Selecting Web Service Compositions under Uncertain QoS

Remaci Zeyneb yasmina[1], Hadjila Fethallah[1], Didi Fedoua[1]
Computer Sciences Department
LRIT Laboratory[1]
UABT University – Tlemcen
Tlemcen Algeria
yasmina_rmc @hotmail.fr, {f_hadjila, f_didi}@mail.univ-tlemcen.dz

Abstract. The uncertain QoS management is gaining a lot of interest in the service oriented computing area. In this work, we propose a framework that allows to select the TopK compositions of services that best meet the user's requirements. This framework not only handles the user's global constraints but it also takes into account the fluctuating nature of the QoS informations. More specifically we present two algorithms that ensure the aforementioned purposes. The first one ranks the services of each abstract class according to the probabilistic dominance heuristic. The second one explores the compositions search space by leveraging the backtracking search. The experimental evaluation shows that the proposed heuristic is more effective than the ranking based on average QoS.

Keyword: Service Oriented Architecture, Web Service Selection, uncertain Quality Of Service, backtracking search, Combinatorial Optimization.

1 Introduction

Over the last decade, the web services have been increasingly published and deployed over the web. Consequently, many providers offer the same functionality, (i.e. the same interface/behavior) but they differ according to their non-functional attributes (or quality of service such as response time, cost, reputation, availability…). In this context, the user has to leverage the QoS to select the best advertised services that meet his/her requirements. On the other hand, we also observe that the service QoS is generally fluctuating and non-deterministic. This is mainly due to the environment circumstances (i.e. the price of a service depends to the season; the response time/the throughput depend to the network load…). As a result, our selection/aggregation models should take into account these fluctuations. Additionally, we notice that a complex user's request is generally fulfilled with a composition of services rather than a single component. This means that the optimization/selection algorithms should not only handle the non-deterministic QoS, but also the global optimization aspects (i.e. global constraints, aggregated QoS,…). In the example cited in table I, we assume a user's request that consists in invoking two types of services: a currency service and a purchase order. Each service is characterized by two criteria: the cost (denoted C in table I), and the latency (denoted L in table I). The selected combination must have a global cost (the QoS sum of the composition services) less or equal than 0.8 (according to a given unit such as $) and a global (aggregated) latency less or equal than 0.9 (according to a given unit such Seconds).

Table 1. Normalized QoS of service instances

Currency conversion						**Purchase Order**				
X			Y			S			T	
	C	L		C	L		C	L	C	L
X_1	0.3	0.3	Y1	0.2	0.1	S1	0.5	0.2	T1	0.3	0.5
X_2	0.2	0.5	Y2	0.3	0.4	S2	0.5	0.3	T2	0.2	0.5
X_3	0.6	0.3	Y3	0.3	0.6	S3	0.7	0.3	T3	0.4	0.7
X_4	0.5	0.6	Y4	0.6	0.6	S4	0.8	0.6	T4	0.6	0.4

Furthermore, the table I, also shows the QoS variation (see the instances lines such as $X_1,\ldots X_4$) of each service. By considering the elements of the currency class (i.e the services X and Y that have same functionality), we notice that the comparison of their performances is not always self-evident. More specifically, if we use the mean QoS as comparison mechanism, this can create a misleading result. Simply speaking, the mean QoS of X is (0.4,0.42), likewise the mean QoS of Y is (0.35,0.42), and therefore Y is better than X (ie. Y >> X), but if we consider all the instances (i.e the QoS variations), then we observe that Y dominate X in 37% of the cases and X is dominate Y in 43% of the cases, furthermore the QoS instances of X have a reduced variance in comparison with those of Y. consequently our initial ordering may be erroneous.To tackle these difficulties, we should use an ordering scheme that takes into account all the sampled QoS (and not the aggregated values). In addition, any proposed service selection system should differentiate between feasible compositions and non-feasible compositions. For example if we consider the median QoS of each component service as the representative value, then the composition c=<Y,S>, is not feasible because :

MedianCost(Y)+MedianCost(S)=0.3+0.6>0.8. (The first global constraint is violated)

On the other hand, the composition c'=<X,T> is feasible since:

MedianCost(X)+MedianCost(T)=0.4+0.35 ≤0.8, and

MedianLatency(X)+MedianLatency(T)=0.4+0.5≤0.9.

