, prior to giving their judgment. In the case where a reviewer receives a paper that she classifies as belonging only 60% to her area (knowledge domain), the grade that is given must be moderated based on 60% of knowledge. If the same grade is given by a reviewer with 90% of know-how on the domain, for sure her grade will be more accurate. Translating this real scenario into our reputation quality, we must include the knowledge domain of end users. After service execution, the user ranks the service, and gives a percentage about her knowledge on the service's domain. It will be, for instance, a simple question as "how much do I know about this area".

In order to measure this criterion, we used fuzzy logic to represent an imprecise quantity, as "nearly 8" or "practically 15" (Moura, 2002). We used the notion of fuzzy number which is represented as  $\tilde{a} = [a, a_0, \bar{a}]$  where  $\tilde{a}$  is the fuzzy number with minimal limit, modal value and maximal limit respectively. The *linguistic variables* that represent our reputation values are: bad, average and good, as shown in Figure 1.

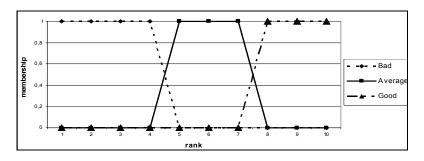


Figure 1. Fuzzy set representation

Figure 1 shows that until number 4, all grades are considered *bad*, from 5 to 7, grades are *average*, and after 8, all grades are *good*. The measure between 4 and 5, for example, depends on membership's values. The membership or degree of pertinence means how much a value is inside a set, for example the *bad* set or inside the *average* set. Thus, if a service has a rank 4.8 we need to analyze its membership  $\mu(d_i)$ . The  $\mu_{bad}=0.33$  (it belongs to *bad* set) and  $\mu_{average}=0.66$  (it belongs to average set). Each service will be ranked *n* times, so we will have a set of fuzzy numbers. However, at the end, what we need is a crisp number that characterizes the reputation value, and for that we need to convert fuzzy sets to a crisp number. Defuzzification is the final phase that does this conversion. There are several defuzzification methods, but we use the CENTROID method that calculates the hypothetical center of gravity for the output fuzzy set (Löfstedt, Svensson, 2000; Fuzzy, 2005). Thus, our reputation criterion is shown in Eq.9.

$$r_{ij} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} \mu(d_{ij})}{\sum_{i=1}^{n} \sum_{j=1}^{m} \mu(d_{ij})}$$
(9)

where  $d_{ij}$  represents the domain value of service *i* for task *j* and  $\mu(d_{ij})$  is the membership value for that domain point. We can consider that the rank is [0, 10], and the knowledge domain given by the user  $\mu(d)$  is [0,1]. Using this model, reputation ranks is more precise and trustworthy. In the composition context, the reputation is the average between all services. We can express our composition formula as Eq.10.

$$\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij}}{n}, n > 0$$
 (10)

We sum the reputation quality  $r_{ij}$  of each service in the composition and divide it by the number of services in this composition.

## 4.2 **Objectives**

In order to select services to obtain tradeoff composition, non-functional (quality of services) features will be used to distinguish between the candidate services. The quality of services criteria used was described in the previous section and corresponds to cost, time, availability and reputation. Existing studies reported in literature often aggregate all criteria into a single objective. This resolution approach is known to be limited from a multiobjective optimization perspective since it leads to one single solution instead of a set of solutions which are needed in a multiobjective context.

Here we highlight the most important difference between our work and Zeng et al's work (Zeng et al., 2003), because we do not give any weight to any criterion. We treat all criteria with the same importance using a multiobjective optimization approach. Even though our objectives are contradictory, they are taken into account simultaneously by our resolution algorithm. Thus, our travel problem explained earlier can be viewed as a multiobjective problem. Our goal is to minimize cost and time and to maximize availability and reputation, considering them together simultaneously. MOEA algorithms take into account contradictory objectives and also aim at finding other solutions that in an aggregate approach. In the case of biobjectives, the Pareto optimal Front can be graphically depicted as a curve.

In our problem, we have four objectives to be simultaneously taken into account. The first one is cost minimization (Eq. 11).

$$Min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} p_{ij} x_{ij}$$
(11)

Cost objective needs to be minimized in order to participate to optimal solutions. Additionally, this value will be obtained based on quality model cost criterion  $c_{ij}$ . The variable  $p_{ij}$  means the service ability to execute a task. Since we do not consider composed services to belong to our composition, it is necessary to distinguish between them, highlighting which is able to execute a specific task. In our model, we consider that a service is specialized only in one task, for example, a service for airplane reservation, cannot do hotel reservations tasks. Thus,  $p_{ij}$  is a binary variable to inform that a service is able or not to execute task *j*. The binary variable  $x_{ij}$  is responsible to express if a service belongs or not to the composition. It is the allocation matrix expressed in section 3.2.

