Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080557

Selecting Dynamic Skyline Services for QoS-based Service Composition

Jian Wu[*], *Liang Chen*[*] *and Tingting Liang*[*]

College of computer science and technology, Zhejiang University, Hangzhou, China

Received: 29 Sep. 2013, Revised: 27 Dec. 2013, Accepted: 28 Dec. 2013
Published online: 1 Sep. 2014

Abstract: With the growing adoption of web services on the Internet, service selection becomes an important issue of service-oriented computing (SOC). Appropriate services selection algorithm is the fundamental guarantee to compose complex services from single-function components effectively. The quality of selected component services is crucial for the performance of the service composition. Therefore, it has become a hot issue to select the best services from a set of services with similar functionality. Recently, skyline has been introduced to solve the problem by selecting skyline services as the best candidate services. In this paper, we focus on selecting skyline services in dynamic environment, where new services may appear, original services may invalidate and QoS of services may change. We propose a skyline service model as well as a novel skyline algorithm to maintain dynamic skyline services. An extensive performance study is propoesed to verify the effectiveness and efficiency of our approach.

Keywords: Keywords-service selection, dynamic skyline, QoS

1 Introduction

Service-oriented computing (SOC) is the computing paradigm that uses web services as fundamental elements to develop solutions. In this architecture, single-function applications are integrated to create new value-added and large-grained services. Service composition has attracted lots of researchers' attention and was applied in many domains, e.g. finances, workflow management and e-Business. With the rapid growing number of web services providing the equivalent functionality but differing quite a lot in QoS parameters, the problem of service composition is converted to the selection of best appropriate candidate services in regard to QoS.

Figure 1 presents an overview of the QoS-based service selection process. The service composition contains several tasks and each task can be accomplished by a set of services whose functionality is equivalent. QoS-based service selection is aimed to select best appropriate candidate services from each set to optimize overall QoS as far as possible subject to satisfying end-to-end QoS requirements.

However, it is inefficient to find the best combination satisfying end-to-end QoS constraints in an exhaustive search. For example, suppose there are four repositories, each of which has m web services and the number of QoS parameters of each service is n. For simplicity, web services in the same repository have equivalent functionality and the process of services composition is defined in advance. In practice, the time that it costs to find the best appropriate combination always exceeds the time constraints for real-time execution (try m^n combinations). With the number of functional equivalent services on the Internet increasing rapidly, this problem becomes especially serious and challenging. The web services search engine Seekda has reported that the number of web services over the last three years increases exponentially [1]. However if we can reduce the search space before the process of selection, then the time cost of selection will decreases substantially.

Recently, skyline approach has been introduced to service composition as the preprocessing before service selection [2],[3],[4]. QoS-based skyline services outperform other non-skyline services with regards to non-functional requirements. However, the service environment in [2],[3],[4] is all static, which is not practical in real application. Dynamic service environment, where new services may appear, original services may invalidate or the QoS of services may

* Corresponding author e-mail: {wujian2000,cliang,liangtt}@zju.edu.cn

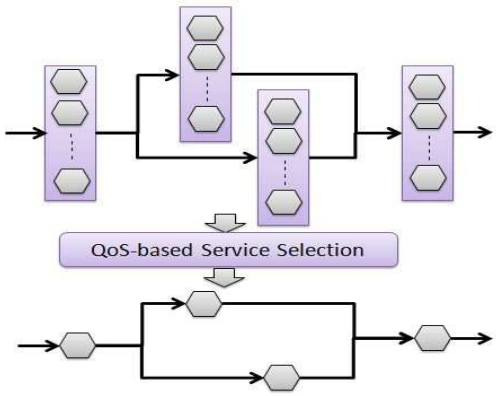


Fig. 1: Conceptual overview of QoS-based service selection

change, is more realistic and should be paid more attention.

In this paper, we investigate the selection of skyline services under dynamic service environment. As services are varying under dynamic environment, skyline services are changing. It is preferred to maintain QoS-based skyline instead of recalculating skyline services once service varies. In order to record each change, we propose a skyline service model called Paper-Tape which can locate the varying service rapidly. In particular, we propose a novel dynamic skyline algorithm which can handle skyline services under dynamic service environment. This paper has three-fold contributions as follows.

- –We address the problem of QoS-aware service selection, selecting skyline services to reduce the search space of service composition.
- –We consider the calculation of skyline services under dynamic service environment, and propose an innovative skyline approach which can maintain dynamic skyline services.
- –We evaluate our algorithm in an experiment on synthetic dataset of services with QoS information based on WS-Ben.

The rest of this paper is organized as follows. Section 2 reviews the related work, and Section 3 introduces the preliminaries. Section 4 shows the skyline service model which is called Paper-Tape. Section 5 presents the innovative skyline algorithm performing well under the dynamic services environment. Section 6 evaluates our method and the results demonstrate the benefits of our approach. Finally, Section 7 concludes the paper.

