Efficient Multi-User Service Selection Based on the Transportation Problem

Adrian Satja Kurdija, Marin Silic, Goran Delac, Klemo Vladimir, and Sinisa Srbljic

University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia
{adrian.kurdija, marin.silic, goran.delac, klemo.vladimir, sinisa.srbljic}@fer.hr

Abstract. Modern service selection in a cloud has to consider multiple requests to various service classes by multiple users. Taking into account quality-of-service requirements such as response time, throughput, and reliability, as well as the processing capacities of the service instances, we devise an efficient algorithm for minimum-cost mapping of mutually independent requests to the corresponding service instances. The solution is based on reduction to transportation problems for which we compare the optimal and a suboptimal but faster solution, investigating the tradeoff. In comparison to the alternative service selection models, the evaluation results confirm the efficiency and scalability of the proposed approach(es).

Keywords: Quality of Service, Service Selection, Transportation Problem

1 Introduction

Modern web applications make use of multiple services with different functionalities [1] and can in effect be described as *service compositions*. An application can be simultaneously used worldwide, and for each of the functional service requests it makes, there are usually many possible candidates – corresponding service instances on cloud servers all over the world [2]. Equivalent services corresponding to the same functional description are referred to as a *service class* or an *abstract service* [3]. Application's quality of service (QoS) may depend on various user-specific, service-specific, and environment-specific parameters [4]. User-specific parameters include the user's location, network and device capabilities, and usage profiles. Service-specific parameters (for a service instance) include its location, computational complexity, and system resources (e.g. CPU, RAM, disk, and I/O operations). Environment-specific parameters include service provider load and network performance at the time of the invocation. In a system with many users consuming different applications with requirements on various criteria such as response time, reliability, and price per service request [5], there is a need of a robust model which will optimize the service selection in the

cloud and, thus, provide an answer which service instances will be invoked by which applications instances (users).

Approaching this problem, many considerations must be taken into account. Some service instances are infeasible to some users (due to QoS requirements of corresponding requests); some are more reliable, but the cost of calling them is higher; some of them might not be available due to their "popularity" among other users or other applications. As for the last point, we must respect the *processing capacity* [6] of each service instance (the maximum number of requests it can process in a given time frame) which is not a trivial task when there are multiple users with requests for the same service class. In cases like this, greedy algorithms will not work: selecting the closest, the most reliable or the cheapest service instance for each request will generally break the processing capacity limit of some service instances, and might also fail with respect to other criteria.

In this paper, we propose an efficient and scalable model which incorporates the aforementioned considerations. Namely, the model considers multiple user requests, each having required performance values (criteria) for QoS. The typical criteria, considered in this paper, are response time, reliability, and throughput, while total price of invoked services is sought to be minimized. The model is easily extendable to other QoS attributes. The model is responsible of selecting the service instances to facilitate the given requests, considering their respective processing capacities and their respective values of response time, reliability, and throughput for all considered users, in order to satisfy the requests' requirements and to minimize the total price.

Minimization is done by reducing the problem to several mutually independent transportation problems. This is a time-efficient optimization, done by solving several transportation problems in parallel, with two variations of the proposed algorithm with time/accuracy tradeoff. We have conducted several experiments to confirm the correctness and efficiency of the proposed approach(es) and to compare them with alternative approaches. The obtained evaluation results strongly demonstrate that the proposed approach(es) meets its design requirements.

The paper is organized as follows. Sect. 2 discusses the related work, Sect. 3 describes a system model and defines the problem in precise terms, Sect. 4 describes the proposed algorithm, experimental results are presented in Sect. 6 and conclusions are given in Sect. 7.

2 Related Work

There has been a lot of work on QoS-based service selection, covering different aspects and problem variations. Many models focus on *predicting* the user-service QoS values. The model we propose assumes they are already known or predicted with reasonable accuracy, focusing on the constrained selection phase instead.

