ICSD: Integrated Cloud Services Dataset

Samar SH. Haytamy[1], Hisham A. kholidy[1, 2] and Fatma A. Omara[3]

[1]Department of Computer Science, Fayoum University,
Fayoum, Egypt
ssa10@fayoum.edu.eg
[2]Distributed Analytics and Security Institute (DASI),
Mississippi State University (MSU), USA
hesham@dasi.msstate.edu
[3]Department of Computer Science, Cairo University,
Cairo, Egypt
f.omara@fci-cu.edu.eg

ABSTRACT. The service composition problem in Cloud computing is formulated as a multiple criteria decision making problem. Due to the extensive search space, Cloud service composition is addressed as an NP-hard problem. Using a proper dataset is considered one of the main challenges to evaluate the efficiency of the developed service composition algorithms. According to the work in this paper, a new dataset has been introduced, called Integrated Cloud Services Dataset (ICSD).This dataset is constructed by amalgamating the Google cluster-usage traces, and a real QoS dataset. To evaluate the efficiency of the ICSD dataset, a proof of concept has been done by implementing and evaluating an existing Cloud service compositing approach; PSO algorithm with skyline operator using ICSD dataset. According to the implementation results, it is found that the ICSD dataset achieved a high degree of optimality with low time complexity, which significantly increases the ICSD dataset accuracy in Cloud services composition environment.

Keywords: Cloud Computing, Cloud Services Composition, Non-functional Attributes, QoS Dataset, Quality of services, Service selection.

1 Introduction

Cloud computing as the next generation platform for conducting business is an internet-based computing where computing resources (e.g., CPU, network, storage, operating system, software applications, etc.) are offered over the internet on-demand and based on Pay-as-you-go [*1*]. Now a day, there are an increasing number of companies such as Amazon, IBM, Microsoft, and Google that offer many Cloud computing services for the consumers [*2*].

One of the main pillars of Cloud computing is the Service Oriented Computing (SOC) paradigm [*3*]. According to SOC paradigm, the Cloud consumer usually needs to use Cloud services as a partial solution to his requirements [*4*]. On the other hand, the appropriate Cloud services have been composed and provided as a single virtual

service to the Cloud consumers. The traditional quality based web services composition models consider the qualities at the time of composition [5-8]. However, the Cloud service composition is considered as a long-term strategy, and an economic target relative to traditional web services. Therefore, the Cloud services composition models should be different than those of traditional web services.

The pay-as-you-go feature of the Cloud computing technology enables the service providers to offer their services with different configuration according to the service level agreement (SLA) [*9*]. Therefore, Cloud consumers will face a challenge to select proper services from a huge number of the variations of the same services offered at different QoS levels.

So, the Cloud service selection models become urgent. Unfortunately, the absence of a common dataset for evaluating the accuracy and efficiency of the proposed services composition models is become a problem. The existing data services composition models use one or more different datasets (e.g. WSDREAM [*10*], QWS [*11*], real Cloud service data [*12*], Cloud Armor [*13*], TPC-W benchmark [*14*]). Unfortunately, these existing datasets lack some QoS parameters as time or cost or both although their importance for service composition problem. So, it is difficult to evaluate the quality and error rate of the service composition models. Therefore, a unified and completed dataset is urgently needed as a standard dataset. According to the work in this paper, an integrated Cloud services dataset (ICSD) has been introduced. This ICSD dataset includes non-functional attributes (i.e., Quality of services QoS) in terms of the functional attributes of CPU, memory, network, and storage.

The remainder of the paper is as follows; the service composition background is introduced in Section 2. Section 3 presents the principles of generating the proposed ICSD dataset. The accuracy evaluation of the proposed ICSD is presented in Section 4. Section 5 concludes the paper.

2 Cloud service composition

Service oriented computing (SOC) has emerged as a powerful concept for building software systems [*15*]. The SOC paradigm provides reusing and composing existing services to construct value-added service compositions able to fulfill complex tasks required by the consumers. One of the requirements of service composition is to meet the consumer quality of services (QoS) constraints and satisfy his preferences. One of the difficult problems in service composition is the selection of optimal single services combined together to provide value-added composed services [*16*].

