Linked Service Selection using the Skyline Algorithm

Mahdi BENNARA[1], Michael MRISSA[2], and Youssef AMGHAR[1]

[1] Université de Lyon, LIRIS
INSA-Lyon - CNRS UMR5205
F-69621, France
{mahdi.bennara,youssef.amghar}@liris.cnrs.fr
[2] Université de Lyon, LIRIS
Université Lyon 1 - CNRS UMR5205
F-69622, France
michael.mrissa@liris.cnrs.fr

Abstract. Recently, resource oriented computing has changed the way Web applications are designed. Because of the increasing number of APIs, centralized repositories are no longer a viable option for discovery. As a consequence, a decentralized approach is needed in order to enable value-added applications. In this paper, we propose a client-side QoS-based selection algorithm that can be executed along the discovery stage. Our solution provides different alternatives based on the skyline approach to select resources and maintain acceptable time performance.

Keywords: RESTful linked Web services, discovery, selection, quality of service

1 Introduction

In the past twenty years, SOAP-based Web services have helped reaching syntactic level interoperability for distributed applications on the Web. More recently, resource-oriented computing, and in particular the REST architectural style [5], has revised the way we interact with services, bringing in new advantages such as uniform interface (and consequently generic client), HATEOAS [3] (hypertext-driven applications), cacheability, etc. In parallel, the Web of services has started an evolution towards semantic-level interoperability, with a lot of work around semantically described Web services [7] [6] to allow services to exchange semantically annotated data. This evolution has moved towards linked data. Combining the RESTful architectural style with semantic annotations has unlocked the benefits of using linked data for Web applications. We talk about Linked Web Services which are RESTful services described with linked data and that exchange linked data. Despite the evolution of service technologies, the need for service composition to build complex applications is still present. However, the challenges have changed. As centralized solutions for service discovery

[3] Hypermedia As The Engine Of the Application State

have proven not to scale well [2], the need for distributed service discovery has emerged [9]. The discovery of services that fulfill a certain task in the process of answering the user's request also brings in the need for selecting the most suitable of these candidates to actually execute the task needed.

In this paper, we introduce a set of algorithms for discovering and selecting semantically described resources. As we work in the context of resource oriented Web, the discovery of new resources that can potentially participate in answering the user's request is done progressively following the principle of HATEOAS. We rely on semantic annotations developed in previous work [3] to describe resources. Such descriptions include resource attributes, available HTTP operations and relations with other resources. We also rely on breadth-first search algorithm combined with a skyline-based approach [4] to select the appropriate resources while maintaining acceptable performance.

The remainder of this paper is organized as follows. Section 2 presents related work and highlights the originality of our solution. Section 3 introduces a scenario that motivates our approach. Section 5 details our contribution and shows the different setups for discovery-selection as well as the algorithms. Section 6 discusses the choices we made in the contribution section and highlights the advantages of our approach. Section 7 resumes our approach and lists some elements of future work.

2 State of the Art

2.1 Quality of Service in Service Oriented Web

During the last years, QoS has been a key challenge for research community.

Ran [8] proposed a model for QoS in Service oriented Web that divides QoS attributes into several different categories. This is one of the efforts to provide a complete set of attributes that describe the QoS aspects of a service. It is designed for the service-oriented architectures and particularly the Web.

AgFlow [13] is a solution that enables quality-driven composition of Web services. It proposes two different approaches to select Web services for a given task within a composite service. The local optimization approach suggests to leave the selection to the last possible moment. In the global planning approach, the selection is done for each task individually but by taking into account the other tasks.

2.2 QoS-based Web service selection

The problem of selection of Web services is a part the composition process that involves the QoS aspect to choose the most suitable services for the user. Finding the optimal solution for this problem with multiple QoS constraints is a NP-hard combinatorial optimization problem [12]. This problem can be modeled as follows: the user emits a request that requires several Web services to be answered. The solution for this request is divided into several tasks, every task can be performed by a single Web service. Every task has several candidate Web services. Finding a solution amounts to select the best candidate for each task in order to obtain the highest overall QoS. In the current paper we follow a different setup where the search space of candidates progressively discovered by following links between resources.