Multi-criteria service selection is complex enough even when focusing on a *single user's* composite service pipeline. To name a few approaches, [7] employs iterative multi-attribute combinatorial auction between services providers. This

approach was further improved by [8] using an incentive mechanism. BigDataspace service selection was tackled by [9], taking into account both qualitative and quantitative user QoS preference with the service trust. Recently, [10] studied the problem from a general Pareto-optimal angle, seeking to reduce search space in service composition. A polynomial time approximation for Pareto optimality was done by [11]. Similar work was done by [12], trying to overcome the limitations of Pareto optimality. Approach by [13] selects representative services adaptively: it divides the value range of each quality attribute into sub-ranges, considering the QoS values of a sub-range as local constraints, and selects the appropriate services from the divided QoS ranges so they can maximize the utility of a user task. Some recent papers such as [14] took into account the possible correlation of services' QoS attributes in a composition, while the probabilistic QoS values are considered in [15]. Recently, [16] developed a mobility-aware approach.

To find the *optimal* solution for a single user's service selection with composite service pipeline, [17] reduced the problem to a Multiple Choice Knapsack Problem (MCKP) which is NP-hard, but solvable in pseudo-polynomial time. However, they considered only a single execution path in a composite pipeline. Taking into account a directed graph which contains all execution paths in a dynamic pipeline, the problem becomes a Multiple Choice Multiple Dimension Knapsack (MMKP), which is more complex and is usually solved by mixed integer programming (MIP) [18, 19]. Others tried to improve on this exponential solution by reducing the size of the NP-hard problem instance: for a single user, papers such as [20] decompose each QoS global constraint c_0 into a set of local constraints $c_1, \ldots, c_n$ where n is the number of abstract service classes in the composite pipeline. For multiple users, [21] formed a reduced service set by dismissing the services dominated by another service, while others reduced the problem size by clustering the services [22] or both users and services [23]. Still, future scalability issues call for a polynomial solution.

With some framework similarities to the present work, polynomial work on service selection was done by Wang et al. [6] (multiple users, single service class) and Jin et al. [24] (multiple users, multiple service classes), on which some efficiency comparisons will be made.

3 System Model

The proposed service selection algorithm assumes a set U of active users, a set F of abstract services (service classes) and a set I_f of service instances in the cloud corresponding to each service class (functionality) $f \in F$. Each service instance $i \in I_f$ has a processing capacity PC_i, a maximal number of requests it can handle in the given time frame. Each user $u \in U$ has a set of (possibly one) mutually independent requests R_u.

Each request $r \in R_u$ has a corresponding service class $f(r) \in F$ and inherits from its corresponding application the required maximum response time RT_r, minimum reliability Rel_r, and minimum throughput TR_r. The way in which

these values are derived from, for example, the global application-level SLA (service-level agreement) is beyond the scope of the paper: this problem is referred to as a *decomposition of global QoS constraints* by e.g. [20], which deals with it by considering quality levels of different service classes; a similar and more recent approach is given in [13].

Let $I = \bigcup I_f$ denote the set of all service instances. Let RT, Rel, TR, and Pr be four matrices of size $|U| \times |I|$ describing the connections between users and service instances. Namely, RT_{ui}, Rel_{ui}, TR_{ui}, and Pr_{ui} are response time, reliability, throughput, and price per request for invocation of service instance $i \in I$ by user $u \in U$. The infeasible user-service connection can be described by very large values of RT and Pr (denoted by $\mathcal{E}_{inf}$), and the values of Rel and TR close or equal to zero (denoted by ε_0). These constants can be set to e.g. 10^9 and 10^{-9}.

We do not further explore how matrices RT_{ui}, Rel_{ui}, TR_{ui}, and Pr_{ui} are obtained. In general, these values can be derived either by prediction or by estimating values using service monitoring approaches. For instance, a variety of service monitoring approaches have been described in the literature [25], [26]. Although they can be effective, these methods can have limitations in practice as frequent service polling for the purposes of reliability estimation can lead to degradation in service performance. One way to mitigate the issues present in service monitoring is to utilize predictive methods, either by fitting the collected data to predefined models or by using statistical methods [27], [28], [29].