By analyzing the literature approaches, we notice that the majority of the service composition works don't handle the non-deterministic QoS aspects and global constraints, at the same time. To deal with this situation, we propose in this paper a general framework that selects the Top-k compositions while managing the following requirements:

- The user's needs (global constraints, QoS optimization, number of services classes, control flow).
- The QoS fluctuations of web services over time.

Since the number of services per class might still be extremely high, it would be preferable to reduce the computational cost of the selection process. This aim can be ensured by introducing heuristics that select the best services of each class. It is worth mentioning that the complexity of this issue is known to be NP-Hard. [1, 2]

In summary, our main contribution referred to as the "selection module" (see figure1) can be described as follows:

1. Firstly, we rank the services of each class, according to the probabilistic dominance relationship shown in formula (5), and we retain only the Top-K services having the highest scores. This step aims to reduce the search space.
2. To rank the service compositions, we leverage an objective function based on the median QoS of the components of the composite solution (see formula (1))
3. We explore the search space constituted of the first K services of each class (see step I), by implementing a backtracking search (inspired from the constraint satisfaction problems) and we return the Top-K optimal compositions.

The reminder of the paper is organized as follows: the section 2 demonstrates a literature review on the QoS aware service selection issue. The third section specifies the problem, in the fourth section we show the proposed framework as well as the selection algorithms, and finally we present in the last section our conclusions and perspectives.

2 State of the Art

The service composition and selection has drawn a lot of attention during the past decade, The existing works either focus on global selection with deterministic/non-deterministic QoS [1,2,3,9,10,12,16] or local service selection with non-deterministic QoS [4,14,18].

The service selection with uncertain QoS is gaining a lot of interest in the service oriented computing area. Existing works such as [11, 14] leverage the dominance probability relationship to extract the most dominant services from a predefined dataset. In nutshell, the work presented in [11] extends the traditional concept of skyline [5] to cover the uncertain data (i.e notion of probabilistic skylines). To get the probabilistic skylines, the authors extract the services that have at least a percentage p to not be dominated by another component. As mentioned in [14], the P-skyline prefers noisy services to the detriment of consistent services. In [14] the authors propose a new concept called P-dominant skyline, which is less sensitive to noisy (inconsistent) services, in addition it is more suitable for including good services. Furthermore the authors leverage an R-tree [6] structure in order to efficiently extract the p-dominants services. In [4], the authors leverage the possibility theory in order to compute the dominants services. The possibility theory is preferred, when the probability distributions of QoS criteria is unknown or cannot be computed. As a results, the QoS attribute are modeled as possibility distribution. The authors also present two novel concepts: the possibility based skyline and the necessity based skyline, in addition they provide a mechanism to control the size of the skylines set. In [13], the authors propose an approach for computing the top k dominant compositions without taking into account the global constraints. The authors handle the QoS uncertainty by proposing the concept of dominance ability (which is based on the dominance

probability). In [17, 18] the authors present a set of formulas for estimating the uncertain QoS (mainly the execution time) of a composite service. To this end, they model each QoS metric of a component service as a probability distribution; in addition the composite service is represented as a graph that leverages several basic patterns (Sequential, Parallel, conditional, and Loop).