2 Related Work

The problem of service selection has been widely studied in lots of literatures during recent years. Zeng et al. [5] use linear programming techniques to select the optimal component services. In the work of Ardagna et al. [6], extended linear programming methods are proposed. However, with the number of services increasing, the performance of linear programming becomes worse. YehiaTaher et al. [3] proposed an approach focusing on minimizing impact on the ongoing in the process of the web service substitution and obtain good perform. In [4], the authors propose TCP-nets to solve the problem of web service selection based on preferences over non-functional attributes. However, the computation of TCP-nets takes account in all the services in repository and the effectiveness of TCP-nets approach hasn't been proved by experiments. The work in [7] has considered service competitiveness, but the solution in this work is just suitable for the situation where only one service attribution is subject to change. Then some literatures such as [8],[9] applied heuristic algorithm to solve the service selection problem. However, most of above methods are inefficient, because the range of candidate services they consider is vary large.

Skyline approach is complementary to these existing solutions. It could be used to decrease the range of selection, so that service selection process is more effective and efficient. Alrifai et al. [1] first apply skyline approach to compose web service, and prove the efficiency of this method. However, this work can only address the service selection problem when the repository is static, and this algorithm is not suitable for dynamic service environment. In order to select services under dynamic environment, we propose an innovative skyline algorithm which performs well under dynamic service environment.

Skyline algorithm can also be used to address the uncertainty of service. In the previous work [10], we utilize probability-skyline to deal with uncertainty in the process of services selection. Through this method, we can calculate the dominance probability between two services and set a threshold. We can put the services whose dominance probabilities exceeds the threshold in the probability skyline. In the work of [11], a p-R-Tree indexing structure and a dual-pruning scheme are proposed to calculate the p-dominant skyline services efficiently.

Skyline method, the fundamental approach we used to solve the problem of QoS-based services selection, is also referred to as the Pareto curve [12] or a maximum vector [13]. The skyline query is also related to some other well-known problems, such as nearest neighbor query [14], top-N problem [15], the contour problem [16], convex hulls [17], and multidimensional indexing. Kung et al. [13] first proposed skyline algorithm employing a D&C approach. Kossmann et al. [18] utilizes a D&C scheme for data indexed by an R-tree to gain the skyline based on a nearest neighbor approach. Recently, the most popular skyline algorithm is BBS (Branch and Bound Skyline) approach, which was proposed by Papadias et al.

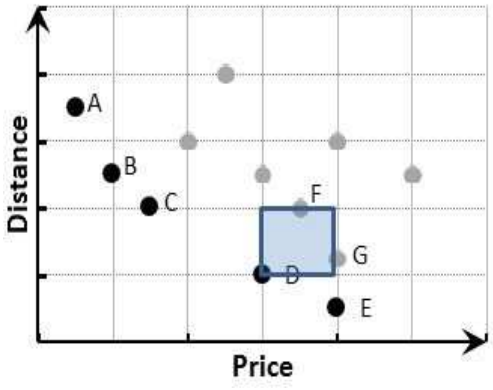


Fig. 2: Example of skyline

in [19]. It utilizes an R-tree structure and has been proven to be very effective and efficient.

3 Background

3.1 Skyline

Given a set of points in *n*-dimensional data space, point P_j is said to be dominated by P_i, if P_i is better or equal than P_j at all dimensions, and meanwhile P_i is better than P_j in at least one dimension [12]. In this *n*-dimensional space, all points which are not dominated by other points combined to a set named skyline. According to skyline query, we can find the best data which users required. Skyline queries are widely applicable to multi-criteria decision making applications. For sake of clarity, we formula the definitions of skyline and its dominance relationship as follows.

Definition 1.*(Dominance Relationship & Skyline) Given a set of points S in n-dimension data space, and two points* $x,y \in S$. *x dominates point y, denoted as* $x \prec y$,*if* $\forall i \in [1,n] : x_i \leq y_i$ *and* $\exists i \in [1,n] : x_i \leq y_i$. *Skyline, denoted as SK, is the set of points that are not dominated by any other points in S, i.e.* $SK = \{x \in S | \nexists y \in S : y \prec x\}$.

Definition 2.*(Dominance Region & Only Dominance Region) The Dominance region of a point x, denoted as* x_{dr}, *is the union of the point k which is the arbitrary point in the data space and is dominated by x, i.e.* $x_{dr} = \{\cup k | x \prec k\}$. *The only dominance region of point x, denoted as* x_{od}, *is the union of the point k which is only dominated by x, i.e.* $x_{od} = \{\cup k | x \prec k, \nexists p \in SK : p \neq x, p \prec k\}$.

For example, consider the classical scenario as in figure 2 where a user wants to find a hotel both near to downtown and cheap in price among a set of hotels. In figure 2, each hotel is described by two parameters, namely price and distance. Then, these hotels are projected into a 2-D space, and the coordinates of each point are corresponding to the values of the above two parameters. It is obvious that point A is not dominated by any other point, i.e. there is no other hotel has lower price and shorter distance than hotel A, so point A is in the skyline. The same holds for points B, C, D and E, which also belong to the skyline. However, the point F and G are not in the skyline, because they are both dominated by point D. The result skyline provides is a set of trade-offs among the parameters to be considered, which could be understood as a anti-correlated set. The blue region in Figure 2 is the only dominance region of point D, because the points in this blue region can only be dominated by point B. However, the points in upper or right boundary of only dominance region could be dominated by other skyline points.