Other criteria can also be considered, which does not change the proposed algorithm. The potential criteria include reputation, availability, compliance, best practices (with respect to WS-I Basic Profile), amount of documentation, etc., defined in e.g. [30].

Each user request $r \in R_{u(r)}$ must be mapped to a concrete service instance $i(r) \in I_{f(r)}$, satisfying the QoS requirements ($RT_{u(r)i(r)} \leq RT_r$, $Rel_{u(r)i(r)} \geq Rel_r$, $TR_{u(r)i(r)} \geq TR_r$). Processing capacity limits must be satisfied: for each $i \in I$ it holds that PC_i is not less than the number of requests mapped to i. Our goal is to find a solution such that the total cost (the sum of all $Pr_{u(r)i(r)}$) is as low as possible.

In the proposed and the alternative approaches, we will separately solve the subproblems for each service class (which can be done in parallel). Thus, in the following text we assume that all observed requests belong to the same class and are enumerated $r \in \{1, 2, \ldots, n\}$, with the service instances in the corresponding class enumerated as $i \in \{1, 2, \ldots, m\}$.

4 Proposed Method

Our approach first constructs an unbalanced *transportation problem* (TP) [31] as follows. The nodes on one side of the bipartite graph represent requests, each having a *demand* equal to 1 (total demand = n). The nodes on the other side of the bipartite graph repesent service instances, each having a *supply* equal to its processing capacity PC_i (total supply = $\sum PC_i$). Matching a request to a

service instance (*shipment edge*) has a given cost. In TP, the goal is to find the shipping distribution (i.e., to choose edges) from the suppliers to the demanders which minimizes the total shipment cost while satisfying all demands and supply constraints. The problem in our case is unbalanced because not all supplies have to be used, i.e., processing capacities do not have to be exhausted.

In the *balanced* TP, for which the standard TP algorithms are used, the difference between the total demand and the total supply should be zero. To balance the problem, we add an "artificial request" with the demand equal to this difference ($\sum PC_i - n$) and connect it to all suppliers with zero cost, making the service instances able to "spend" their unused capacity on the artificial request.

The transportation problem is solved in two steps:

1. Finding an initial (heuristic) solution using Vogel Approximation Method (VAM) [31, 32].
2. Iteratively improving the solution until it is optimal, using the Transportation Simplex Method (TSM) which is a (polynomial) specialization of the simplex method designed for TP [33, 34].

This two-step approach will be called SS-TSM (*Service Selection using Transportation Simplex Method*). If we stop after the first step, we get a faster but, generally, more expensive solution in terms of the defined goal function. We will call this approach SS-VAM (*Service Selection using Vogel Approximation Method*). This tradeoff will be investigated in the experimental results. We have also experimented with other optimal solutions of TP, such as reducing it to a minimum cost flow problem, but the time performance was inferior to the proposed algorithm by an order of magnitude.

5 Alternative Approaches

The following paragraphs describe two existing, competing solutions to the problem defined in Sect. 3.

Integer Programming (IP). The given problem is usually formulated and solved using integer programming (see e.g. [18,19]). In our model, the unknowns are integers $x_{ri} \in \{0, 1\}$ – whether request r is mapped to the service instance i. The given requirements can then be expressed in linear form as follows:

- $\sum_i x_{ri} = 1$ for each r – each request is mapped to exactly one service instance,
- $\sum_r x_{ri} \leq PC_i$ for each i – processing capacity limits,
- $\sum_i x_{ri} RT_{ri} \leq RT_r$ for each r – response time requirement,
- $\sum_i x_{ri} Rel_{ri} \geq Rel_r$ for each r – reliability requirement,
- $\sum_i x_{ri} TR_{ri} \geq TR_r$ for each r – throughput requirement.