Another objective is concerning the time (Eq. 12). Time objective needs also to be minimized to be part of service composition.

$$Min\sum_{i=1}^{n}\sum_{j=1}^{m}t_{ij}p_{ij}x_{ij}$$

(12)

In our model, time value is attributed to  $t_{ij}$ . The other variables as  $p_{ij}$  and  $x_{ij}$  are the same functionality as explained earlier.

On the other hand, availability objective (Eq.13) shows the probability that a service can be accessed and used. In this case, this objective must be maximized, because it is preferable that its probability is as high as possible.

$$Max \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} p_{ij} x_{ij}$$
 (13)

Variable  $a_{ij}$  should thus belong to [0,1]. The last objective is related to the reputation (Eq. 14) a service has about a determined domain.

$$Max \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} p_{ij} x_{ij}$$
(14)

The  $r_{ij}$  stands for reputation value. Also, reputation needs to maximize the objective, because the higher the reputation the better the service is judged. Thus these four objectives must be used together, and no preference (weight) is given to any of them.

## 4.3 Constraints

In order to model the problem, the constraints should also be verified. In our model three constraints must be satisfied in all interactions. The first one (Eq. 15) means that only one service belongs to a composition for each task.

$$\sum_{i=1}^{n} x_{ij} p_{ij} = 1, \forall_{j} \in [1, m]$$
(15)

The variable  $x_{ij}$  specifies whether or not a service belongs to a composition (matrix X). The variable  $p_{ij}$  means that a service *i* is able to execute a task *j*. The second restriction (Eq. 16) concerns user's budget.

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \le W, W > 0$$
 (16)

This constraint means that the cost of using a service in a composition should not exceed a given value W. Finally, the last constraint used is only to verify that  $x_{ij}$  must have binary values:  $x_{ij} \in \{0,1\}$ . These binaries values will represent which services belong or not to the composition.

# 5. **RESOLUTION METHOD**

One of the main contributions of this work concerns the resolution approach employed. As explained earlier, we have four objectives and solutions should be searched considering these four criteria simultaneously. To achieve this, we use a multi-objective evolutionary algorithm NSGA-II.

## 5.1 Genetic Algorithm - NSGA-II

The multi-objective evolutionary algorithms (MOEA) are among the most powerful resolution methods for multiobjective optimization (Coello Coello et al., 2002). Thanks to its population, such an algorithm is able to produce naturally a set of non-dominated solutions for each run of the algorithm. In this paper, we used NSGA-II (Non-dominated Sorting Genetic Algorithm) (Deb et al. 2002), though any other MOEA such as SPEA (Zitizler and Thiele, 1998), PAES (Knowles and Corne, 1999) and PICPA (Barichard and Hao, 2003) could have been used.

In NSGA-II, the tournament selection, crossover and mutation operators are used to create a child population that will be added to a result population given by the later generation. The new population is sorted based on non-domination. In this step, elitism is ensured because the best non-dominated sets will be chosen for the next population. Using constraints, the relation of domination between two individuals can be characterized as a feasible or infeasible solution. Thus, the ranking will be done based also on feasible solutions.

Using the NSGA-II algorithm, it is not necessary to create a mathematical expression to determine the weights associated to the criteria which would be necessary with an aggregation approach.

#### 5.2 **Experimentation**

Applying this algorithm to our problem, several experiments using our optimal composition model were done in order to validate the multiobjective approach for selecting web services.

#### 5.2.1 Tests set

The main objective of our tests was to find a set of optimal compositions from which a user can select her preferred solution. Thus our main contribution is creating optimal compositions based on discovered services. The first test that we did was using the same number of services, but changing the number of generations and populations. The number of services was fixed to 15 services and the number of tasks to 3. We choose to allocate the same number of candidate services to each task. This was done in order to see how the algorithm treats services composition.

The other we did was aimed at studying the scalability of the service composition algorithm with regards to the number of candidate services and to the number of tasks. Population and generation were kept constant in all experiments, but the number of services and tasks was changed. In fact, we increased candidate services for each tasks. The population was fixed to 50 individuals and the generations were fixed to 100. These values were taken considering other experiments using the NSGA-II algorithm.

The number of services is proportional to the number of variables, because each service is represented as a variable in our model. We consider in our experiments that the numbers of candidate services for each task are equal.

