2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)

Multi-Tenant Cloud Service Composition using Evolutionary Optimization

Satish Kumar
School of Computer Science
University of Birmingham
Birmingham, U.K.
s.kumar.8@cs.bham.ac.uk

Rami Bahsoon
School of Computer Science
University of Birmingham
Birmingham, U.K.
r.bahsoon@cs.bham.ac.uk

Tao Chen
Department of Computing and Technology
Nottingham Trend University
Nottingham, U.K.
tao.chen@ntu.ac.uk

Ke Li
Department of Computer Science
University of Exeter
Exeter, U.K.
k.li@exeter.ac.uk

Rajkumar Buyya
School of Computing and Information System
The University of Melbourne
Melbourne, Australia
rbuyya@unimelb.edu.au

***Abstract*—In Software as a Service (SaaS) cloud marketplace, several functionally equivalent services tend to be available with different Quality of Service (QoS) values. For processing end-users multi-dimensional QoS and functional requirements, the application engineers are required to choose suitable services and optimize the service composition plans for each category of users. However, existing approaches for dynamic services composition tend to support execution plans that search for service provisions of equivalent functionalities with varying QoS or cost constraints to meet the tenants' QoS requirements or to dynamically respond to changes in QoS. These approaches tend to ignore the fact that multi-tenant execution plans need to provide variant execution plans, each offering a customized plan for a given tenant with its functionality, QoS and cost requirements. Henceforth, the dynamic selection and composition of multi-tenant service composition is a NP-hard dynamic multi-objective optimization problem. To address these challenges, we propose a novel multi-tenant middleware for dynamic service composition in the SaaS cloud. In particular, we present new encoding representation and fitness functions that model the service selection and composition as an evolutionary search. We incorporate our approach with two Multi-Objective Evolutionary Algorithms (MOEA), i.e., MOEA/D-STM and NSGA-II, to perform a comparative study. The experiment results show that the MOEA/D-STM outperforms NSGA-II in terms of quality of solutions and computation time.**

***Index Terms*—Multi-Tenant SaaS, Service Composition, Quality of Service, Evolutionary Optimization.**

I. Introduction

The growing popularity and the utilization of SaaS (Software as a Service) model has lead to the deployment of more services in cloud service market [1] [4] [7]. Gartner Inc. predicted that the SaaS model will be the largest public cloud service model and the cloud-based service market will be shifting up to 45% towards the SaaS model by 2021 [2]. The nature of elasticity, on-demand self-service and resource pooling in cloud environment brings significant benefits to SaaS service providers for building a service-oriented application by composing several existing services. In multi-tenant cloud, tenants can be differentiated based on their Service Level Agreement (SLA) which contains services information (such as service type, QoS, and cost constraints etc.). In order to satisfy tenants' SLA, application engineers are required to select suitable services from the SaaS cloud and optimize the service composition plans for each category of the tenant. In SaaS Cloud, several functionally equivalent services are available with different QoS values. The selection of candidate service from the SaaS cloud is a NP-hard multi-objective optimization problem [3] which takes a significant amount of time and cost to find the optimal service composition plans from the huge search space. This can be particularly challenging in real-time deployment scenarios, that characterized by scale, large number of multi-tenants, functionalities and varying QoS.

With the increasing interest in non-functional requirements in service composition, many research articles have focused on SLA- or QoS-aware cloud service composition problem [4] [5] [6] [7] [8]. Existing approaches for services composition tends to support execution plans that search for service provisions of equivalent functionalities but with varying QoS and cost constraints to meet the tenants' QoS requirements or to dynamically respond to changes in QoS. However, these approaches tend to ignore the fact that multi-tenant execution plans need to provide variant service execution plans, each offering a customized plan for a given tenant with its functionality, QoS and cost constraints. Furthermore, during composite service execution, run-time QoS is determined by the dynamic execution environment so that the expected QoS is not always

978-1-5386-7308-9/18/$31.00 ©2018 IEEE
DOI 10.1109/ICPADS.2018.00129

ensured in the SaaS cloud. In addition, the unpredictable tenant' requests, partner service failure or environmental changes can affect the performance or unavailability of service during the execution. Dynamic service re-composition is required for managing these issues. The goal of service re-composition is to find a new service composition plans that satisfy the multi-tenants' SLA requirements and optimize the service instances in SaaS Cloud.