3.2 QoS Constraints

In real application, users will have one or more requirements on service QoS. The aggregated service should satisfy users QoS constraints. The QoS of aggregated service, denoted as $Q_{overall} = \{q_1, q_2, ..., q_n\}$, should satisfy users QoS constraints, denoted as $Cons = \{c_1, c_2, ..., c_n\}$. The value of c_i represents the constraint value of the *i*-th attribute of the service (i.e., as for cost, constraint value stands for the maximum cost of service). As for satisfaction, each parameter of the aggregated service should satisfy the corresponding constraint. Service S satisfies its overall QoS constraints, denoted as $Cons(S) = True$, if $\forall i \in [1,n], q_i sat(c_i) = True$.

3.3 QoS Computation

The composite QoS is the emergence of all component services QoS. There are about three kinds of merge: 1) Sum; 2) Multiplication; 3) Minimum. The computation of composite cost belongs to Sum, as the cost of the aggregated service is the sum of all components cost, i.e. $Q_{overall}(c) = \sum_{i=1}^{m} Q_{si}(c)$ (m is the number of services). The computation of composite response time is the same as the computation of cost. In addition, the computation of throughput is Minimum, which means the value of composite throughput is the smallest one in all components throughput, i.e. $Q_{overall}(tp) = min_{1 \leq i \leq m} Q_{si}(tp)$. The computation of reliability belongs to the second kind, i.e. $Q_{overall}(re) = \prod_{i=1}^{m} Q_{si}(re)$.

4 Skyline Service Model

Before calculating skyline services, we should first project services into multi-dimension data space. Given a set of services $\check{S} = \{s_1, s_2, ..., s_m\}$, each service contains n non-functional attributes, denoted as $s_i\{q_1, q_2, ..., q_n\}$. In the *n*-dimension data space, each dimension corresponds to an

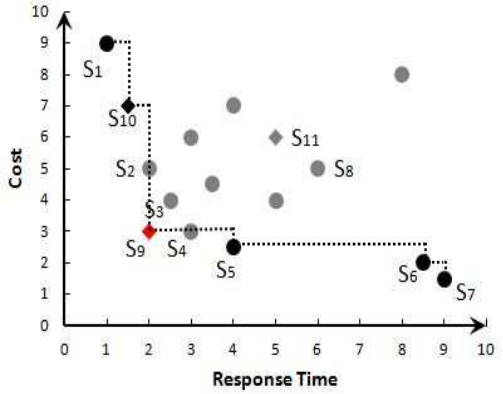


Fig. 3: Example of dynamic skyline services

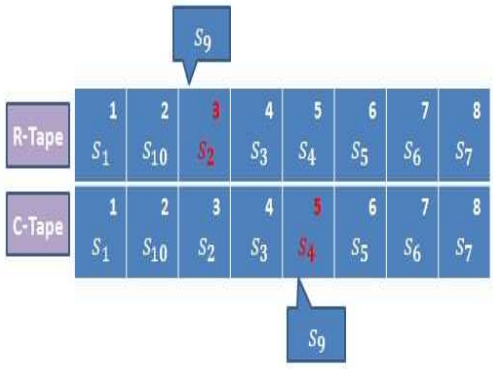


Fig. 4: Example of Paper-Tape model

attribute of service and the coordinate of service in this dimension equals to the value of corresponding attribute. In this way, we change the QoS-based service selection into the traditional scenario of skyline computation.

Many researchers [13],[18],[19] have studied the calculation of static skyline, and some efficient algorithms are proposed, e.g. Branch and Bound Skyline algorithm. In our work, we concentrate on the calculation of dynamic skyline services in the realistic scenario. In dynamic service environment, varying services will influence the skyline. For example, in the fig 3, we take round dots as original services and rhombus dots as varying services. For sake of simplicity, those three varying services are all new services added into the dynamic service environment and the varying order is s_{11} first, s_{10} second and s_9 third. Before varying services are added, original skyline services are $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$. The addition of s_{11} doesnt bring change to the skyline services as s_{11} is a dominated service. The addition of s_{10} affects skyline services as no services can dominate it. So the skyline services change to be $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$. The addition of s_9 also has an effect on skyline services as s_9 dominates three original services $\{s_2, s_3, s_4\}$. Therefore, the final skyline services are $\{s_1, s_{10}, s_9, s_5, s_6, s_7\}$.