#### 5.2.2 Algorithm's Parameter

In the first experiment we used populations range from 5 to 100 and generation range from 10 to 500. The crossover probability was 0.9 and the mutation was 1/l where *l* is the number of binary variables. In our case, we used 15 binary variables. These 15 binary variables represent 3 tasks and each task can be executed by 5

#### Selecting Web Services for Optimal Composition

candidate services. The crossover used was single-point. We used 4 objective functions and 4 constraints as previously defined in our model. The 3 constraints determine the candidate services and the last one represents maximal budget given by the user. This value was fixed for all compositions. A QoS value was given randomly to each service.

In the second test, the population size was fixed to 50 and generation was fixed to 100. We did these experiments taking 15, 30 and 60 services and 3 and 5 tasks respectively. It means that, for example, using 60 services and 3 tasks, we have 20 candidate services evenly distributed for each task. The crossover mutation and probability was maintained, of course changing according to the number of variables. In both experiments, all constraints are verified and they must be satisfied in all generations. Only solutions with satisfied constraints were selected for the next generation. Thus optimal solutions must satisfy all constraints.

#### 5.2.3 Results

As execution results, a set of chromosomes is found and each one represents an optimal composition. Since we defined a population size of 50, the maximum number of optimal solutions found was also 50. Between these 50 optimal solutions, the user can select one of them to be executed.

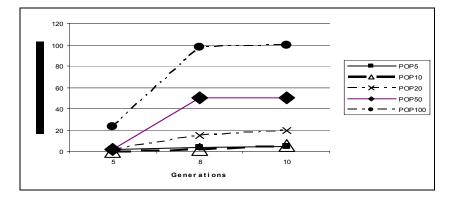


Figure 2. Non-dominated set solutions - Population by generations

In figure 2, we show the evolution of our model based on the number of optimal solutions found. We can ensure that when population is over 20, tradeoff solutions are found at the 8<sup>th</sup> generation. The tradeoff solutions do not violate any constraints. Thus, using 15 services for 3 tasks, results show that a GA does not have any problem in finding tradeoff solutions. In fact, found solutions are non-dominated, since once the solutions are achieved, they are maintained.

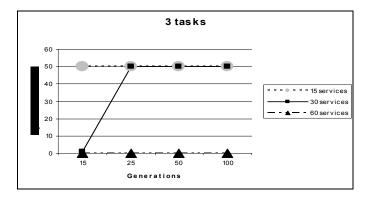


Figure 3. Non-dominated set solutions - 3 tasks

The next experiment consisted in changing the number of services and the number of tasks. In figure 3 we can ensure that as the number of services increases, more generations are necessary to obtain a set of optimal solutions. In some cases, as seen with 60 services, any optimal solution was found without constraint violation. The optimal solutions will only be found at the 155<sup>th</sup> generation. Also here, using 60 services for 3 tasks, it means that there are 20 candidate services. The difficulty in finding tradeoff solutions increases with the number of candidate services. In order to show this fact, we changed the number of tasks, minimizing the number of candidate services. We considered 5 tasks for 60 services given, so only 12 candidate services for each task. Augmenting the number of tasks also means increasing the number of constraints and so facilitating the acquisition of tradeoff compositions solutions. All obtained solutions are composed of a set of binary variables that form a chromosome. This chromosome represents the composition chain and each binary variable indicates whether the service belongs to the optimal composition or not. Thus we can ensure that the experiments done can validate our approach to make tradeoff services compositions.

## 6. **PROPOSED ENVIRONMENT**

This work is positioned as part of an environment that includes a planner, a discovery method and an optimization structure. The planner is responsible for determining the execution order of the tasks. However, in order to determine tasks plan, the planner must know the tasks definitions. In this way, all possible tasks will be located into a repository that the planner can consult for getting tasks interfaces. This repository can be represented as ontology. Thus, before knowing the tasks interface, it is necessary to find a composition which satisfies user request. A planner can automatically resolve this type of problem (McIlraith and Son, 2002). Indeed, the problem is composed of three components: user request, services and user initial parameters. These three components represent respectively the goal, the

#### Selecting Web Services for Optimal Composition

actions and the initial planning state. In this paper, we pre-defined that tasks should be executed in sequence as: Airplane, Hotel and at last CarRental.

After creating the initial plan, the discovery process will take place. It is responsible for finding services that correspond to each task that belongs to the plan. For discovering services, we use the profile ontology from OWL-S (OWL-S Coalition, 2005). The next phase is aimed at optimizing services composition and is the point treated in this paper. The user can choose any one of these solutions to execute. In the execution phase, if any service has a problem such as invalid URL or changed location, the environment proposes another optimal solution to the user. If after some predefined time the problem continues, the environment will propose to construct another plan, for example, by reordering the tasks.