To address these challenges, this paper introduces a novel Multi-Tenant Middleware for Dynamic Service Composition in the SaaS cloud. The service composition is integrated into a MOEA (Multi-Objective Evolutionary Algorithm) using novel encoding representation and fitness evaluation strategy, which explicitly considers the multi-tenant and QoS requirements. We conduct a comparative study using MOEAD-STM (Stable Matching-based Selection in Multiobjective Evolutionary Algorithm based on Decomposition) [15] and NSGA-II (Non-dominated Sorting Genetic Algorithm II) [19], two state-of-the-art MOEAs, to evaluate the effectiveness of our proposed approach. Our dynamic multi-tenant middleware goes beyond the state-of-art to envision scenarios, where variants of functionalities can be supported, offering rooms for customized services plans for tenants as requested or once they become available. This can be useful in applications with SaaS product-line offerings, where tenants may request functionally variant services customized to their need or given context. In addition, our middleware monitors and predicts the likely behavior of future requests for composite service so it can proactively re-compose execution plans. To achieve this objective, we introduced a time series forecasting based approach for predicting the future requests.

The remaining parts of this paper are organized as follows. Section II describes the multi-tenant middleware architecture. Section III introduces the motivating example. Section IV presents our service composition approach in multi-tenant SaaS environment. Section V presents experimental results and comparison with existing optimization approaches. Section VI discusses related works. Section VII concludes.

II. Multi-tenant Middleware Architecture

This section describes the proposed middleware architecture for implementing multi-tenant service composition in SaaS cloud. There are three different components in the middleware architecture namely Service Broker, Service Composition Engine and Request Predictor as shown in Figure 1. REST architecture style is used for implementing RESTful web services. A tenant can access these web service using Uniform Resource Identifier (URI) over the HTTP protocol. SaaS cloud provider publishes web services with functional and non-functional properties in the service pool.

Service Broker: In the SaaS cloud environment, tenants request for a service which is available in different packages (professional and enterprise service plan). Requests coming through HTTP protocol are processed by Request Processor. It calls the service composition engine to generate service execution plans for the tenant of each category. Furthermore, Service Broker maintains a log of each and every request. The maintained service request log is used by the request predictor for forecasting future request workload.

Request Predictor:This component analyzes the past pattern of requests generated by the tenants over the composite service and forecasts the future request workload. It continuously sends the future request workload information to the service execution planner for taking a proactive re-composition decision.

Service Composition Engine: Service composition engine manages the composition and execution of a service plan. Service plan composer receives the service broker call and identifies the type of service requested by the tenant. It searches the available web services published by the service provider and generates the set of optimal composite service execution plans according to the tenants functional and QoS requirements. The service execution planner chooses an appropriate composite service execution plan and informs the service broker to select the suitable combination of the web services from the SaaS cloud for composing a service-oriented application.