Cloud service composition problem is considered as NP-hard optimization problem due to extensiveness of search space. The composite service can be defined as the composition of the required services classes SC and presented using Equation (1).

$$SC= [C_1, C_2, \ldots, C_j, \ldots, C_n] \quad (1)$$

Where C refers to a concrete single service class and n refers to the number of required single services classes. The single service class C_j is defined in Equation (2).

$$C_j = \{S_1, S_2, \ldots\ldots, S_q\} \quad (2)$$

Where each single service class C_j contains q(q>1) functionally equivalent services with different QoS values. If there is m QoS attributes should be considered to find the optimal single services, then the quality of SC can be determined by Equation (3), which is defined as the objective function.

$$Q(SC) = [Q(C_1), Q(C_2), \ldots, Q(C_j), \ldots, Q(C_n)] \quad (3)$$

The ultimate goal in Cloud service composition is to find the optimal composite service in which QoS attributes are maximized. The QoS values of a composite service are aggregated by the selected service candidates by mapping a vector of QoS values into a single real value, to enable sorting and ranking the candidate services. There are several approaches have been introduced to solve cloud service composition problem [4, 17-22]. Unfortunately, these approaches have been evaluated using different datasets with lacking some important parameters (e.g., cost and time).

3 generating The integrated Cloud services dataset (ICSD)

This section provides the principles of constructing the proposed ICSD dataset. On the other hands, the proposed ICSD is designed to measure the performance of the Cloud service composition frameworks by considering quality of services (i.e., nonfunctional attributes), and it is based on the Google cluster-usage traces and real QoS dataset [23, 12] (see Fig. 1). The proposed ICSD dataset is generated and constructed using different scenarios.

The Google cluster-usage traces and real QoS datasets are available in the comma separated values (CSV) format. The comma separated values data file is a structured data file that can be loaded and searched using many tools like MS excel, LibreOffice, open spread sheets. Unfortunately, it is not SQL enabled data. Therefore, by loading this data into database engine as SQL enabled data will enable SQL query and generate deep relationship between data that will be used in QoS measures. Then, the power of database engines could be used.

The proposed ICSD dataset has been constructed using four steps; Pruning QoS Performance Values, Mixing the Resource Utilization Dataset, Generating the Cost Parameter, and Constructing the Proposed ICSD Format.

3.1 Pruning QoS Performance Values

The real world Cloud QoS performance data includes five time series data (i.e., Availability, Max response time, Min response time, Avg response time and throughput) for 100 Cloud services. In each service, the QoS history is 6 month old found as 28 time slot [*12*]. Here, are some used queries to prune QoS_attributes table values.