Wang et al. [10] propose an approach for selecting services based on Generic and Domain-related QoS attributes (DGQOS). Generic QoS attributes (GQoS) can apply on any type of Web service. Domain QoS attributes (DQoS) apply only on a certain class of Web services. The authors define evaluation models for DQoS and GQoS attributes, which help calculate the overall QoS of composite Web services based on its components. They use the C-MMAS (Cultural Min-Max Ant System) algorithm in order to solve the selection problem.

Alrifai et al. [1] propose a solution for selection Web services based on the skyline approach. The goal is to identify for each task the services that will never be part of the final solution simply because they are outclassed by another candidate service in every QoS-related aspect. The authors try to keep the set of candidates as small as possible in order to apply constraint optimization algorithms to obtain the best solution. They also propose a solution to further reduce the size of the set of candidates by identifying representative candidates that replace a subset of candidates that have similar QoS parameters.

2.3 Analysis

We have to highlight that in our context, the set of candidate resources to be selected is progressively discovered. Indeed, the setup of a centralized registry for all resources is not adapted to the distributed and large scale nature of the Web. Therefore, the discovery process gradually discovers resources that can fulfill a given task and does not have all the candidates until the end of the algorithm. Plus, the space of solutions expands as the discovery algorithm finds new candidates which increases the computational time of the selection process. Here, we face two possibilities. The first one is to run the selection algorithm while the discovery algorithm is exploring the resources on the Web. When a new candidate is discovered, it may be a part of the final solution. The second possibility is to run the selection algorithm at the end after having all candidates for every resource, the selection algorithm is run a single time after the end of the discovery algorithm.

3 Scenario and Motivation

In order to motivate our work, we consider a scenario where a user wants to buy a book online. We assume there are three book selling Web resources, three shipment Web resources and two online payment Web resources in competition. Every Web resource has its own quality of service properties, and the user has its own preferences. The disposal of the scenario setup includes the human user, the machine client software and the set of Web resources listed above.

The process of buying a book requires shipment and online payment, and is described as follows: the user emits a request explaining his needs e.g. “buy book”. This request is combined with a set of QoS-related information which guide the machine in the processing of his request, we agree to call this set of information “user profile”. For example, the user may want the services with the best performance and then the best rated among these and does not care about their availability. The client-side reasoner infers that it needs to discover a book selling, a shipping service and a payment service. The process of deduction is achieved through a reasoner with a simple subsumption technique. If we take the example of the user’s request : “buy book”, the reasoner infers that there is a need for a service that sells books in order to allow the user to buy books. The ontology we use for this scenario can be found here : `http://soc.univ-lyon1.fr/bookselling.owl`. The client begins to crawl the Web looking for the resources needed to answer the user’s request. The client needs to discover three services according to the reasoning on the request: the book selling service, the shipping service and the online payment service. The client may find multiple services that fulfill each functionality and has to select one based on the user’s QoS profile.

4 Problem statement

We consider a user request that involves a set of tasks, each task is represented by an ontology concept. The set of tasks is organized as a workflow that represents the series of actions that the client needs to perform in order to deliver the result to the user. The Web resource that can fulfill a specific task is not unique due to the nature of the Web and the client can discover many candidate resources to fulfill the given task. However, not all these candidates match the user requirements in terms of QoS, and therefore a selection phase is needed in order to determine the most suitable candidate for each task.

$T = \{OntologyConcept\}^N$ is the set of tasks in the workflow, where N is the total number of tasks.

$C_i = \{(URI, operation)\}^{m_i}$ is the set of candidates for a task t_i, where m_i is the total number of candidates for the task t_i. The candidate number j for the task t_i is therfore c_{ij}.

Finding a solution for the user’s request amounts to finding the set of candidates c_k for each task t_k, where every candidate meets the hard constraints for the user at least (and preferably the soft constraints) and the overall QoS of the set is the best among all combinations. We define hard constraints as conditions that must be fulfilled by a discovered resource otherwise it is not eligible for the task. On the other hand soft constraints are optional conditions that are not necessary to select a candidate for a task.

As we have opted for a hypermedia-driven approach for exploring the Web, we need an on-the-fly selection strategy in order to be able to select relevant resources along the discovery process. In the remainder of this section, we detail a “select while you discover” strategy to enable on-the-fly selection.