In the area of deterministic service selection, we can review a lot of approaches that handle the QoS as a non-varying phenomenon. In [3], the scholars handle both the functional aspects (inputs/outputs) and the non-functional aspects (QoS/global constraints), they propose an optimization framework based on the harmony search meta-heuristic .In [15, 16], the authors aim to avoid the user's implication (which usually assigns a set of numerical weights to the criteria) by focusing on skylines compositions (denoted C-SKY). In addition, the authors present a set of heuristics in order to accelerate the computation of C-SKY. To this end, they sort the skylines of each abstract class according to a predefined objective function (that sums all the QoS criteria), thereafter they explore the compositions space by scanning at first, the top services of each class. In [9] the authors leverage the harmony search meta-heuristic to get the near optimal compositions; the results can be further improved by tuning the meta-heuristic parameters. In [2] the authors address this issue by taking into account multiple control flow. Their main idea consists in extracting the skylines of each abstract class, thereafter; the authors create a hierarchical clustering of each skyline's set by leveraging the K-means Algorithm. Finally they explore the compositions space by combining the clusters heads and checking the global constraints fulfillment.

3 Problem Formalization

In this section, we will formalize the problem of Top-k dominant compositions under uncertain QoS. In what follows, we will assume a set of hypothesis and notations in order to simplify the problem specification:

- All the QOS attributes are positive (i.e all the positive attributes need to be maximized).
- The composition model is sequential.
- The QoS criteria of a composition are aggregated according to the sum function (such as reputation), if there are multiplicative criteria, then we replace them with their log value and we treat them as additive criteria. The other types of QoS criteria are not handled in this paper.
- n :is the number of abstract classes.
- $Cl_1, Cl_2, \ldots Cl_n$: are the set of abstract classes.
- m: is the number of services per classes.
- r: is the number of QoS attributes.
- l: is the number of service instances (i.e the number of QoS realizations or the sample size).

- QoS_{piju} : is the value of the p^{th} QoS attribute related to the u^{th} instance of the service $S_i \in Cl_j$
- $AVGQoS_{pij}$:the average QoS computed over all the instances of $S_i \in Cl_j$
- $b_1,b_2,..b_r$: are the user's global constraints (i.e the limits which need to be met by the QoS of the composition).
- $w_1,..w_r$: are the weight of the QoS criteria, the default value of each w_p is 1/r
- k: the size of the returned list (of compositions)
- The overall utility of a service composition $c = (x_1,x_2,\ldots,x_n)$ is computed as follows:
  - $U'(c) = \sum^{r}_{p=1} w_p * ((MedianQ'_p(c)-Qmin'(p))/(Qmax'(p) -Qmin'(p))$ (1)
  - x_1 (resp $x_2,\ldots x_n$): represents the id of the selected service related to Cl_1 (resp $Cl_2,\ldots,Cl_n$)
  - $Qmin'(p) = = \sum^{n}_{j=1} Qmin(j, p)$, (2)

    (the first normalization constant), were $Qmin(j, p) = MIN_{Si \in Clj, u \in \{1,..l\}} (QoS_{piju})$
  - $Qmax'(p) = = \sum^{n}_{j=1} Qmax(j, p)$, (3)

    (the second normalization constant), were $Qmax(j, p) = MAX_{Si \in Clj, u \in \{1,..l\}} (QoS_{piju})$
  - $MedianQ'_p(c) = \sum^{n}_{j=1} Median_{u \in \{1,..l\}} (QoS_{p\,xj\,j\,u})$ (4)

Since a component service S_{xi} is characterized by several QoS realizations, we choose the median QoS Value to evaluate its performance; consequently the composition c is also evaluated according to the median performance (see formula 1). We have chosen the median aggregation for the QoS realizations, because it is less sensitive to the variations and the outliers of the sample. In addition, we use the formula (5) to compare the compositions according to their degree of satisfying the global constraints. Roughly speaking, a composition c is ranked above another composition c' if the score of c with respect to (5) is higher than the score of c' with respect to (5). If c ties with c', then we order them according to formula 1 (which is also termed fitness or function U'(.)), the higher the score of U' the better the rank. Formally: c is ranked above c' iff :

$$\begin{cases} 1/r. \sum^{r}_{p=1} Pr(MedianQ'_p(c)) > 1/r. \sum^{r}_{p=1} Pr(MedianQ'_p(c')) \ \textbf{or} \\ 1/r. \sum^{r}_{p=1} Pr(MedianQ'_p(c)) = 1/r. \sum^{r}_{p=1} Pr(MedianQ'_p(c')) \text{ and } U'(c) \geq U'(c') \end{cases}$$