1. Add idx column to be used as the primary key.

```
updateQOS_attributes set idx=rownum;
```

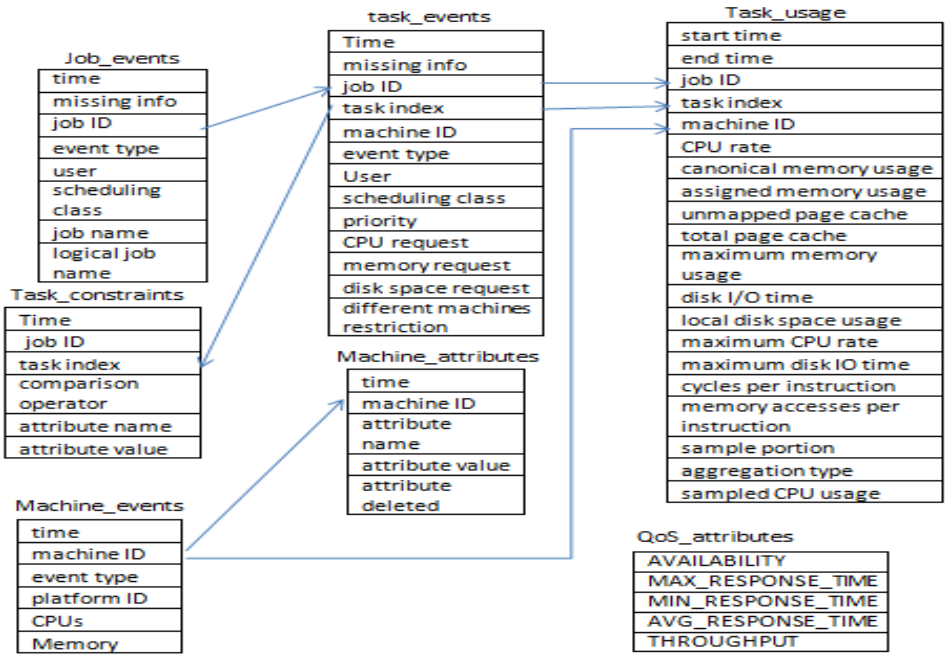

Fig. 1. ICSD raw datasets.

2. Add service_id to express about the Cloudservice id.

```
select 'update QOS_ attributes set SERVICE_ID='|| num||' where
idx>'||(rownum-1)*28||' and idx<='||rownum*28||';' from(select
to_char(rownum) num from job_events where rownum<=100 order by
to_char(rownum)DESC) ;
```

3. Discard the service providers' data that has zero availability and zero throughputs.

```
DELETE from QOS_ATTRIBUTES where SERVICE_ID in(SELECT
SERVICE_ID from QOS_attributes where availability = 0 AND
THROUGHPUT=0);
```

3.2 Generate the Integrated ICSD

One of the missed things in other Cloud QoS performance dataset is the utilization parameter of the used resources. There is a question needs to be answered "from what infrastructure resources the QoS attributes values will be so?" To answer this

question, we mix the Google cluster resource utilization data with the real world Cloud QoS performance data. Because the Google trace data has too many jobs, we randomly select only 100 jobs and make one to one mapping with 100 services of the Cloud QoS performance data. The randomness of job selection occurs only once during ICSD dataset generation. For each service in QoS performance data, 28 readings have been taken over 6 months. Therefore, 28 tasks of the mapped job are mapped to build the ICSD schema (see Fig. 2).

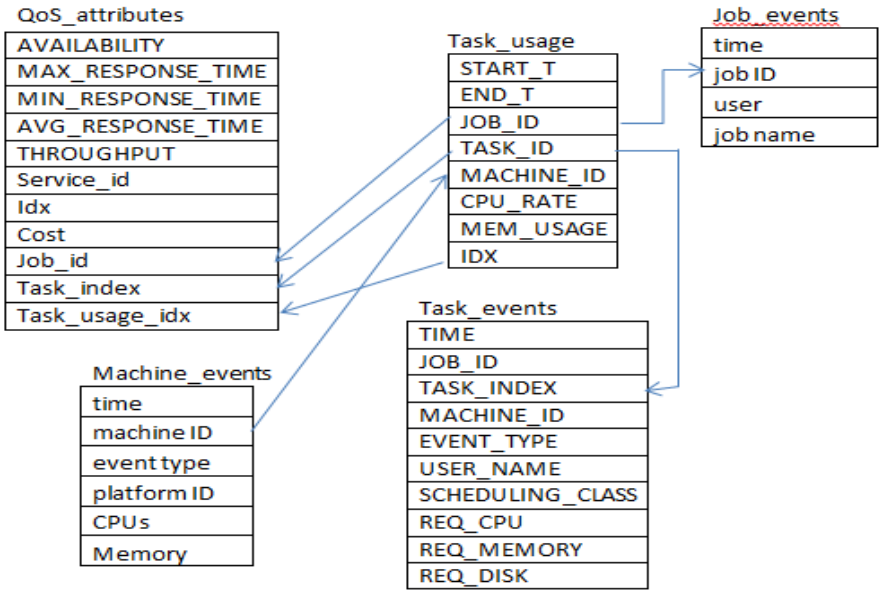

Fig. 2. ICSD Schema

The ICSD job mapping is done by the following queries:

1. First, add idx column to task_usage table to be used as the primary key to fast search operations.

```
update TASK_USAGE set idx=rownum
```

2. One-to-one Mapping of the jobs and services.

```
update QOS_attributes set job_id=  ( select e.job_id  from( select
job_id,rownum n from (select distinct  job_id from ( select job_id from
TASK_USAGE where idx is not null )order by job_id)) e  where
e.n=SERVICE_ID);
```