4.1 A minimal QoS model for Web resources

In this paper, we rely on a minimal model based on [8] in order to describe some important non-functional properties of a Web resource. QoS : { Performance: [0-10], Availability: [0-100], Reputation: [0-5] }

4.2 QoS-based resource selection problem specification

The problem of selection of resources is part of the composition problem. This problem has been proven to be NP-Complete (c.f. Section 2). The process of reasoning on the user's request, as described in the scenario (c.f. Section 3), can be assimilated to a Web service selection problem. Every task of the composite solution can be fulfilled by a resource, that has to be discovered. Multiple resources can be candidates for a single task. Tasks are semantically identified by the concepts the reasoner infers after analyzing the user's request. We start with N tasks and for each task i we have M_i candidates. The problem is to identify the set of candidates S where each candidate s_j fulfills the task t_j and the overall QoS of the set is the best according to the user's preferences.

5 Contribution

In this paper, we propose a quality-driven approach that relies on a minimal QoS model to improve selection of RESTful resources. We want the QoS attributes to be a guide together with the resource descriptions (including links between resources) for the selection process. We rely on a minimal QoS model based on a set of quality attributes in order to incorporate it on the resource descriptions according to the Descriptor concept [3]. The selection phase is the step where we have multiple candidates for each task of the process aiming to answer the user's request. The selection phase aims at selecting the best candidate for each task to obtain an overall QoS that matches with the user's QoS profile. In this paper, we present two configurations of the discovery and selection processes.

5.1 On-The-Fly Selection

In this configuration, the selection process is executed at the same time as the discovery goes on. Each time a resource is discovered, the selection algorithm is run in order to verify if this new resource can be the best candidate for its task among the other previously discovered resources for that same task while, at the same time, making sure the set of selected candidates for all the tasks verify certain conditions (best overall QoS matching with user's profile, compatibility between resources, etc.).

In table 1, we present four different setups to enable the on-the-fly selection (select as you discover)

The general selection algorithm takes as input the set of all candidates T for each resource, the current set of best candidates s and the new candidate c

Setup	**Advantage**	**Drawback**
Selection of the first solution that matches the user's QoS profile	Fast solution	Low overall QoS, may not match with the user's soft constraints
Selection of the best candidate for each task	Selection of the best candidate for each task	May not be the best solution to obtain the best overall QoS
Selection of the best solution, by exploring all combinations	Ensures the best solution amongst all combinations	Slow solution, exponential processing time
Selection of the best solution while eliminating irrelevant candidates using skyline approach	Ensures the best solution amongst all combinations, while having less candidates to work with	Still a relatively solution but a lot better than naive exploring of all combinations

Table 1. Different on-the-fly selection setups

resource as well as the user's QoS preferences *qos* and returns a new set of best candidates and updates If it is selected the set of best candidates is updated with the better new solution. For this purpose, we consider a discovered candidate as an object composed of two attributes resource URI and HTTP operation (`uri` and `operation`) plus the concept that matches is with the operation (`concept`). In the context of the algorithms presented below, we define the concept of domination as follows : a candidate c_1 dominates another candidate c_2 if all of c_1's QoS attributes are equal of better compared to c_2's QoS attributes.

Algorithm 1 shows the optimized global planning method. This algorithm eliminates the candidates that will not be part of the final solution before reevaluating the solution. If the new candidate cannot be added to the set of candidates (is irrelevant), the algorithm does nothing and skips this iteration.

Algorithm 2 shows how to insert the new candidate and how the irrelevant ones are removed right after.

5.2 N-Periodic Selection

Launching the selection every time we have N new candidates for a given task can reduce processing time for the selection phase, with the skyline based setup. Note that the number of new candidates for a given task t is n_t where $\sum_{i=1}^{M} n_t = N$ (M being is the total number of tasks).

Algorithm 3 shows how to apply the skyline approach to reduce the size of candidates for a given task t when n_t new candidates are discovered in each iteration of the selection process, instead of only one.