Assignment Problem (AP). Another approach is using reduction to the assignment problem (by [6] and [24]). Namely, for a service instance with processing capacity k, they create k virtual instances, each being able to handle a single request. The obtained assignment problem (weighted matching) is first

tackled by greedy algorithm, and then if greedy does not find the optimal solution, the Hungarian (Kuhn-Munkres) algorithm for the assignment problem is applied. Multiplying the number of service instances strongly degrades the efficiency of such an algorithm when processing capacities are high.

6 Experimental Results

Here we describe the basic setup used in generating the artificial test data.[1] The services were randomly divided into a set number of service classes of similar size using uniform distribution; the generated requests were divided among a set number of users in the same manner. The required (request-based) and actual (user-service) properties of response time, reliability, throughput, and price per request were generated artificially according to normal distributions. The existence of a feasible solution was ensured by relaxing the required properties of requests, for about 30% on average, with respect to the actual ones. For the same reason, the sum of services' processing capacities was set to be about 30% higher than the total number of requests to be handled.

We have used two different experiment setups with respect to the total number of users, service instances, service classes, and requests. In setup A, these numbers respectively were (50, 50, 4, 100) in the *small* experiment, (500, 500, 16, 1000) in the *medium* experiment, and (5000, 5000, 32, 10000) in the *large* experiment. In setup B, these numbers respectively were (50, 50, 4, 200) in the *small* experiment, (500, 500, 16, 2000) in the *medium* experiment, and (5000, 5000, 32, 20000) in the *large* experiment. The difference between the setups lies in the total number of requests and, consequently, the service instance capacities, whose size affects the AP algorithm, but not the proposed algorithm. Each experiment was run 50 times with different random seeds and the results were averaged.

The upper figures in Table 1 depict the average obtained total cost. As shown, the proposed SS-TSM algorithm is equally expensive in terms of the goal function as the alternatives, because all three are optimal, while the proposed suboptimal heuristic SS-VAM achieves the cost higher by $9\% - 25\%$.

The lower figures show timing results on logarithmic scale. The proposed SS-TSM always outperforms IP, by a factor of up to 8.7x. The proposed SS-TSM is slower than AP in setup A by up to 3.7x, but faster than AP in setup B (where the processing capacities are higher) by up to 8.0x. The proposed SS-VAM always outperforms both alternatives and SS-TSM, being up to 13x faster than SS-TSM, up to 70x faster than AP, and up to 84x faster than IP. This shows a tradeoff between accuracy (SS-TSM) and efficiency (SS-VAM) and the variant can be decided on by the system administrator based on the system's properties and requirements. In many cases, fast service selection is preferred regardless of the suboptimal cost since high execution time makes the algorithm infeasible. For these reasons, SS-VAM seems to be the best method overall.

[1] The details can be found in the implementation:
https://bitbucket.org/satja/rsoa-selection