It could be directly observed that whether the skyline change depends on the location of the varying service. Through comparing the location of varying service with each skyline service simply, we can get its influence on skyline. However, the cost of this method is too large on account of the times of pair-wise comparison. In order to obtain the effect of varying service, we propose a approach named as **Paper-Tape model** to locate the coordinate of varying service rapidly. Because the services information of each non-functional attribute is recorded in corresponding paper-tape, we can use an array to store a paper-tape. For simplicity‘s sake, we only consider two attributes: response time and cost, and name them as R-Tape and C-Tape respectively.

In the pre-process stage, we rank the services in increasing order of services response time, and label each service with its serial number, i.e. before adding new services, the response time of s_3 is the third smallest one in skyline services, so the label of s_3 in R-Tape is 3. In particular, the two queues of services in C-Tape and services in R-Tape are the same, i.e. the label of s_3 in C-Tape is 3 too. Therefore, the label of skyline service in R-Tape is the same to the one in C-Tape. According to the definition of skyline, the values in C-Tape is in descending order.

When we first insert service s_9 into service registries and the latest skyline services are $\{s_1, s_{10}, s_2, s_3, s_4, s_5, s_6, s_7\}$, as showed in fig 3, we can find s_9 has its response time equals to s_2 and its cost equals to s_4. Thus we can get its paper-tape model as fig 4 shows.

The labels of varying service s_i in the two tapes are denoted as $L_R(i)$ and $L_C(i)$ respectively. For example, the label of service s_9 is $L_R(9) = 3$, $L_C(9) = 5$. In particular, if the value of s_i is between the value of s_i (label=k) and s_j (label=$k+1$) in R-Tape, then $L_R(i) = k + 0.5$. In particular, if the value of new service is less than the smallest one in R-Tape, then the label is 0.5. Also, if the value of new services is larger than the largest one in R-Tape, then the label is $n + 0.5$.

Theorem 1. *Paper-Tape Dominance. If $L_R(i) < L_C(i)$, this implies s_i dominates one or more original skyline services and is a skyline service. If $L_R(i) = L_C(i)$, this implies s_i doesnt dominate original skyline service and is skyline service. If $L_R(i) > L_C(i)$, then s_i is a dominated service.*

Proof: If $L_R(i) < L_C(i)$, then there are one or more services s_j that $L_R(j) > L_R(i)$ and $L_C(j) < L_C(i)$. Because the value order in R-Tape is ascending, so $L_R(j) > L_R(i)$ signifies the response time of s_j is longer than s_i. And $L_C(j) < L_C(i)$ means the cost of s_i is less than s_j because in C-Tape the value order is descending. Due to the cost and response time of s_i are both less than those of s_j, we can get s_j is dominated by s_i.When

$L_R(i) = L_C(i)$, we can get there is no service that dominates s_i. Therefore, s_i is skyline service. $L_R(i) = L_C(i)$, we can get these is a service s_j that dominates s_i. So s_i is a dominated service and has no effect on original skyline.

5 Dynamic Skyline Service Algorithm

There are about three kinds of service changes to be considered: 1) new service appears; 2) old service invalidates; 3) value of QoS changes. In particular, the third case could be considered as the combination of the second and the first one. Therefore, the proposed algorithm DSCA contains two parts, DSCA-add is used to deal with the appearance of new services and the other one DSCA-delete is for the invalidation of original services. The following algorithm 1 is for the scenario that a new service appears. In particular, the initiation of R-Tape and C-Tape are considered as preprocessing.

Algorithm 1 DSCA-add

Input: R-Tape,C-Tape,$s_i(q_r,q_c)$
Output: R-Tape,C-Tape
1: $L_R(i) \leftarrow$ compare q_r with values in R-Tape
2: $L_C(i) \leftarrow$ compare q_c with values in R-Tape
3: **if** $L_R(i) < L_C(i)$
4: Delete services belong to $[L_R(i), L_C(i)]$
5: Insert $s_i(q_r,q_c)$
6: Update R-Tape and C-Tape
7: **end if**
8: **else if** $L_R(i) = L_C(i)$
9: Insert $s_i(q_r,q_c)$
10: Update R-Tape and C-Tape
11: **end else if**
12: **else**
13: end else

We first get the coordinate of added service in R-Tape and C-Tape through comparing q_r and q_c with corresponding Tape (Line 1, 2); If $L_R(i) < L_C(i)$, according to Theory 4, we can know this service dominate some or more original skyline services. We delete those dominated services in previous skyline ,add new service to skyline and renew R-Tape, C-Tape(Line 5, 6); if $L_R(i) = L_C(i)$, we can know the new service doesnt have dominance relationship with original skyline services, thus we can simply add the service to skyline and update R-Tape, C-Tape (Line 9, 10); if $L_R(i) > L_C(i)$, we can get that the new service is dominated by some original skyline services, thus we do nothing to the skyline (Line 12, 13).

As for the invalidation of service, two cases should be considered: 1) the invalidated service s is a skyline service. In this case, the services in ss only dominate region are not dominated by any other services out of this region. Therefore, the local-skyline which is computed from services in ss only dominated region will be part of new skyline. we compute the local skyline, delete the invalidated service and add the local skyline to previous skyline, and then we get the full skyline. 2) the invalidated service is not skyline service. In this case, nothing is needed to do.