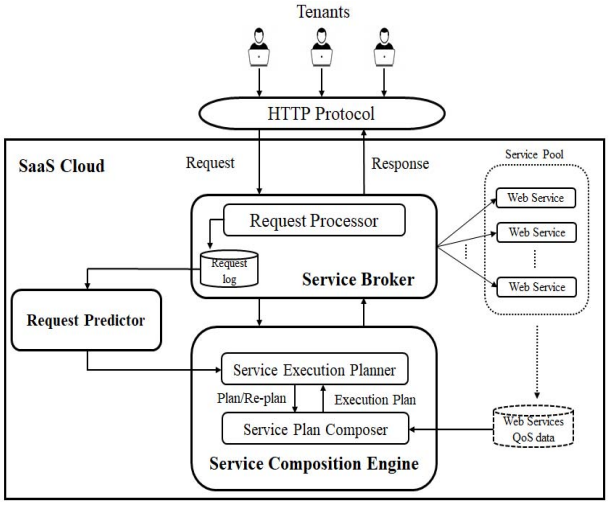

Fig. 1: Multi-Tenant Middleware for Dynamic Service Composition

In addition, service execution planner monitors the behavior of composite service execution. It reacts to changes occurring during the execution of a composite service (e.g., partner service failure and environmental changes etc.) and re-compose the service execution plan that ensures the QoS required by the tenant. Furthermore, It continuously monitors the behavior of the incoming requests generated by the tenants over the current execution plan and gets the predicted future request demand from the request predictor. If the future request demand is higher than the capacity of the current service execution plan then the service execution planner takes a proactive decision by re-planing the service execution plan for maintaining SLA violation.

III. Motivating Scenario

We take Sales CRM (Customer Relationship Management) service as our motivating scenario that illustrates the challenges of multi-objective optimization of multi-tenant service composition in SaaS cloud. Let us consider, different type of users request for the Sales CRM service which is available in different service packages. For example, Salesforce [9] provides Sales CRM service in different packages namely professional, enterprise, and unlimited. These service packages are differentiated based on the number of functionalities in the service. In multi-tenant SaaS cloud, end-users request for a different sales CRM service package based on their SLA requirements. The SaaS cloud facilitates Sales CRM service to multiple users according to their SLA requirements. Suppose, multiple-users submit their request to the SaaS cloud; in response to the requests, SaaS cloud returns Sales CRM service package as per user's SLA requirements. However, users may have multi-dimensional QoS and functional requirements, one user may request for the high throughput despite the cost of professional service while another user is interested in getting enterprise service with lower response time and cost. In these scenarios, each user has diverse requirements, a service execution plan needs to be created for the user of each category by selecting suitable concrete service from the service class. Figure 2 depicts two different application instances (application workflow) of Sales CRM service namely professional and enterprise. These application workflows contain a different number of tasks (abstract services). In the SaaS cloud environment, several functionally equivalent services are available with the different QoS (such as throughput and latency) values for attaining each task in the application workflow. Therefore, the selection of suitable candidate services from the SaaS cloud is a combinatorial optimization problem and become highly challenging when application engineer needs dynamic optimization of composing/re-composing the service execution plan for each category of user.

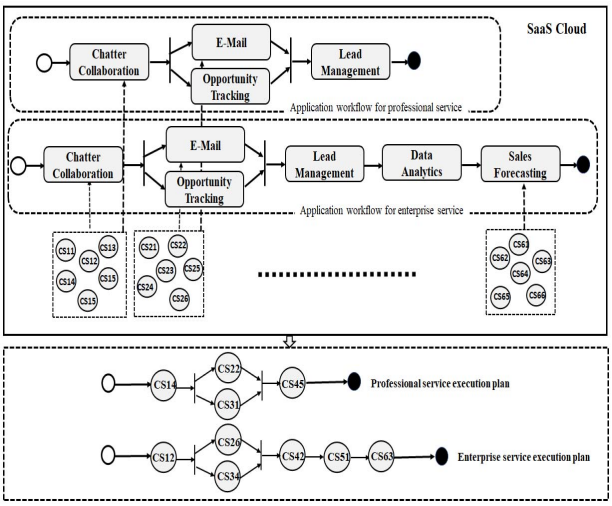

Fig. 2: Motivation Example

IV. Modelling of Multi-tenant Service Composition

This section gives basic definition of service composition in multi-tenant SaaS environment.