3. Make function take job_id as a parameter and return the idx of task_usage table.

```
create or replace FUNCTION GET_TASK_USAGE_IDX(p_job_id number)
RETURN NUMBER AS

p_result NUMBER:=0;

BEGIN

selectidx into p_result  from( select idx from TASK_USAGE where
JOB_ID=p_job_id and idx is not null order by dbms_random.value)

whererownum=1;

returnp_result;

END GET_TASK_USAGE_IDX;
```

4. Map the job's tasks to the 28 readings

```
update  QOS_attributes set TASK_INDEX=(select TASK_ID from
TASK_USAGE where
TASK_USAGE.idx=QOS_DATA.TASK_USAGE_IDX);
```

After these steps, the job mapping is done. Therefore, using database engine facilitates the search operation for getting any wanted information.

3.3 Generating Cost Parameter

It is realized that availability, response time and cost are considered important QoS parameters which affect the quality of the service [*18*]. Therefore, the cost parameter is one of the significant missing parameters in the available datasets [10-14]. The importance of cost parameter is due to its impact of building economic models for Cloud consumers and Cloud providers. So, cost values are generated and included in the proposed ICSD dataset.

In the Cloud environment, the Cloud providers do their best efforts to provide excellent services, reliable hardware resources with affordable prices, so they focus on preparing best prices to maximize their profit. In the other side, the Cloud consumers do their best efforts to contract with the best provider's offer to minimize their cost. In this paper, the cost value to service the Cloud consumers is concerned. So, the Cloud service cost can be presented as in Equation (4):

$$Total\ cost = (Cloud\ service\ price * hour\_num) + fixed\ cost \quad (4)$$

The fixed cost is distributed over the building cost, employee salary, insurance cost, and power consumption cost. The Cloud providers take into account this cost to proper estimate the service price. However, from consumer side, during the building service composition framework, this cost will be ignored because it will be paid

anyway. Therefore, the cost will determine from the Cloud service price.

To setup the economic values, we assume that the short term pricing model of the Cloud provider will not change in the long term [*24*]. The price of the resources and QoS are set up using Rackspace pricing model based on the combination of infrastructure and service level [*25*]. For general1-1 unit, the price is $0.032/hour for raw infrastructure plus $0.005/hour for management infrastructure.

According to the proposed ICSD dataset, updating cost values is done by the following queries:

1. First, knowing the instances types we work on:

```
select distinct cpu,memory from machine_events where machine_id in (
select machine_id from task_usage where idx in ( select task_usage_idx
from Qos_attributes ) );
```

The result of this query is presented in Table 1.Then; we assume that we have 6 instances types. According to the Rackspace price model, the Price can be presented as in Table 2.

2. The following query based on equation (4) is used for each instance to determine the cost.

```
update QOS_DATA set cost=(0.030+0.0004)*(select (end_t-
start_t)/(1000*60*60) from TASK_USAGE where TASK_USAGE.idx
=QOS_DATA.task_usage_idx) where QOS_DATA.task_usage_idx in (select
distinct idx from TASK_USAGE where TASK_USAGE.MACHINE_ID in
(select machine_id from MACHINE_EVENTS WHERE cpu=1 and
MACHINE_EVENTS.MEMORY=0.5));
```

3.4 ICSD Format Details

The ICSD dataset will be available in both database files and comma separated values (CSV) files. The tables of task constraint and machine attributes, and some columns in the remaining tables of Google cluster traces dataset are neglected because they are hashed and are not needed to calculate the cost (see Fig. 2).

4 Evaluating The Proposed ICSD dataset accuracy

Two methodologies have been done to prove the accuracy of ICSD dataset; using an existing service composition model, and finding the correlation relationship between ICSD parameters.

4.1 Using an existing composition model

To evaluate the accuracy of the proposed ICSD dataset, an existing service composition approach, called PSO-skyline, has been implemented using the proposed

ICSD dataset [*22*]. The main components of the PSO-skyline service composition approach is shown in Fig. 3.

Table 1. Instances Types.

CPU	MEMORY
0.25	0.2498
0.5	0.749
1	1
0.5	0.1241
0.5	0.4995
0.5	0.2493
1	0.5

Table 2. Instances with price.