# 7. CONCLUSION

In this paper we have explained how to select a service in order to make an optimal web service composition. We proposed some improvement on quality models, highlighting reputation. Reputation is calculated based on fuzzy numbers, but at the end, we use a defuzzification method to convert this fuzzy numbers into crisp values. Furthermore, when optimizing composite services using non-functional features (QoS), we can have contradictory objectives. Moreover, we do not wish to give any preference (weight) to anyone of these objectives. Thus we treat services composition as a multiobjective problem. We used a multiobjective evolutionary algorithm called NSGA-II. As a result, we obtained the population (a set of chromosomes) representing the optimized compositions. Each solution is represented by a set of binary values that determines which service belongs to the composition or not. The experimentations done validate our approach and show its feasibility in a real world problem.

# 8. ACNOWLEDGEMENT

The authors would like to thank all anonymous reviewers that helped us improve the contents of the paper. Daniela Barreiro Claro is supported by a research scholarship given by the Région du Pays de La Loire (2003-2006).

# 9. **REFERENCES**

- Barichard V. and Hao J-K. 2003. A population and Interval Constraint Propagation Algorithm. In Second International Conference Evolutionary Multi-Criterion Optimization (EMO) Lecture Notes in Computer Science 2632: 88-101.
- Cardoso J., Sheth A., Miller J., Arnold J., Kochut K. 2004. Quality of service for workflows and web service processes. *In Web Semantics: Sciences, Services and Agents on the World Wide Web* 1(2004) 281-308, Elsevier.

- Coello Coello C.A, Van Veldhuizen D.A and Lamont G.B, 2002. Evolutionary algorithms for solving multi-objective problems. Kluwer Academic Publishers, New York.
- Deb K., Pratap A., Agarwal S., Meyarivan T. 2002. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II, *in Journal IEEE Trans Evol Computat*, Volume 6, pp. 182-197, April, Indian Institute Technologic.
- Fuzzy Logic Fundamentals. 2005. *Chapter 3, pg 61-103*, Retrieved June 4, 2005, Available on http://www.informit.com/content/images/0135705991/ samplechapter/ 0135705991.pdf.
- Knowles J., Corne D, 1999. The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. *In proceedings of the 1999 Congress* of Evolutionary Computation, Piscataway, New Jersey: IEEE Service Center, pp. 98 – 105
- Liu Y., Ngu A.H.H., Zeng L, 2004. QoS Computation and Policing in Dynamic Web Service. In proceedings of the Thirteenth International Conference of WWW 2004, May 17-22, New York, New York, USA.
- Löfstedt J. and Svensson M., Baltazar A Fuzzy Expert for Driving Situation Detection, 2000. *Master Diss., Department of Sciences,* Lund University.
- McIlraith, S. and Son, T.C. 2002. Adapting Golog for Composition of Semantic Web Services. In 8<sup>th</sup> International Conference on Principles of Knowledge Representation and Reasoning, April 22-25, Toulouse, France.
- Moura L. 2002. A Genetic algorithm to fuzzy multiobjective optimization. *Master* diss. Department of Electric Engineer, Campinas University.
- Narayanan S. and McIlraith S.A. 2002. Simulation, Verification and Automated Composition of Web Services. In Proceedings of Eleventh International World Wide Web Conference (WWW 2002), Honolulu, May 7-10.
- OWL-S Coalition. 2005. OWL-S: Semantic Markup for Web Services. OWL Services Coalition. Retrieved June 4, 2005, from http://www.daml.org/services/owl-s/1.1/.
- Ran S. 2003. A Model for Web Services Discovery with QoS. in ACM SIGecom Exchanges, Volume 4, Issue 1, Spring, pp. 1-10, ACM Press, New York, NY, USA.
- Sreenath R.M., Singh M.P. 2004. Agent-based service selection. *In Web Semantics: Science, Service and Agents on the World Wide Web*, pp.261-279, 2004.
- Zeng L., Benatallah B., Dumas M., Kalagnanam J., Sheng Q.Z. 2003. Quality Driven Web Services Composition. In proceedings of the Twelfth International Conference of WWW, May 20-24, Budapest, Hungary.
- Zitizler E., Thiele L, 1998. Multiobjective optimization using evolutionary algorithms – A comparative case study. *In Parallel Problem Solving from Nature* V, A.E.Eiben, T.Bäck, M.Schoenauer and H-P. Schwefel, Eds. Berlin, Germany: Springer, 1998, pp. 292-301.