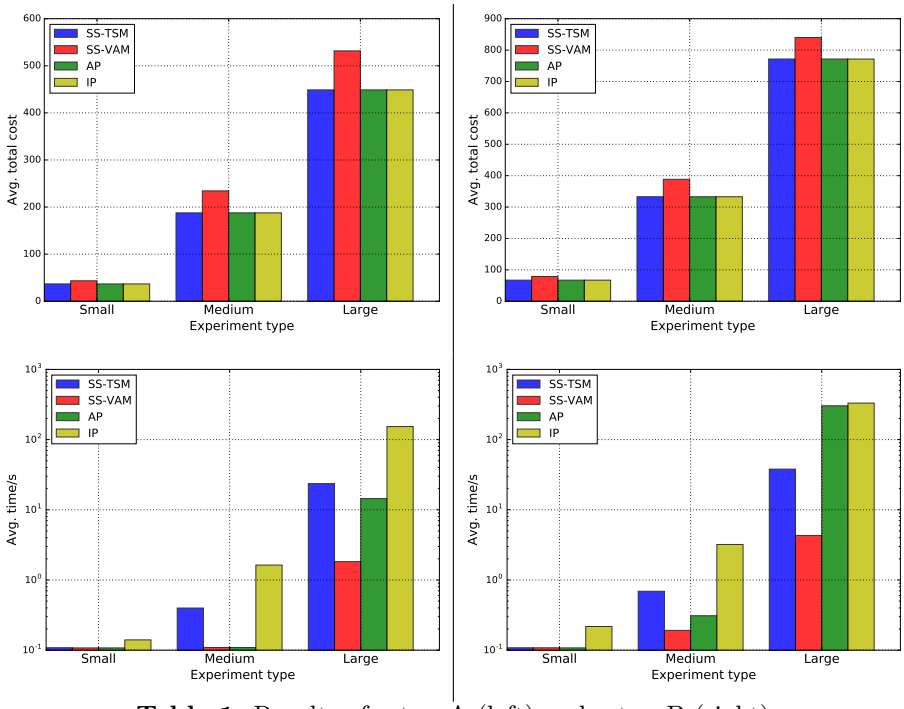

Table 1: Results of setup A (left) and setup B (right)

Although we have tried to make the test data realistic, the numerical results partly depend on the values generated artificially for testing purposes. These will certainly change in different environments. Still, the experiments make their point, which was to compare the approaches with respect to the problem size, and to show the effect of time-accuracy tradeoff.

7 Conclusion

The proposed approach for multiple-user service selection problem is shown to be more efficient than the alternatives. It relies on efficient solution to the transportation problem, where two possible variants allow the tradeoff between a high reduction in execution time (SS-VAM) and a decrease in the obtained cost (SS-TSM).

Acknowledgment

This research has been partly supported by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS). The authors acknowledge the support of the **Croatian Science Foundation** through the *Recommender System for Service-Oriented Architecture* **(IP-2014-09-9606)** research

project. The Titan X Pascal used for this research was donated by the NVIDIA Corporation.

References

1. Rao, J., Su, X. In: A Survey of Automated Web Service Composition Methods. Springer Berlin Heidelberg, Berlin, Heidelberg (2005) 43–54
2. Liu, Y., Ngu, A.H., Zeng, L.Z.: Qos computation and policing in dynamic web service selection. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters. WWW Alt. '04, New York, NY, USA, ACM (2004) 66–73
3. Cardellini, V., Casalicchio, E., Grassi, V., Presti, F.L.: Flow-based service selection forweb service composition supporting multiple qos classes. In: IEEE International Conference on Web Services (ICWS 2007). (July 2007) 743–750
4. Silic, M., Delac, G., Krka, I., Srbljic, S.: Scalable and accurate prediction of availability of atomic web services. Services Computing, IEEE Transactions on **7**(2) (April 2014) 252–264
5. Xu, Z., Martin, P., Powley, W., Zulkernine, F.: Reputation-enhanced qos-based web services discovery. In: IEEE International Conference on Web Services (ICWS 2007). (July 2007) 249–256
6. Wang, S., Hsu, C.H., Liang, Z., Sun, Q., Yang, F.: Multi-user web service selection based on multi-qos prediction. Information Systems Frontiers **16**(1) (2014) 143–152
7. He, Q., Yan, J., Jin, H., Yang, Y.: Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction. IEEE Transactions on Software Engineering **40**(2) (Feb 2014) 192–215
8. Wang, P., Du, X.: Qos-aware service selection using an incentive mechanism. IEEE Transactions on Services Computing **PP**(99) (2017) 1–1
9. h. wang, Yu, C., Wang, L., Yu, Q.: Effective bigdata-space service selection over trust and heterogeneous qos preferences. IEEE Transactions on Services Computing **PP**(99) (2015) 1–1
10. Chen, Y., Huang, J., Lin, C., Hu, J.: A partial selection methodology for efficient qos-aware service composition. IEEE Transactions on Services Computing **8**(3) (May 2015) 384–397
11. Trummer, I., Faltings, B., Binder, W.: Multi-objective quality-driven service selection – a fully polynomial time approximation scheme. IEEE Transactions on Software Engineering **40**(2) (Feb 2014) 167–191
12. Nacer, A.A., Bessai, K., Youcef, S., Godart, C.: A multi-criteria based approach for web service selection using quality of service (qos). In: 2015 IEEE International Conference on Services Computing. (June 2015) 570–577
13. Cho, J.H., Ko, H.G., Ko, I.Y.: Adaptive service selection according to the service density in multiple qos aspects. IEEE Transactions on Services Computing **9**(6) (Nov 2016) 883–894
14. Deng, S., Wu, H., Hu, D., Zhao, J.L.: Service selection for composition with qos correlations. IEEE Transactions on Services Computing **9**(2) (March 2016) 291–303
15. Hwang, S.Y., Hsu, C.C., Lee, C.H.: Service selection for web services with probabilistic qos. IEEE Transactions on Services Computing **8**(3) (May 2015) 467–480
16. Deng, S., Huang, L., Hu, D., Zhao, J.L., Wu, Z.: Mobility-enabled service selection for composite services. IEEE Transactions on Services Computing **9**(3) (May 2016) 394–407