We know that the set of old candidates for the task t is a skyline i.e. no old candidate in $T[t]$ is dominated by another one in the same set. The first step is to eliminate the new candidates in $(N[t])$ that are dominated at least one

Algorithm 1: On-The-Fly optimized selection algorithm

```
Input: s: array of Candidate
Input: c: Candidate
Input: qos : QoSprofile
Input: T : array of array of Candidate
Output: s: array of Candidate
1  var s2: array of Candidate = s
2  // If the new candidate matches user requirements :
3  if QoSmatch(c, qos) then
4      // add c while removing irrelevant candidates
5      // if c is irrelevant quit if (skyline(T, c) = true) return
6      // and verify if there is a new best solution :
7      for i = 0 to T.size do
8          for j = 0 to T[i].size do
9              s2[i] = T[i,j]
10             if QoScalculate(s2, qos) > QoScalculate(s, qos) then
11                 s = s2
```

Algorithm 2: Inserting the new candidates and removing the irrelevant ones using the skyline approach

```
Input: c: Candidate
Input: T : array of array of Candidate
Output: T : array of array of Candidate
1  var x : type = init
2  // if c is not dominated by any other candidate for the same task
3  for i = 0 to T[c.concept].size do
4      if dominate(T[c.concept][i], c) then
5          return true
6  // insert it T[c.concept].insert(c);
7  // remove candidates dominated by c foreach c2 in T[c.concept] do
8      if dominate(c, c2) then
9          T[c.concept].remove(c2);
10 return false
```

Algorithm 3: Selection of n resources at a time instead of one

```
Input: N : array of array of Candidate
Input: T : array of array of Candidate
Output: T : array of array of Candidate
// apply the skyline on the set of new candidates first
for i = 0 to N[t].size do
    for j = i+1 to N[t].size do
        if dominate(N[t][i], N[t][j]) then
            N[t].remove(j);
        if dominate(N[t][j], N[t][i]) then
            N[t].remove(i); break;
// remove new candidates dominated by old ones
for i = 0 to N[t].size do
    for j = 0 to T[t].size do
        if dominate(T[t][j], N[t][i]) then
            N[t].remove(i); break;
// remove old candidates dominated by new ones
for i = 0 to T[t].size do
    for j = 0 to N[t].size do
        if dominate(N[t][j], t[t][i]) then
            T[t].remove(i); break;
// merge the two new sets
T[t].merge(N[t]);
```

candidate of the same set. After that, we eliminate the new candidates (which is now a skyline) that are dominated by at least one old candidate ($T[t]$). Next, we eliminate old candidates dominated by at least one new candidate. Now we know that no candidate in $T[t]$ or $N[t]$ is dominated by any other candidate in the two sets. Finally, we merge the two sets in order to obtain the skyline of candidates for the task t.

6 Discussion and Theoretical Evaluation

6.1 One-periodic selection versus N-periodic selection

Executing the selection algorithm every time a new candidate is discovered can hinder the processing performance to answer the user's request. With the skyline-based solution the overall execution time can be optimized through waiting for N new candidates to start the selection.

Lets suppose the number of new candidates for each task t is m_t, where $\sum_{i=1}^{N} m_i = n$ where N is the number of tasks. Let us suppose the number of the old candidates for each task is l_t.

In the worst case (i.e no new nor old candidate is dominated by another), the number of iterations is exactly the same with One-periodic or N-periodic selection : $2l_t m_t + \frac{m(m-1)}{2}$. But in the general case, the number of iterations in N-periodic selection is lower. Indeed, we eliminate the irrelevant candidates in the set $s1$ of the newly discovered n candidates to obtain a set $s2$ of $m \leq n$ candidates. After that, we consider $s2$ and eliminate the candidates that are dominated by at least one element of $T1$ to obtain $s3$ with $|s3| \leq |s2|$. Next, we consider the complete set of candidates $T1$ and eliminate the candidates that are dominated by at least one element of $s2$ to obtain a new set $T2$ where $|T2| \leq |T1|$. Finally we merge $T2$ and $s3$ to obtain the final set $T3$ that represents the whole set of candidates without irrelevant candidates.

6.2 Selection algorithms of the skyline set of services

After reducing the number of candidates with the skyline algorithm, we need to choose the best solution for selection. In our contribution we show the naive combinatorial algorithm in order to explore the reduced space of solutions. We use a double loop in order to explore the two dimension array of candidates. There are some optimized algorithms [11] specifically aimed at obtaining better performances for this class of optimization problems.