We also notice that the computational cost of formula (1) is O(r.n), (we assume that median values of the services are already computed), likewise the computational cost of formula (5) is O(r.n). In summary, our main objective is to select the Top-K compositions, $C_1,\ldots C_k$ which :

- Maximize the chance of satisfying the global constraints: $1/r. \sum^{r}_{p=1} Pr(MedianQ'_p(c_y)) \geqslant b_p)$, where $y \in \{1,..,k\}$. (5)

- Maximize the function U'(.).

4 Proposed Approach

In this section, we present our selection framework (shown in figure 1). It is constituted of three main modules:

The class management module: its purpose is to assign each service to a given abstract class (which represents the main functionality of the service such as: hotel booking, currency conversion, maps services…), the module also updates the classes.

The QoS management and integration module: it allows to store the fluctuating QoS of each service, the QoS data can be drawn from: the service provider itself (ex: the cost), the social networks (ex: the reputations) the third parties (ex: the latency…)

The selection module: Its main goal is to provide the Top K dominants service compositions for each user's request. This module is constituted of two algorithms (algorithm1: service ranking and algorithm2: backtracking search). The algorithm1 aims to reduce the search space of algorithm2, thereafter the backtracking search is executed in order to give the final compositions.

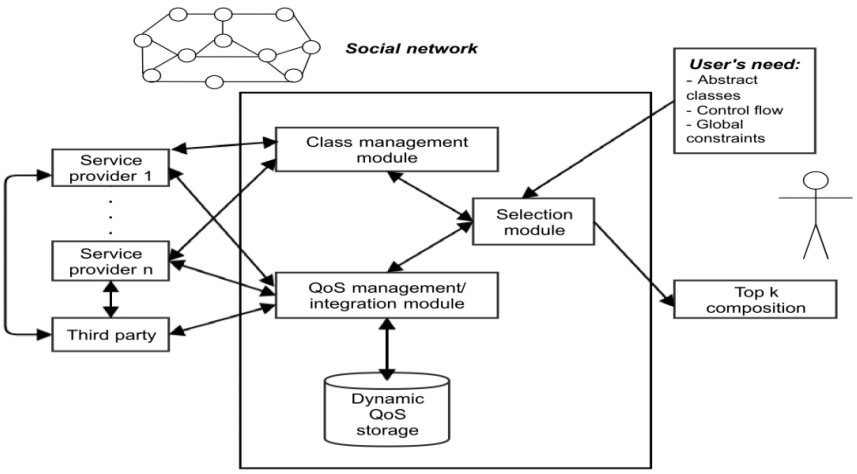

Fig. 1. Service selection framework

The service ranking sorts the services of each abstract class through the use of the probabilistic dominance relationship. We notice that the dominance relationship and its variants are widely used in the preference queries [8, 4] as well as the service discovery [7]. Simply speaking, we compare the QoS of each pair of services S_i, $S_{i'}$ with respect to the probabilistic dominance, thereafter we increment the ranking score of the wining service. The more the score is high the better the rank. The probabilistic dominance between two services $S_{i'}$ and S_i measures the average fraction of the instances of S_i that are weakly dominated by an instance of $S_{i'}$. It is given as follows:

$$\text{prob-dom}(S_{i'},S_i) = 1/l \sum^{l}_{u'=1} \text{individual-prob-dom}(u',i',i) \quad (6)$$

and individual-prob-dom(u',i',i) = (|{(QoS_{1iju},.., QoS_{riju})/ ($QoS_{1i'ju'}$,.., $QoS_{ri'ju'}$)>> (QoS_{1iju},.., QoS_{riju})}|/l) , u∈ {1,…l}

The relation >> denotes the weak dominance relationship, it is defined as follows:
Let X and Y be two vectors of R^r