17. Yu, T., Lin, K.J.: Service selection algorithms for web services with end-to-end qos constraints. In: Proceedings. IEEE International Conference on e-Commerce Technology, 2004. CEC 2004. (July 2004) 129–136
18. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Transactions on Software Engineering **30**(5) (May 2004) 311–327
19. Ardagna, D., Pernici, B. In: Global and Local QoS Guarantee in Web Service Selection. Springer Berlin Heidelberg, Berlin, Heidelberg (2006) 32–46
20. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient qos-aware service composition. In: Proceedings of the 18th International Conference on World Wide Web. WWW '09, New York, NY, USA, ACM (2009) 881–890
21. Kang, G., Liu, J., Tang, M., Liu, X., Fletcher, K.K.: Web service selection for resolving conflicting service requests. In: 2011 IEEE International Conference on Web Services. (July 2011) 387–394
22. Wang, H., Cheng, Y.: Interval number based service selection for multi-users' requirements. In: 2016 IEEE International Conference on Web Services (ICWS). (June 2016) 712–715
23. Wang, Y., He, Q., Yang, Y.: Qos-aware service recommendation for multi-tenant saas on the cloud. In: 2015 IEEE International Conference on Services Computing. (June 2015) 178–185
24. Jin, H., Zou, H., Yang, F., Lin, R., Zhao, X.: A hybrid service selection approach for multi-user requests. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems. (June 2012) 1142–1149
25. Zheng, Z., Lyu, M.R.: Ws-dream: A distributed reliability assessment mechanism for web services. In: 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN). (June 2008) 392–397
26. Ardagna, C.A., Jhawar, R., Piuri, V.: Dependability certification of services: a model-based approach. Computing (2013) 1–28
27. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Qos-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4** (2011) 140–152
28. Silic, M., Delac, G., Srbljic, S.: Prediction of atomic web services reliability for qos-aware recommendation. Services Computing, IEEE Transactions on **8**(3) (May 2015) 425–438
29. Silic, M., Delac, G., Srbljic, S.: Prediction of atomic web services reliability based on k-means clustering. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ESEC/FSE 2013, New York, NY, USA, ACM (2013) 70–80
30. Saleem, M.S., Ding, C., Liu, X., Chi, C.H.: Personalized decision-strategy based web service selection using a learning-to-rank algorithm. IEEE Transactions on Services Computing **8**(5) (Sept 2015) 727–739
31. Taha, H.A.: Operations Research: An Introduction (8th Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2006)
32. Reinfeld, N., Vogel, W.: Mathematical programming. Prentice-Hall (1958)
33. Dantzig, G.: Linear programming and extensions. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ (1963)
34. Winston, W., Goldberg, J.: Operations Research: Applications and Algorithms. Thomson Brooks/Cole (2004)