Defintion 1: (Service composition) In multi-tenant service composition, suppose n Composite Services (S) consist t tasks in the application workflows are represented by S_{nt} =$(S_{1t}, S_{2t}, ..S_{nt})$. Suppose there are p service classes C_i, $i = (1, 2, 3, ..p)$ and each service class contains q candidate services for attaining an abstract service in the application workflow, C_{ij}, $j = (1, 2, ..q)$. Let S_n=$(C_{1j}, C_{2j}, ..C_{tj})$ be an application workflow, where a candidate service from each service classs is selected to finish t tasks.

QoS attributes are the non-functional properties of a web service and these QoS need to consider for differentiating the service composition plan during the selection. Usually, multiple QoS attributes are considered in service composition.

Definition 2: (QoS attributes) Suppose there are l QoS attributes in a service composition, Q_r, $r = (Q_1, Q_2, ..Q_l)$ and Q_r attribute indicates the r^{th} non-functional property of the composite service.

QoS attributes can be exhibited in positive and negative criteria. Service composition process should optimize the higher value for the positive QoS attribute (e.g., throughput, and availability and reliability) and lower the values for the negative QoS criteria (response time, latency and cost etc.) [10]. These QoS attributes have numerical values in different scale of units. For example, response time is expressed in milliseconds while reliability is expressed in percentage. The opposite direction and the different scale units create the inconsistency in estimating the QoS of a composite service. To give the equal preference of all QoS attributes for computing the utility of a composite service, we calculate the normalized value of each QoS attributes in the range of (0,1). Equation (1) is used to normalize the negative QoS attributes and positive QoS attributes are normalized using equation (2) [10] [11].

$$N(Q^-) = \begin{cases} \frac{Q_i^{max} - P}{Q_i^{max} - Q_i^{min}} & if Q_i^{max} \neq Q_i^{min} \\ 1 & if Q_i^{max} = Q_i^{min} \end{cases} \quad (1)$$

$$N(Q^+) = \begin{cases} \frac{P - Q_i^{min}}{Q_i^{max} - Q_i^{min}} & if Q_i^{max} \neq Q_i^{min} \\ 1 & if Q_i^{max} = Q_i^{min} \end{cases} \quad (2)$$

Where Q_i^{max} and Q_i^{min} indicate the maximum and minimum values of the Q_i^{th} attributes of all candidate services involved in service composition and P is the current attribute value of a candidate service.

A. QoS Model for Service Composition

In this research, we consider three QoS parameters for the service composition namely throughput, response time, and cost.

-Throughput: Throughput of SaaS application is defined as the number of request the application is able to process per second.

TABLE I: QoS aggregation functions for sequence and parallel patterns

QoS Attributes	Sequence	Parallel
Cost (C)	$\Sigma_{i=1}^{t} C(s_i)$	$\Sigma_{i=1}^{t} C(s_i)$
Throughput (T)	$min_{i=1}^{t} T(s_i)$	$\min_{i=1}^{t} T(s_i)$
Response Time (RT)	$\Sigma_{i=1}^{t} RT(s_i)$	$\max_{i=1}^{t} RT(s_i)$

-Response Time: Response time of SaaS application is defined as the time required to send a request and receive the response from the service.

-Cost: The execution cost of a SaaS application is the fee that a tenant need to pay for invoking operations.

We use Sales CRM service as our testing environment of the service composition as shown in Figure 2. Table-I shows the QoS aggregation functions [5] [12] [13] [14], which are used to compute the aggregate value of each QoS attribute involved in the process of service composition.

B. MULTIOBJECTIVE OPTIMIZATION OF MULTI-TENANT SERVICE COMPOSITION USING MOEA/D-STM

MOEA/D-STM is an Evolutionary Algorithm (EA) for solving optimization problems based on the principle of decomposing a Multiobjective Optimization Problem (MOP) into a set of scalar optimization subproblems [15] [16]. MOEA/D-STM has several advantages over other EA in terms of objective scalability, computational efficiency and better performance on combinatorial optimization problems [10] [17].