In some cases, the size of the set of candidates is very large that, even with the algorithms we proposed, the solution can not be obtained in a reasonable amount of time. Some solutions have been proposed to resolve this problem, such as the representative skyline services proposed in [1].

7 Conclusion

In this paper, we propose a skyline-based approach to enable Web resource selection. We show that a solution based on the HATEOAS principle, where we select the Web resource candidates along the discovery stage, is more efficient for selection than a classical solution that consists in waiting for discovery results before the selection stage. We rely on a minimal QoS model to demonstrate our approach. We provide four different setups in order to satisfy the user requirements according to the QoS profile and preferences. We enhance the performance of our solution with a skyline-based algorithm in order to reduce the set of candidates for a given task and demonstrate that it gives the same output as with a fully combinatorial algorithm but with less candidates and therefore less overall computational time.

As future work, we envision to consider constraints between candidates for different tasks while running the selection process. In other words, the set of candidates for a given task can be different depending on the chosen candidate for other tasks and also on the user's preferences.

8 Acknowledgment

We would like to thank Karim Benouaret for his fruitful discussions about the application of the skyline approach over the selection-on-the-fly setup.

References

1. Alrifai, M., Skoutas, D., Risse, T.: Selecting skyline services for qos-based web service composition. In: Proceedings of the 19th international conference on World wide web. pp. 11–20. ACM (2010)
2. Anadiotis, G., Kotoulas, S., Lausen, H., Siebes, R.: Massively scalable web service discovery. In: Awan, I., Younas, M., Hara, T., Durresi, A. (eds.) The IEEE 23rd International Conference on Advanced Information Networking and Applications, AINA 2009, Bradford, United Kingdom, May 26-29, 2009. pp. 394–402. IEEE Computer Society (2009), http://dx.doi.org/10.1109/AINA.2009.106
3. Bennara, M., Amghar, Y., Mrissa, M.: Managing web resource compositions. In: Reddy, S. (ed.) 24th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE Workshops 2015, Larnaca, Cyprus, June 15-17, 2015. pp. 176–181. IEEE (2015)
4. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: Data Engineering, 2001. Proceedings. 17th International Conference on. pp. 421–430. IEEE (2001)
5. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine (2000), aAI9980887
6. Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing 11(6), 60–67 (2007)
7. Martin, D.L., Paolucci, M., McIlraith, S.A., Burstein, M.H., McDermott, D.V., McGuinness, D.L., Parsia, B., Payne, T.R., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.P.: Bringing semantics to web services: The OWL-S approach. In: Cardoso, J., Sheth, A.P. (eds.) Semantic Web Services and Web Process Composition, First International Workshop, SWSWPC 2004, San Diego, CA, USA, July 6, 2004, Revised Selected Papers. pp. 26–42. Springer (2004)
8. Ran, S.: A model for web services discovery with qos. SIGecom Exchanges 4(1), 1–10 (2003), http://doi.acm.org/10.1145/844357.844360
9. Verborgh, R., Hausenblas, M., Steiner, T., Mannens, E., de Walle, R.V.: Distributed affordance: an open-world assumption for hypermedia. In: Carr, L., Laender, A.H.F., Lóscio, B.F., King, I., Fontoura, M., Vrandecic, D., Aroyo, L., de Oliveira, J.P.M., Lima, F., Wilde, E. (eds.) WWW (Companion Volume). pp. 1399–1406. International World Wide Web Conferences Steering Committee / ACM (2013)
10. Wang, Z.J., Liu, Z.Z., Zhou, X.F., Lou, Y.S.: An approach for composite web service selection based on dgqos. The International Journal of Advanced Manufacturing Technology 56(9-12), 1167–1179 (2011)
11. Wolsey, L.A., Nemhauser, G.L.: Integer and combinatorial optimization. John Wiley & Sons (2014)
12. Yu, T., Zhang, Y., Lin, K.: Efficient algorithms for web services selection with end-to-end qos constraints. TWEB 1(1) (2007)
13. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Software Eng. 30(5), 311–327 (2004)