CPU	MEMORY	price $/h	
		Infrastructure	Management
0.25	0.2498	0.0079	0.000125
0.5	0.749	0.026	0.00037
1	1	0.032	0.0005
0.5	0.1241	0.0112	0.000186
0.5	0.4995	0.0159	0.00025
0.5	0.2493	0.0142	0.000229
1	0.5	0.030	0004

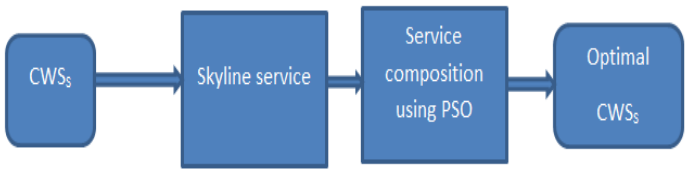

Fig. 3. The main components of the PSO-skyline service composition approach

Firstly, by applying the skyline service, the redundant services that will not be a part of the final solution is removed. So, this process will reduce the search space, and as a result, the total execution time of the service composition model will be reduced. The skyline operator is used according to the following definitions [*26*]. When a set of points in a k-dimensional space is given, the points that are not dominated by any other points are selected by using the Skyline operator.

Definition 1: Dominance

A point $\bar{p}(p_1, \cdots, p_k)$ dominates another point $\bar{q}\ (q_1, \cdots, q_k)$, denoted as $\bar{p} \prec \bar{q}$, if $\forall i \in [1, k], pi \geq qi$, and $\exists i \in [1, k], p_i > q_i$ where$\geq$ represents better than or equal to and $>$ represents better than.

Definition 2: Skyline

The Skyline of $P\ (\bar{p}, \bar{q} \in P)$, denoted by SL_P, comprises the set of points in P that are not dominated by any other point, *i.e.*, $SLp = \{\bar{p} \in P | \neg \exists \bar{q} \in P : \bar{q}, \prec \bar{p}\}$.

According to the Cloud service composition, a Skyline service can be seen as a set of service candidates that is not dominated by others in terms of all QoS attributes, such as availability, response time, and throughput. So, the previous definition can be written as follows:

Definition 3: Service Dominance

Consider a Cloud based service class **S**, and two services $s_1, s_2 \in S$, characterized by a set of **K** of QoS attributes. Service s_1 dominates service s_2, denotes as $s_1 \prec s_2$, if $\forall i \in [1, k], s_1 i \geq s_2 i$ and $\exists i \in [1, k], s_1 i > s_2 i$.

Definition 4: Skyline Service

Skyline service of a Cloud -based service class ***S***, symbolized by ***SLs***, involves those services in S that are not dominated by any other services, *i.e.*, $SLs = \{si \in S | \neg \exists sj \in S : sj \prec si\}$.

Secondly, the service composition is modeled by using particle swarm optimization (PSO) to explore the problem search space for finding the best parameters to maximize a particular objective target by maintaining simultaneously several candidate solutions in the search space [27, 28]. Through the algorithm iterations, each candidate solution is evaluated by the optimized objective function to determine the fitness value of that solution as expressed in equation (5):

$$F = \sum_{m=1}^{r} \sum_{j=1}^{n} \sum_{i=1}^{l} X_{ji} * q_m\left(S_{ji}\right) * w_m \qquad (5)$$

There is no way for the PSO algorithm to know if any of the candidate solutions are near to or far away from a local or global maximum. Therefore, the objective function is used to evaluate the PSO candidate solutions, and upon the resultant fitness, PSO algorithm operates.