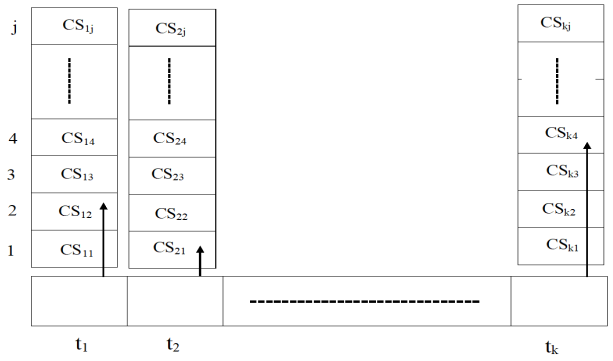

Fig. 3: Chromosome encoding

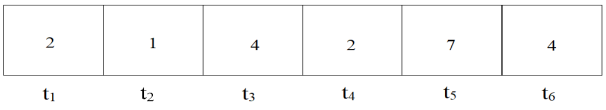

Fig. 4: Solution representation in chromosome encoding

1) Genome Encoding and Optimization Process in MOEA/D-STM: The important aspects of an evolutionary algorithms are its chromosomes and their representation because a chromosome capture all the relevant information required for a solution to the problem being considered. A genome (chromosome) is represented by a vector of t genes where t is the number of tasks in composite service (application workflow). The formulation of a genome represents the composite service solution; in which a gene encodes the concrete service for each task in composite service that could be a possible candidate solution as shown in Figure 3. In addition, Figure 4 shows the solution representation encoded by the chromosome, the value of each gene represents which concrete service (its index value) has been seleected for the corresponding abstract service such as abstract service t_1 selects the concrete service CS_{12}, abstract service t_2 selects the concrete service CS_{21} and abstract service t_3 selects the concrete service CS_{34} and so on.

We consider two type of application workflows (professional and enterprise) in our multi-tenant service composition model that are represented by a chromosome as shown in Figure 5. In optimization process, the chromosome is split into two sub-chromosomes. The first chromosome processes four genes (e.g., $t1$ to $t4$) for optimizing professional service composition plan, while the second chromosome processes all genes for optimizing enterprise service composition plan. Currently, MOEA/D-STM supports the independent execution of two application workflows and optimize them in each generation by applying genetic or problem specific operators such as selection, crossover, mutation, and reproduction.

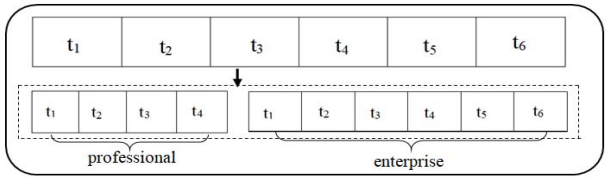

Fig. 5: Chromosome encoding for the professional and enterprise application workflow

In optimization process, MOEA/D-STM maintains the population of the individuals that represent a candidate Composite Service (CS). It uses Tchebycheff approach for decomposing MOP into N subproblems and each subproblem has one solution in current population [9]. Each subproblem is characterized by a uniform spread N weight vectors (λ). The Tchebycheff approach finds closest feasible solution CS to an ideal point (commonly known as reference point) using measure the distance between feasible solution CS and ideal point as defined follows

$$D_\lambda(cs, q) = \min_{1 \le i \le m} \{\lambda_i | q_i(cs) - q_i^*\} \quad (3)$$

Where q_i and λ_i are the QoS value and weight of the i^{th} dimension of n^{th} feasible solution CS (subproblem) respectively. For the i^{th} dimension, q_i indicates the good solution value from the set of all neighbour possible solutions of the n^{th} subproblem.