Algorithm 2 DSCA-delete

Input: R-Tape,C-Tape,$s_i(q_r,q_c)$
Output: R-Tape,C-Tape
1: $L_R(i) \leftarrow$ compare q_r with values in R-Tape
2: $L_C(i) \leftarrow$ compare q_c with values in R-Tape
3: **if** $L_R(i) = L_C(i)$
4: $\check{s} \leftarrow$compute the local skyline of s_is only dominance region
5: Deletes$_i(q_r,q_c)$,insert $\check{s}$ into previous skyline
6: Update R-Tape, C-Tape
7: **end if**
8: **if** $L_R(i) \neq L_C(i)$
9: **end if**

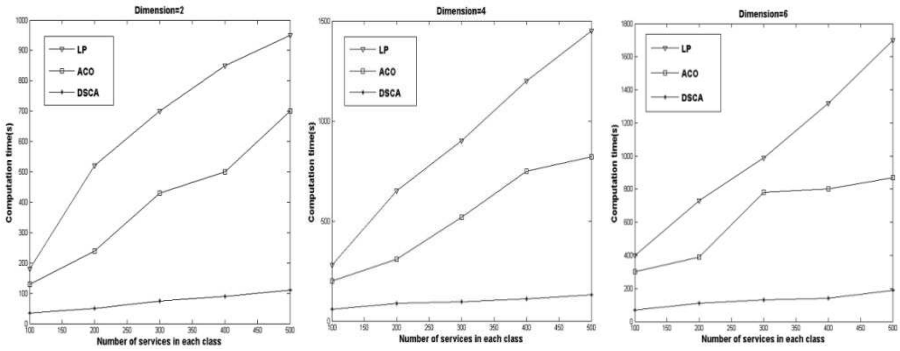
In DSCA-delete algorithm, we first locate the invalidated service through retrieving R-Tape and C-Tape(line1,2). If $L_R(i) = L_C(i)$, according theory 4, the invalidated service is skyline service. Therefore, we compute the local skyline of the invalidated service, delete the invalidated service from skyline and update two Tapes (line 3,4,5,6). If $L_R(i) \neq L_C(i)$, according theory 4, the invalidated service is not skyline service. In this case, nothing is needed to do (line 8,9).

6 Experiment

Weve implemented the algorithms described in Section 5 in C++. The experiments were conducted on a PC with Intel Core Duo E7400 2.8GHz CPU and 2GB main memory running Windows 7 operating system. The dataset used for experiment s is generated by WS-Ben, which is a web services generator and produces scaled WSDL documents with user-defined characteristics and sample queries to run a users proposal.

6.1 Overall Composition Performance

In this section, we evaluate the overall performance of dynamic QoS-based service selection. The dataset used for evaluation is generated by WS-Ben platform. WS-Ben is a web service generator which produces scaled WSDL documents with user-defined characteristics and sample queries to run a users proposal. For the purpose of evaluation, we consider a scenario, where a composite application concludes 10 different service classes and each class has 500 alternative functionally-equivalent services. In particular, the service environment is dynamic, which means there will be new services appear



(a) (b) (c)

Fig. 5: Overall performance

and old services invalidate. The selection ends until the best optimal combination found. We use WS-Ben to generate these 5000 services and collect their QoS data for experiments. We compare the performance of the following service composition method:

–*LP (Linear Programming)*: this is the standard global optimization method with all service candidates represented in the linear programming model.
–*ACO (Ant Colony Optimization)*: A heuristic algorithm used for optimization and has been applied in the composition of services by some researchers.
–*DSCA*: This is the algorithm we proposed in this paper, which could maintain skyline services under dynamic environment.

We measure the computation time required by each of the above methods for selecting the best combination for service composition. The results are presented in figure 6. In fig 5 (a), the number of non-functional attributes we consider is 2. We can get that the efficiency of DSCA method outperforms the other two methods drastically. DSCA only select the best combination from dynamic skyline services while the other two methods take all services into consideration. We consider 4 non-functional attributes in fig 5(b) and 6 in fig 5(c). However when the number of attributes that we consider increases, the results are the same.

6.2 DSCA Performance

In this section, we focus on evaluating the performance of DSCA in computing dynamic skyline services. The dataset used for evaluating is also generated by WS-Ben, but the size of dataset increases to 10,000. We adopted three scenarios for evaluating the performances:(a) in this scenario, varying services are the whole new services which are inserted into this service environment; (b) in this scenario, varying services are all invalidated services; (c) the probability that varying service is new service and the one that varying service is invalidated service are equal.

We compare the efficiency of following approaches that can calculate dynamic skyline.

–*Repeat*: A naive method which computes the skyline while there is a change.
–*LookOut*: An efficient continuous skyline algorithm proposed by D. McLain [16]. It can adjusts dynamic skyline through two sub-skyline.
–*DSCA*: A dynamic skyline algorithm proposed in this paper. It estimates the influence of varying services and adjusts skyline correspondingly.