Initially, the PSO algorithm randomly selects candidate solutions (particles) within the search space, which consist all the possible solutions. Each particle maintains its position, its velocity and its best fitness value which achieved during the algorithm, which referred as the individual best fitness. Finally, the PSO algorithm maintains the best fitness value achieved among all particles in the swarm, called the global best fitness, and the candidate solution that achieved this fitness, called the global best position or global best candidate solution. The PSO algorithm consists of just three steps, which are repeated until a stopping condition is met:

1. Evaluate the fitness of each particle
2. Update individual (p_{best}) and global (p_{gbest}) best fitness values
3. Update velocity and position of each particle as follow:

$$v_{id}{}^{t+1} = w.\, v_{id}{}^{t} + c1.r1\,(p_{best}(t) - x_{id}{}^{t}) + c2 \cdot r2 \cdot (p_{gbest}(t) - x_{id}{}^{t}) \qquad (6)$$

$$x_{id}{}^{t+1} = x_{id}{}^{t} + v_{id}{}^{t+1} \qquad (7)$$

To evaluate the accuracy of the proposed ICSD dataset, a comparative study has been done by implementing the PSO-skyline composition model using the proposed ICSD dataset and the existing qws dataset with the same experimental environment [*22*].

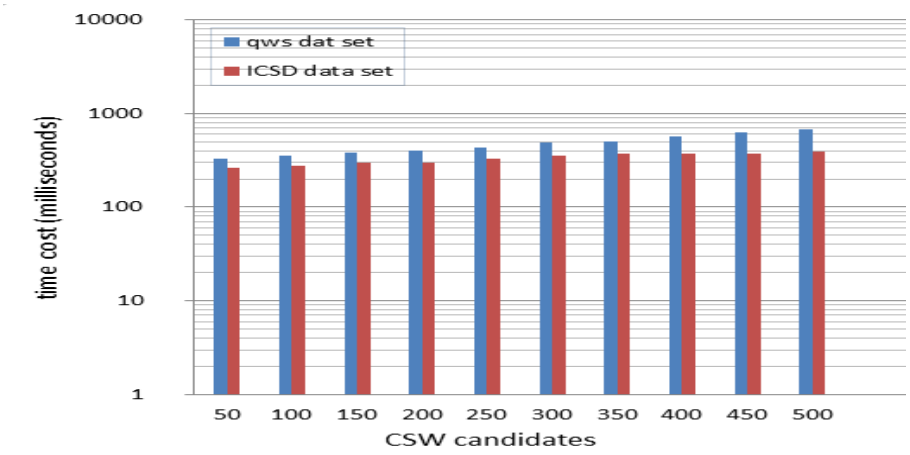

Fig. 4. Time complexity comparison of the PSO-skyline approach using qws and ICSD datasets

According to the comparative results, it is found that the time cost and optimality are nearly close (see Fig. 4 and Fig. 5). According to the results in (Figure 4), the average time of implementing PSO-skyline approach using ICSD dataset is decreased by 17% than that the time of implementing using qws dataset. The average optimal value using ICSD dataset is increased by 3% relative to qws dataset (see Fig. 5).

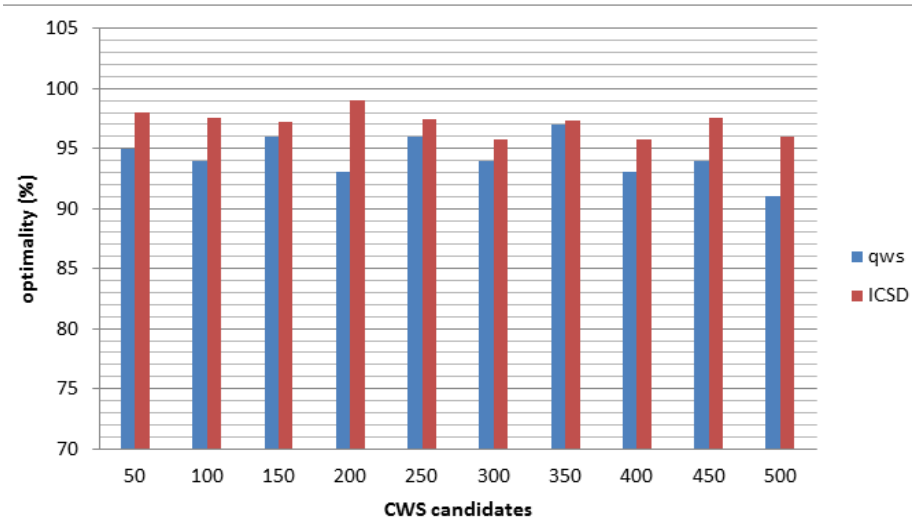

Fig. 5. Optimality comparison of the PSO-skyline approach using qws and ICSD datasets

4.2 The correlation relationship between QoS attributes

In ICSD, there exists the historical behavior of the Cloud services, which is represented as time series equations. According to Cloud consumer, his target is to maximize his profit by having shorter response time, higher throughput and lower cost as possible. We can say that there is a negative correlation between the throughput and response time attributes but they have no correlation with the cost attribute as seen in Fig. 6.