Algorithm 1 describes the process of optimizing N subproblems and corresponding individuals in the set of population. We compute the Euclidean distance between two weight vectors (subproblems) and then form a group $B(i)$ of T closest weight vectors. At each generation, randomly select

Algorithm 1: MOEA/D-STM

Initialize the population $\mu \leftarrow \{x^1, x^2, .., x^N\}$, a set of weight vectors $\lambda \leftarrow \{\lambda^1, \lambda^2, .., \lambda^N\}$, the ideal and nadir objective vectors z^*, z^{nad}.
`/* Compute neighbour group B of T closest weight vector */`
for $i \leftarrow 1 \quad to \quad N$ **do**
  $B(i) \leftarrow \{i_1, i_2, .., i_T\}$, where$\lambda_{i1}, \lambda_{i2}, .., \lambda_{iT}$ are the T closest weight vector to λ_i
end
while *Stopping criterion is not satisfied* **do**
  $p \leftarrow \emptyset$
  for $i \leftarrow 1 \quad to \quad N$ **do**
    `/* Random selection of solution set between B(i) and μ */`
    if *(rnd* $<$ *neighbourSelectionProbability)* **then**
      $S \leftarrow B(i)$
    else
      $S \leftarrow \mu$
    end
    `/* Randomly select two solution from S */`
    $x^a, x^b \leftarrow parentSelection(S)$
    `/* Reproduction operations */`
    $y \leftarrow crossoverOperation(x^a, x^b)$
    $y' \leftarrow mutationOperation(y)$
    evaluate the F-Function value of y'
    $p \leftarrow y'$
    update the current $z^* and z^n ad$ objective vectors
  end
  $e \leftarrow \mu \cup p$
  $\mu \leftarrow STM(e, \lambda, z^*, z^{nad})$
end
return μ

two solutions (parent) from the $B(i)$ or current population. The crossover operation is applied on the selected parents with the crossover probability of 0.9 for producing a new offspring, the new offspring genes are mutated then with the probability of mutation rate $1/n$ using polynomial mutation operator, where n indicates the number of genes in the individuals. Using Tchebycheff approach, we evaluate the fitness of new offspring against the individual in $B(i)$ or current population. If new offspring indicates improved solution quality then replace it with unfeasible solution in the current population. After generating the new offspring population, STM maintains the diversity in search space by allocating most preferable parent solution to each subproblem in the current population. This reproduction process is repeated with all genetic and STM operations until reaching the number of generation defined in the algorithm.

TABLE II: Algorithms Parameters

Parameter Name	Value and operators
Population size	100
Crossover Operator	SBXCrossover with crossover probability 0.9
Mutation Operator	Polynomial Mutation with mutation probability 1/n
Parent selection	Binary Tournament selection and Random selection for NSGA-II and MOEA/D-STM respectively
Maximum Generation	200
Neighborhood Size	20
Neighborhood selection, probability	0.9
Max. no.of replaced solutions	2
Function type	Tchebycheff (TCH)

V. PERFORMANCE EVALUATION AND RESULTS

We report on the implementation of the approach and its evaluation against alternative optimization techniques using real world QoS dataset [18]. We present the experimental results and evaluation of our approach using two MOEA (MOEA/D-STM and NSGA-II). The comparisons are estimated based on the solution quality and computational time.

All experiments are conducted on the same machine with Intel Core i7 2.60 GHz. Processor, 8GB RAM and Windows 10.

A. Experimental Setup and Results

We performed several experiments to evaluate the solution quality and performance of our proposed service composition approach. We used the Sales CRM service as our testing environment (shown in Figure 2) which helps in setting up the experimental parameters. We consider three dimensions of QoS attributes including positive quality properties such as throughput and negative quality properties like response time and cost. The Sales CRM service consists of 4 and 6 different tasks for professional and enterprise application workflow respectively. Randomly, we partitioned the QWS dataset into 6 groups, which shows 6 different service classes corresponding to 6 different tasks in application workflow. We randomly generate cost as an additional attribute added with each candidate service. We also compare MOEA/D-STM to NSGA-II [19] based on the performance and solution quality. We used the same parameter setting for both algorithms as shown in Table II.