The *Repeat* method includes two steps to maintain a dynamic skyline: (1) Compute the whole services to get the first skyline; (2) If there is a change, computing whole services again to get the skyline. *LookOut* and *DSCA* approaches both maintain dynamic skyline through adjusting skyline while these is a change. Therefore, we hold that the performances of *LookOut* and *DSCA* should be better than the one of Repeat. In particular, the processes of *LookOut* and *DSCA* are similar. However, we believe the performance of *DSCA* should be better than the one of *LookOut*. Because in the process of measuring the influence of varying service, *LookOut* uses enumeration approach while *DSCA* locates it quickly through Paper-tape model.

The experiment results bear out our supposition. In figure 6(a), we add 10 new services into the set of basis services. The result shows that the performance of DSCA is the best and the one of Repeat is the worst. In figure 6(b) the number of basis services is 10000, we change the number of new services. DSCA approach also outperforms the Repeat and LookOut approaches. Figure 6(c) and (d) shows the results in the second scenario where varying services are all invalidated services. As like as the first scenario, the DSCA method performs better than Repeat method and Outlook method. In fig 6(e) and (f), we get a view of the performance of these three methods in the third scenario. In some ways, the

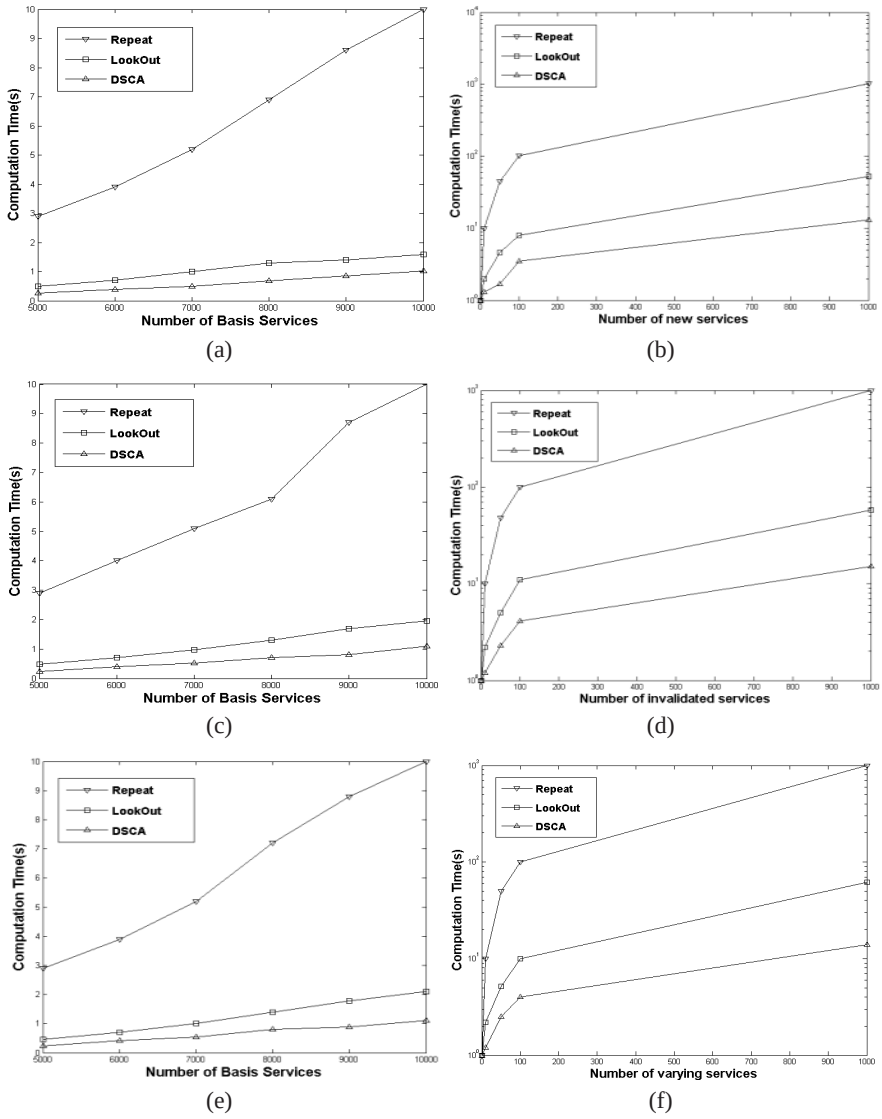


Fig. 6: Performance of three methods

performance in the third scenario is the aggregation of performances of the first and second scenario. It is obvious that the performance of DSCA is still greater than the performances of Repeat and LookOut.

We also measured the average computation time required by DSCA approach for computing dynamic skyline, varying the number of attributes from 2 to 6. In addition, the size of dataset is 10000. The result of this experiment is presented in Figure 7. It is obvious the computation cost increases with the increase of attributes or the number of varying services. However, we could get that the amplification of computation time is acceptable while the number of attributes increase. The scalability of Paper-Tape in multi-dimension guarantees the scalability of DSCA. We also evaluate the relation between basis service number and computation time of DSCA. In order to get this relation, we vary the number of basis services but fix the number of varying services. The result is presented in figure 8. As the increase rate of skyline services is minor while the number of basis services increases, the amplification of cost is also minor too.