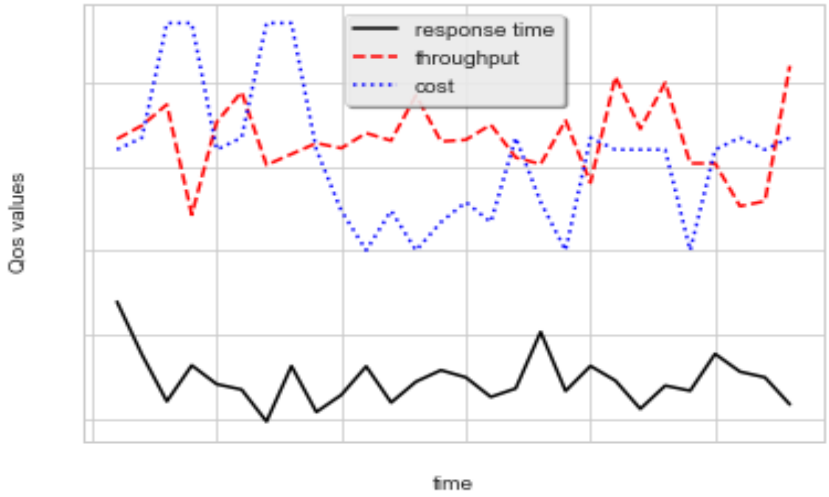

Fig. 6. QoS attributes values over time

The correlation factor between QoS attributes is common and useful in Cloud service composition environments when predicting the time series of specific QoS attributes because it is useful to predict them with considering the correlated time series ones. Equation (8) is used to determine the correlation operator [*29*] .

$$CP_{(x,y)} = \frac{1}{m-1}\sum_{i=1}^{m}\left(\frac{Q_{xi}-\overline{Q_x}}{S_{Q_x}}\right)\left(\frac{Q_{yi}-\overline{Q_y}}{S_{Q_y}}\right) \quad (8)$$

Where x and y are the time series of QoS attributes (*i.e.*, response time and throughput) and represented as follow:

$x = \{Q_{x_1}, Q_{x_2}, \ldots., Q_{x_t}, \ldots, Q_{x_{t+m}}\}$
$\overline{Q_y}$ is the mean of the series
S_{Q_x} is the standard deviation

5 conclusion

Service composition is a hot topic in service oriented computing (SOC). One of the biggest problems facing the researcher is that finding a standard and complete dataset to evaluate the accuracy of their results. According to the work in this paper, an optimal and a complete dataset, called ICSD, has been introduced, which includes the functional and non-functional parameters of the Cloud service in addition to the cost parameter. The dataset will be available[1] to be used to construct different service composition models and evaluate their accuracy using standard dataset.

References

1. Motahari-Nezhad, H. R., Stephenson, B., Singhal, S.: Outsourcing business to cloud computing services: Opportunities and challenges. IEEE Internet Computing 10(4), 1-17(2009).
2. Yu, Q., Liu, X., Bouguettaya, A., Medjahed, B.: Deploying and managing web services: issues, solutions, and directions.The VLDB Journal—The International Journal on Very Large Data Bases 17(3), 537–572 (2008).
3. Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ... & Stoica, I.: Above the clouds: A berkeley view of cloud computing. Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28(13), (2009).
4. Gutierrez-Garcia, J. O., Sim, K. M.: Self-organizing agents for service composition in cloud computing. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom),pp. 59–66. Indianapolis, USA(2010).
5. Yu, T., Zhang, Y., Lin, K. J.: Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Transactions on the Web (TWEB) 1(1), 1–26(2007).
6. Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327(2004).
7. Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., Bouguettaya, A.: Adaptive service composition based on reinforcement learning. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 92–107. Springer, Heidelberg (2010).
8. Yu, T., Lin, K.-J.: Service selection algorithms for composing complex services with multiple qos constraints. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 130–143. Springer, Heidelberg (2005)
9. Selcuk, K., Li, W. S., Phan, T., Zhou, M.:Frontiers in information and software as services. In: ICDE: 2009 IEEE 25th international conference on data engineering, Shanghai, China(2009), pp. 1761-1768.
10. Zheng, Z., Zhang, Y., Lyu, M. R.:Distributed QoS Evaluation for Real-World Web Services. In:*2010 IEEE International Conference on Web Services (ICWS)*,Miami, FL, USA(2010), pp. 83-90.
11. Al-Masri, E., Mahmoud, Q. H.: Discovering the best web service: A neural network-based solution. In:*IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, USA(2009), pp. 4250-4255.
12. Jiang, W., Lee, D., Hu, S.: Large-scale longitudinal analysis of soap-based and restful