Execution Time Comparisons

For the fair assessment of our evaluation study, all experiments were executed 30 times independently on the specified setup for each case and the average measure is used for the evaluation study.

1) Execution Time Vs Number of Generations: We examine the variations in execution times of both algorithms under the number of generations are increased constantly. We used the same Sales CRM testing environment and concrete services for every test case. In our testing environment, we used 6

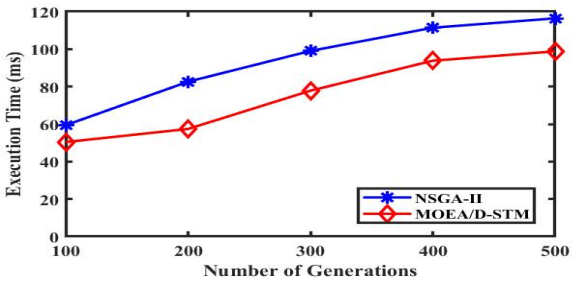

Fig. 6: Execution Time Vs Number of Generations

abstract services and 20 concrete services for attaining each abstract service in the application workflow. Figure 6 shows the execution time of MOEA/D-STM is lower than NSGA-II at each generation. Note that, MOEA/D-STM has consistently better performance at increased generation time than NSGA-II.

2) Execution Time vs. Number of Concrete Services: We analyze the variations in execution time based on the number of concrete service in each execution. For measuring execution time, we deployed 10, 20, 30, and 40 concrete services for attaining an abstract service in the same application workflow respectively. As shown in Figure 7, the number of increased concrete services couldn't make any significant effect on the execution time. However, the execution time of both algorithms increases as the number of concrete services in the execution grow. Overall, MOEA/D-STM takes lower execution time than NSGA-II at every execution.

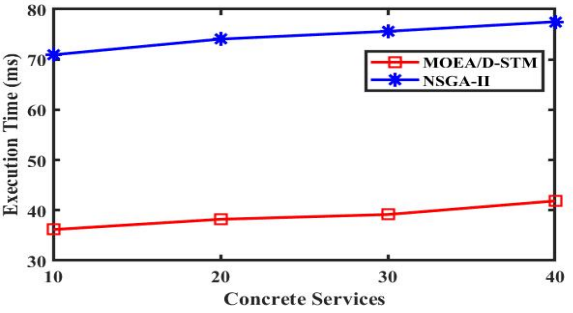

Fig. 7: Execution Time Vs Number of Concrete Services

Evaluation of Solution Quality

1) Solution comparision based on quality indicator : To asses the solution quality of MOEA/D-STM and NSGA-II algorithms, we choose two widely used performance indicators, namely Hypervolume (HV) and Inverted Generational Distance (IGD). These indicators are used to evaluate the convergence and diversity of Pareto solution set [20]. HV calculates the volume of the dominated portion of the objective space. IGD determines the convergence by computing the average distance of the obtained solutions points from the Pareto Front (PF). Algorithms with a smaller value of IGD

TABLE III: Mean and Std. Deviation of HV and IGD

Quality Indicator		MOEA/D-STM	NSGA-II
HV	Mean	9.52954	6.77102
	Standard Deviation	1.32796	1.97753
IGD	Mean	0.00352	0.00511
	Standard Deviation	1.12979 E-05	1.73 E-06

and a large value of HV are more desirable to find the better approximation to the Pareto Front. Figure 8 and Figure 9 show the standard statistic boxplot about the normalized HV and IGD values obtained from the 30 independent runs of MOEA/D-STM and NSGA-II. Furthermore, the mean and standard deviation of HV and IGD are presented in Table III. The higher mean value of HV and the lower mean value of IGD shows that the algorithm finds a good approximation to the PF. As the result, MOEA/D-STM outperforms NSGA-II as shown in Figure 8 & 9 and Table III. We have a statistically sound conclusion, from the Wilcoxon's rank sum test at the significant level of 5%, the P-value is less than 0.05 which strongly suggests that there is the difference in mean of HV and IGD; between the MOEA/D-STM and NSGA-II.