X>>Y iff for each dimension i ∈ {1,..,r}: X(i)≥Y(i)
We assume that ≥ denotes "better than"
The pseudocode of algorithm1 is given below:

Algorithm1:ServiceRanking

Input: Cl_1,..Cl_n

Output: $RankedCl_1$,…$RankedCl_n$

1. For i=1to n Do $RankedCl_i$=<>;

2. For i=1 to n Do Begin

2.1 For y=1 to m Do score(y)=0;

2.2 For j=1 to m Do Begin

2.2.1 For j'=1 to m Do Begin

2.2.1.1 If (j!=j') Then Begin

2.2.1.1.1 If (probdom((S_j,$S_{j'}$)≥ probdom(($S_{j'}$,S_j)) Then

Begin

2.2.1.1.1.1 score(j)=0 score(j)+1;

End

End

End

End

2.3 $RankedCl_i$=decreasing-sort(Cl_i)

End

3. return <$RankedCl_1$,…$RankedCl_n$>

The explanation, of algorithm1 is given as follows:

In line 1, we initialize the ranked Class $RankedCl_i$ (with an empty structure).

In line 2.1, we initialize the ranking score of each service of the current class i.

In lines 2.2 up to 2.2.1.1.1, we compare each pair of services (S_j,$S_{j'}$) of the same class i, through the use of the probabilistic dominance formula (6).

In line 2.2.1.1.1.1, we update the score of S_j if it wins the test.

In line 2.3, we sort the elements of $RankedCl_i$ according to the scores updated in 2.2.1.1.1.1

We return the ranked classes in line 3.

It is worth noting that, the overall complexity of algorithm 1 is $O(nm^2.r.l^2 + n.mlogm)$, and the complexity of formula (6)is $O(r.l^2)$. The pseudo-code of algorithm2 is given below (we notice that the symbol <> denotes an empty structure):

Algorithm2:BacktrackingSearch

Input: $RankedCl_1,\ldots,RankedCl_n$

$b_1,b_2,..b_r$: global constraints

k: size of the results set, t: the minimum % of the preserved constraints.

Output: TopKCompositions

1.TopKCompositions =<>

2.For i=1 to k^n Do Begin

2.1 c=GetNextComposition($RankedCl_1,\ldots,RankedCL_n$);

2.2 degree= $1/r.\sum^{r}_{p=1}$ $Pr(MedianQ'_p(c)) \geqslant b_p$)

2.3 If (degree) $\geq$ t) Then Begin

2.3.1 If better (c, TopKCompositions) Then Begin

2.3.1.1 Update (c, TopKCompositions)

End

End

End

3. Return (TopKCompositions)

The explanation is given as follows:

In line 2, we explore all the possible compositions. In line 2.1, we get the current composition c. In line 2.2, we compute the fraction of satisfied global constrained. In line 2.3, we check that the fraction of the preserved global constraints is above the threshold. In line 2.3.1, we compare c with the existing "TopKComposition" elements through the use of formulas (5) and (1) (see section 3 for more details about the ordering of compositions). In line 2.3.1.1, we update the result TopKCompositions if c is better than an existing composition. We return the final result in line 3.It is worth noting that, the overall complexity of algorithm2 is $O(k^n(k.r.n+n+klogk))$.

5 Experiments

In this section we analyze the performance of our framework in terms of execution time and optimality. To this end, we conduct a set of experiments, with several configurations of parameters (see table 2). The experiments were conducted on a machine having an Intel I3 core 2.53GHz processor, 4 GB RAM, and running

Windows 7. The figures 1 up to 5 are related to algorithm2; however the figure 6 is related to algorithm1

Table 2. Parameters and examined values

Parameters	Values
Number of Tasks (n)	10, 15, 20
Number of Services (m)	100 to 1100
Number of QOS criteria (r)	2 to 10
Instances (l)	100 to 400
size of the result (k)	2 ,6,10