7 Conclusion

In this paper, we apply an innovate method to deal with the problem of QoS-based dynamic service selection. We take the advantage of skyline to decrease the range of choices effectively without pruning potential candidates. In order to deal with dynamic service environment, we

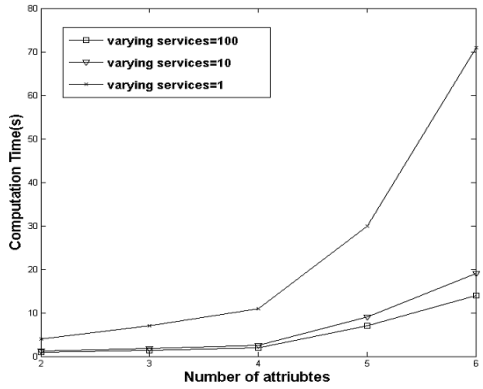


Fig. 7: Performance VS. dimension

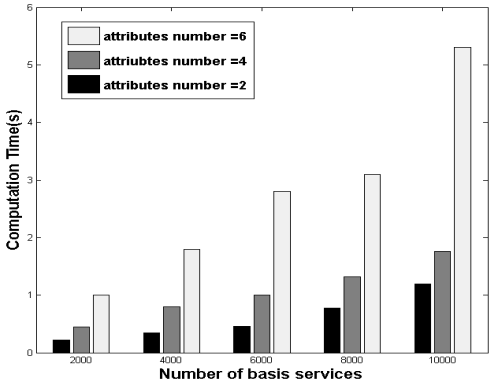


Fig. 8: Performance VS. service number

propose a Paper-Tape model which could estimate the influence of varying service quickly as well as an algorithm to maintain skyline services based on Paper-Tape model. Our extensive experiment shows that our approaches outperform existing methods in efficiency.

In the future work, we aim to examine the performance of our approaches for the recommendation in P2P network. We select sources according to some attributes of them in P2P network, which is similar to selecting services from repositories. Therefore, skyline method should be also suitable in P2P network.

Acknowledgment

This research was partially supported by the National Technology Support Program under grant of 2011BAH16B04, the National Natural Science Foundation of China under grant of 61173176, National Key Science and Technology Research Program of China 2013AA01A604.

References

[1] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *Proceedings of the 19th international conference on World wide web*, pp. 11–20, 2010.

[2] Y. Liu, A. H. Ngu, and L. Z. Zeng, "Qos computation and policing in dynamic web service selection," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers and posters*, pp. 66–73, 2004.

[3] Y. Taher, D. Benslimane, M.-C. Fauvet, and Z. Maamar, "Towards an approach forweb services substitution," in *Proceedings of the 10th International Database Engineering and Applications Symposium*, pp. 166–173, 2006.

[4] G. Santhanam, S. Basu, and V. Honavar, "Web service substitution based on preferences over non-functional attributes," in *Proceedings of the 2009 IEEE International Conference on Services Computing*, pp. 210–217, 2009.

[5] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. On Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[6] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions onSoftware Engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[7] M. Papazoglou, "Service-oriented computing: concepts, characteristics and directions," in *Proceedings of the Fourth InternationalConference on Web Information Systems Engineering*, pp. 3–12, 2003.

[8] I. Maros, *Computational Techniques of the Simplex Method*. International Series in Operations Research and Management Science,, Springer, 2003.

[9] T. Yu, Y. Zhang, and K. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, 2007.

[10] L. Chen, J. Wu, and R. Jia, "Recommendation on uncertain services," in *Proceedings of IEEE International conference on web services (ICWS) 2010*, pp. 683–684, 2010.

[11] Q. Yu and A. Bouguettaya, "Computing service skyline from uncertain qows," *IEEE Transaction on Service Computing*, vol. 3, no. 1, pp. 16–29, 2010.

[12] C. Papadimitrious and M. Yannakakis, "Multiobjective query optimization," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 52–59, 2001.

[13] H. Kuang, F. Luccio, and F. Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM*, vol. 22, no. 4, pp. 469–476, 1975.

[14] G. Hjaltason and H. Samet, "Distance browsing in spatial database," *ACM Transactions on Database Systems*, vol. 24, no. 2, pp. 265–318, 1999.

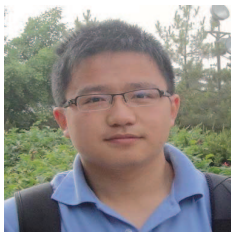
[15] M. Carey and D. Kossmann, "On saying enough already! in sql," in *Proceedings of ACM SIGMOD International Conference on Man-agement of Data*, pp. 219–230, 1997.