[1]https://github.com/SamarShabanCS/Math_for_ML/tree/master/time%20series%20data%20QoS

web services. In: *2012 IEEE 19th International Conference on Web Services (ICWS)*, Honolulu, HI, USA (2012), pp. 218–225.

13. cloud armor project. [Online]. http://cs.adelaide.edu.au/~cloudarmor/ds.html, last accessed 2018/3/31.
14. TPC-W Benchmark. http://www.tpc.org/tpcw/default.asp. Last accessed 2018/3/31.
15. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges.*Computer* 40(11), 38–45(2007).
16. Liang, Q., Wu, X., Lau, H. C.: Optimizing Service Systems Based on Application-Level QoS. *IEEE Transactions on Services Computing* 2(2), 108-121(2009).
17. Sheng, Q. Z., Benatallah, B., Maamar, Z., Ngu, A. H.: Configurable Composition and Adaptive Provisioning of Web Services. *IEEE Transactions on Services Computing* 2(1), 34 - 49(2009).
18. Jula, A., Sundararajan, E., Othman, Z.:cloud Computing Service Composition: A Systematic Literature Review. *Expert Systems with Applications* 41(8), 3809-3824(2013). doi:http://dx.doi.org/10.1016/j.eswa.2013.12.017.
19. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. *IEEE Transactions on software engineering* 33(6), 369–384 (2007).
20. Liu, S., Wei, Y.,Tang, K., Qin, A.K., Yao, X.: Qos-aware long-term based service composition in cloud computing. In: *2015 IEEE Congress on Evolutionary Computation (CEC)*, Sendai, Japan (2015), pp. 3362-3369.
21. Mistry, S., Bouguettaya, A., Dong, H., Qin, A. K.: Metaheuristic optimization for long-term iaas service composition. *IEEE Transactions on Services Computing* PP(99), 1-1 (2016).
22. Wang, S., Sun, Q., Zou, H., Yang, F.: particle swarm optimization with skyline operator for fast cloud-based web services composition. *Mobile Networks and Applications* 18(1), 116–121(2013).
23. Reiss, C., Wilkes, J., Hellerstein, J. L.: Google cluster-usage traces: format + schema. Google Inc., White Paper, 1-14 (2011).
24. Goiri, Í., Guitart, J., Torres, J.: Economic model of a cloud provider operating in a federated cloud. *Information Systems Frontiers* 14(4), 827–843(2012).
25. rackspace Home Page. [Online]. https://www.rackspace.com/cloud/servers/pricing.Last accessed 2018/3/31.
26. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. *ACM Transactions on Database Systems (TODS)* 30(1), 41–82 (2005).
27. Del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., Harley, R. G.: Particle swarm optimization: basic concepts, variants and applications in power systems. *IEEE Transactions on evolutionary computation* 12(2), 171–195(2008).
28. AlRashidi, M. R., El-Hawary, M. E.: Hybrid particle swarm optimization approach for solving the discrete OPF problem considering the valve loading effects. *IEEE transactions on power systems* 22(4), 2030–2038(2007).
29. Walpole, R. E., Myers, S. L., Ye, K., Myers, R. H.: *probability & statistics for engineers & scientists*. Pearson (2007).