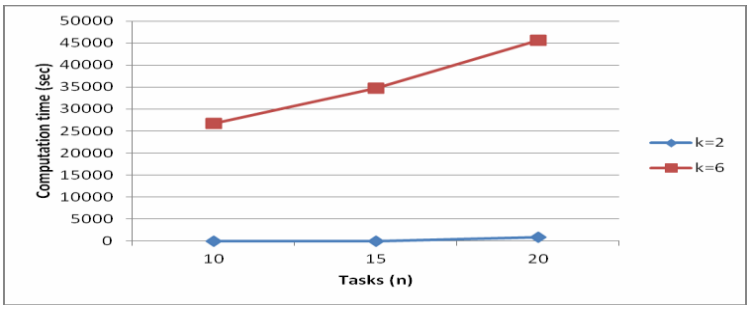

Fig. 2. CPU Time versus n (r=3, m=50, l=10)

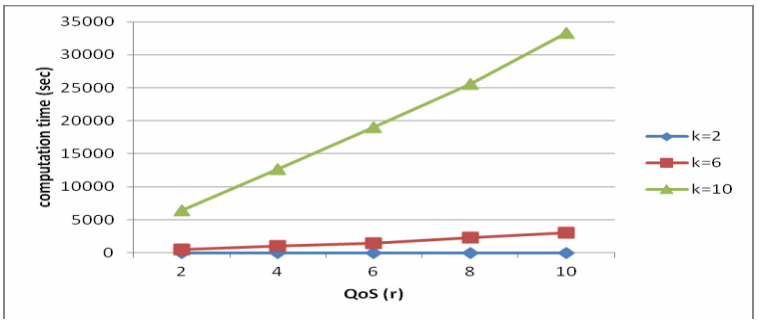

Fig. 3. CPU Time versus r (n=5, m=200, l=100)

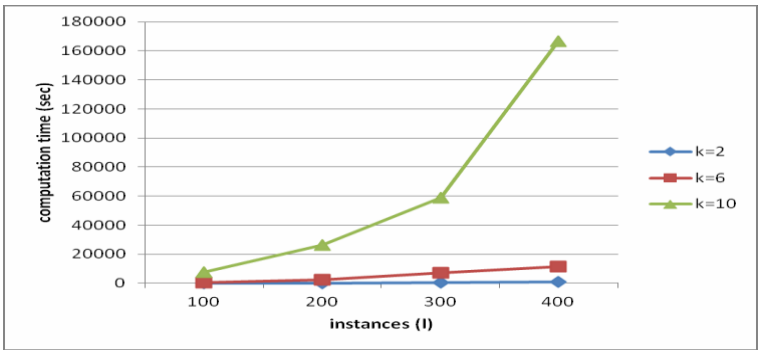

Fig. 4. CPU Time versus l (n=5, r=3, m=200).

The figure 2 shows the exponential growth of the execution time with respect to n. if k=2, then the execution time is acceptable for all values of n. however when k=6, the time overhead is not tolerable for n ≥ 10.

The figure 3 shows the impact of r over the execution time. We observe that all values of r, are tolerable for k=2 and k=6, however for k=10 and r ≥4 the execution time will be inacceptable (more than 10 minutes). The same observation is made for figure 4, for k=2 and k=6, the execution time is tolerable, however for k=10 and l≥ 200 the computational is not acceptable.

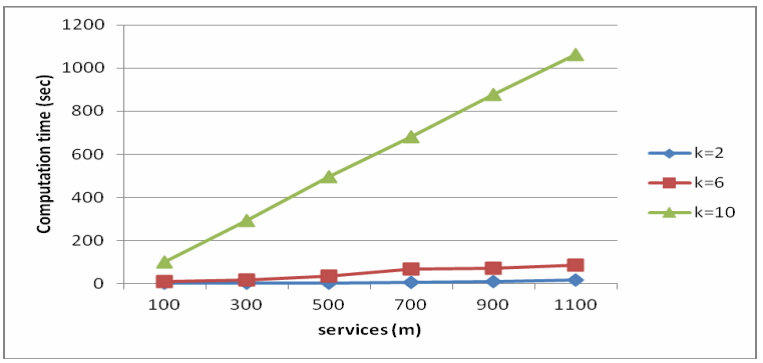

Fig. 5. CPU Time versus m (n=5, r=3, l=10).