[16] D. McLain, J. Patel, and W. Grosky, "Efficient continuous skyline computation," *Information Sciences*, vol. 177, no. 17, pp. 3411–3437, 2007.

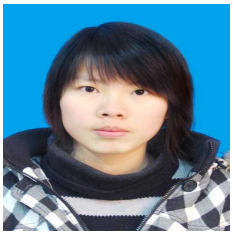
[17] C. Bohm and H. Kriegel, "Determining the convex hull in large multidimensional database," in *Proceedings of Third International Conference, DaWaK 2001 Munich*, pp. 294–306, 2001.

[18] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: an online algorithm for skyline queries," in *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 275–286, 2002.
[19] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 467–478, 2003.
[20] M. Papazoglou and D. Georgakopoulos, "Service-oriented computing," *Communications of the ACM*, 2003.
[21] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pp. 881–890, 2009.
[22] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, no. 3, pp. 993–1022, 2003.
[23] W3C, "Web services description language (wsdl) 1.1." http://www.w3.org/TR/wsdl, 2001.
[24] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pp. 448–456, 2011.
[25] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition," in *Proceedings of 12th International Conference on World Wide Web (WWW)*, pp. 411–421, 2003.
[26] S. Ran, "A model for web services discovery with qos," *ACM SIGecom Exchanges*, vol. 4, no. 1, pp. 1–10, 2003.
[27] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction for web services via collaborative filtering," in *Proceedings of International Conference on Web Service*, p. 439C446, 2007.
[28] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Trans. Service. Computing*, vol. 4, no. 2, p. 140C152, 2011.
[29] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *Proceedings of International Conference on Web Service(ICWS)*, pp. 9–16, 2010.
[30] Z. Zheng, H. Ma, M. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Trans. Service. Computing*, vol. 1, no. 1, 2012.
[31] W. Lo, J. W. Yin, S. Deng, Y. Li, and Z. H. Wu, "An extended matrix factorization approach for qos prediction in service selection," in *Proceedings of IEEE 9th International Conference on service computing*, pp. 162–169, 2012.
[32] S. R and M. A, "Probabilistic matrix factorization," *Advances in Neural Information Processing Systems (NIPS)*, pp. 1257–1264, 2008.
[33] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.
[34] H. Shan and A. Banerjee, "Generalized probabilistic matrix factorizations for collaborative filtering," in *Proceedings of the 2010 IEEE International Conference on Data Mining*, p. 1025C1030, 2010.
[35] Z. B. Zheng, Y. L. Zhang, and M. R. Lyu, "Distributed qos evaluation for real-world web services," in *Proceedings of the 8th International Conference on Web Services (ICWS)*, pp. 83–90, 2010.
[36] Y. L. Zhang, Z. B. Zheng, and M. R. Lyu, "Exploring latent features for memory-based qos prediction in cloud computing," in *Proceedings of the 30th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pp. 1–10, 2011.
[37] J. Wu, L. Chen, Y. P. Feng, Z. B. Zheng, M. C. Zhou, and Z. H. Wu, "Predicting quality of service for selection by neighborhood-based collaborative filtering," *IEEE Transactions on Systems,Man,and Cybernetics:Systems*, vol. 43, no. 2, pp. 428–439, 2013.
[38] M. Serhani, R. Dssouli, A. Hafid, and H. Sahraoui, "A qos broker based architecture for efficient web services selection," in *Proceedings of the 2005 IEEE International Conference on Web Service*, pp. 113–120, 2005.
[39] L. Chen, Y. Feng, J. Wu, and Z. Zheng, "An enhanced qos prediction approach for service selection," in *Proceedings of the 2011 IEEE International Conference on Services Computing*, pp. 727–728, 2011.
[40] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovianworkload characterization for qos prediction in the cloud," in *Proceedings of the 2011 IEEE International Conference on Cloud Computing*, pp. 147–154, 2011.

Jian Wu received his B.S. and Ph.D. Degrees in computer science from Zhejiang University, Hangzhou, China, in 1998 and 2004, respectively. He is currently an associate professor at the College of Computer Science, Zhejiang University, and visiting professor at University of Illinois at Urbana-Champaign. His research interests include service computing and data mining. He is the recipient of the second grade prize of the National Science Progress Award. He is currently leading some research projects supported by China National Natural Scientific Foundation and National High-tech R&D Program of China (863 Program).



Liang Chen received the B.S. degree in computer science from Zhejiang University, Hangzhou, China, in 2009. He is currently working toward the Ph.D. degree in the College of Computer Science, Zhejiang University. His publications have appeared in some well-known conference proceedings and international journals. He also served as a Reviewer for some international conferences and journals (CIKM, TSMC, TSC, ICWS, etc). His research interests include service computing and data mining. He received the award of "Excellent Intern" from Microsoft Research Asia in 2010.



Tingting Liang received the B.S. degree in the school of science ,Jiangnan University, Wuxi, China, in 2013. She is currently working toward the Ph.D degree in the College of Computer Science, Zhejiang University, Hangzhou. Her research interests include service computing, machine learning and data mining.