As depicted in figure 5, the global search (algorithm 2) is not very sensitive to m, this is mainly due to the fact that the generation of compositions depends on the number of filtered services i.e. K. In what follows, we compare the effectiveness/efficiency of algorithm1 DSR (Dominance service ranking) with respect to the ranking based on average QoS termed ASR (Average service ranking). The latter computes the average QoS for each service $S_i \in Cl_j$, where $j \in \{1, 2,\ldots, n\}$. Thereafter ASR sorts the elements of Cl_j according to the sum of average QoS ,i.e the rank of each $S_i \in Cl_j$ is :

$$rank(i,j)= \sum_{p=1}^{r} AVGQoS_{pij}. \quad (7)$$

The more the score is high, the better the rank. The complexity of formula 6 is $O(r.l^2)$, consequently if we rank the services of Cl_j through the use of formula 6 ,

then the overall complexity will be $O(r.l^2.m + mlogm)$. The formula 7 is chosen instead of the dominance relationship, to alleviate the problem of curse dimensionality (i.e, with large r, the probability that a service s dominates another service s' is very weak).

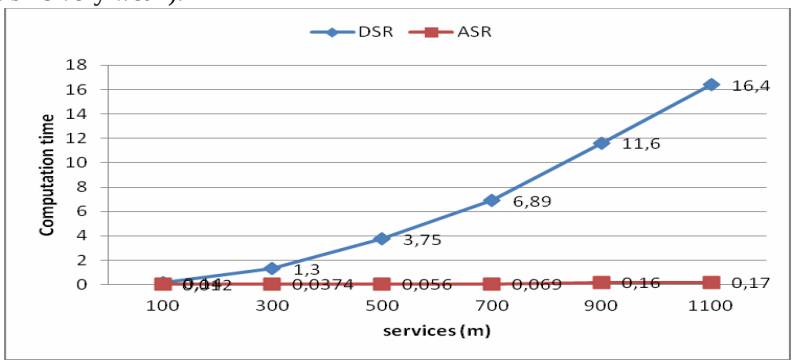

Fig. 6. CPU Time comparison (n=5, r=3, l=10).

As shown in figure 6, the ASR approach is better than DSR in terms of execution time. This is due to the fact that ASR is principally based on formula (7) which is only O(r.l), in addition ASR doesn't depend on m, however the DSR algorithm is based on formula 6 (which is $O(r.l^2)$), and depends on m.

According to table 3, we observe that the percentage of respected global constraints is the same for both approaches (ASR and DSR). We also notice a slight fitness superiority (i.e. the function U') of DSR with respect to ASR. This observation is valid for all values of K.

Table3. Performance comparison between DSR and ASR

Parameters	Solutions	Algorithms	Fitness	Respected Constraints (%)
n=5, r=3, m=200, l=300	TOP 2 Solutions	DSR+Algo2	0.4063	100
			0.4053	100
		ASR+Algo2	0.3823	100
			0.3813	100
	TOP 6 Solutions	DSR+Algo2	0.4075	100
			0.4073	100
			0.407	100
			0.4068	100
			0.4065	100
			0.4064	100
		ASR+Algo2	0.3997	100
			0.3992	100
			0.3989	100
			0.3984	100
			0.3983	100
			0.3982	100

6 Conclusion

In this work, we have investigated the problem of service selection under uncertain QoS. Our approach consists of two steps: the first one sorts the uncertain services according to the probabilistic dominance relationship, and the second one explores the search space by using a backtracking algorithm. The effectiveness/efficiency of the approach is confirmed with a set of experiments.

For future work, we will consider alternative sorting relationships (such as the dominance related to the necessity/possibility distributions). In addition we will adapt this framework to the selection of cloud services.

References

1. Alrifai, M., Risse, T.: Selecting Skyline Services for QoS-based Web Service Composition. In Proceedings of *WWW 2010*, April 26–30, 2010, Raleigh, North Carolina, USA. 2010.
2. Alrifai, M., Risse, T., Nejdl, W. : A hybrid approach for efficient Web service composition with end-to-end QoS constraints. *ACM Transactions on the Web (TWEB)* 6 (2), p7. 2012.
3. Bekkouche, A., Benslimane, S.M. , Huchard, M., Tibermacine, C., Hadjila, F., & Merzoug, M.: QoS-aware optimal and automated semantic web service composition with user's constraints. Service Oriented Computing and Applications, 11(2), 183-201.2017.
4. Benouaret, K., Benslimane, D., Hadjali, A.: Selecting skyline web services from uncertain qos. In: *2012 IEEE 9th international conference on Services computing (SCC)*, pp 523–530. 2012.
5. Borzsony,S. , Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of 17th international conference on data engineering. IEEE, pp 421–430.2001.
6. Guttman, A., R-trees: A dynamic index structure for spatial searching (Vol. 14, No. 2, pp. 47-57). *ACM*. 1984.
7. Hadjila, F., Belabed, A., Halfaoui, A.: Hybrid Web Service Discovery Based on Fuzzy Condorcet Aggregation. In : *East European Conference on Advances in Databases and Information Systems*. Springer, Cham, p. 415-427. 2015.
8. Hamiche, M., Drias ,H., Allel, H.: A strong-dominance-based approach for refining the skyline. In : *Programming and Systems (ISPS), 12th International Symposium on*. IEEE, p. 1-8. 2015.
9. Merzoug, M., Chikh, M.A., & Hadjila, F.: Qos-aware web service selection based on harmony search. In *4th International Symposium IEEE-ISKO-Maghreb: Concepts and Tools for knowledge Management*. (pp. 1-6). Alger.Algeria. 2014.
10. Minjung, K, Byungkook, O., Jooik, J., Kyong-Ho, L.: Outlier-robust web service selection based on a probabilistic QoS model. International Journal of Web and Grid Services. 12(2). pp.162 – 18.2016.
11. Pei, J., Jiang, B.Lin, X., and Yuan,Y.: Probabilistic skylines on uncertain data. In *VLDB Endowment*, Vienna, Austria.2007.
12. Rosenberg, F., Müller, M.B., Leitner, P., Michlmayr, A., Bouguettaya, A., & Dustdar, D.: Metaheuristic optimization of large-scale qos-aware service compositions. *In IEEE International Conference on Services Computing (SCC'10)* (pp. 97-104). IEEE. 2010.
13. Wen,S., Tang,C.,Li, Q., Chiu, D. K. W., Liu, A., Han, X.: Probabilistic top-K dominating services composition with uncertain Qos. In *Service Oriented Computing and Appication*, Volume 8, Issue 1, pp 91–103.2014.
14. Yu, Q., Bouguettaya,A.: Computing service skyline from uncertain qows. *IEEE Transactions on Services Computing*, 3(1):16– 29, 2010.
15. Yu, Q., Bouguettaya, A.: Computing service skylines over sets of services. In: *IEEE international conference on web services (ICWS).* pp 481–488. 2010.
16. Yu, Q., Bouguettaya, A.: Efficient service skyline computation for composite service selection. *IEEE Transactions on Knowledge and Data Engineering*, 25(4), 776-789. 2013.
17. Zheng, H., Zhao, W., Yang, J., Bouguettaya, A.: "QoS analysis for web service compositions based on probabilistic QoS." In Proceedings of *International Conference on Service-Oriented Computing*. Springer, Berlin, Heidelberg, 2011.
18. Zheng, H., Zhao, W., Yang, J., Bouguettaya, A.: QoS Analysis for Web Service Compositions with Complex Structures. *IEEE Trans. Services Computing* 6(3): 373-386 . 2013.