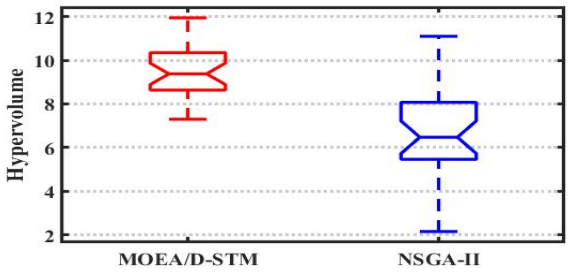

Fig. 8: Hypervolume based solution comparison

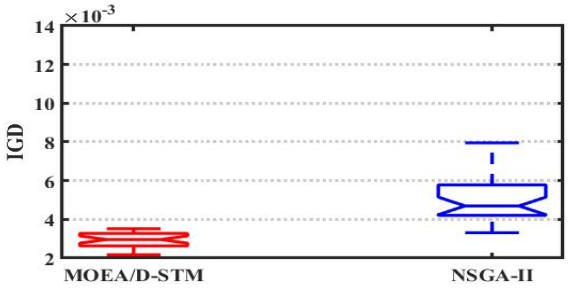

Fig. 9: IGD based solution comparison

2) Graphical representation of the solutions: We analyze the quality of solutions obtained from the MOEA/D-STM and NSGA-II. We use the same execution plans consist of 6 abstract services and 20 concrete services for each abstract service along with three QoS objectives namely throughput, response time and cost. In our service composition scenario, a good solution has high throughput with lower response time and cost. Figure 10 shows good diversity in solutions obtained from the MOEA/D-STM than NSGA-II. MOEA/D-

STM yields solution with higher throughput than the solutions obtained from the NSGA-II. Even few MOEA/D-STM solutions dominate the solutions generated by NSGA-II.

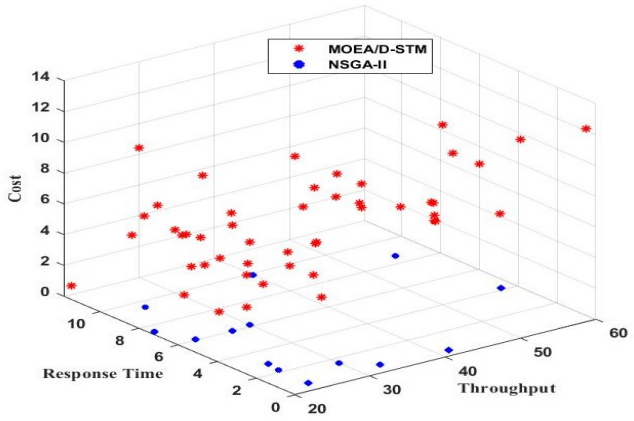

Fig. 10: Solutions obtained by MOEA/D-STM and NSGA-II

VI. Related Work

Service composition is an enabling technique for developing distributed applications by composing existing services. In past, many research efforts have been done towards QoS/SLA-aware service composition [4] [5] [6] [7] [14] [24] and researchers addressed service composition problem from different viewpoints such as QoS dependency [21], service dependency [23], and internal complementarity of services [24]. For considering the multi-dimensional QoS requirements of an end-user, the service composition problem is modeled as a multiobjective optimization problem and many optimization approaches (e.g., exact, heuristic or metaheuristic algorithms) have been proposed for producing Pareto optimal solutions [22]. Chen et.al. [14] proposed e-dominance multi-objective evolutionary algorithm (EDMOEA) for QoS-aware web service composition. The algorithm is used to find the Pareto optimal services and it facilitates the users to select the best service with the tradeoff of QoS risk and performance. Liu et.at., [6] considered long-term values in service composition model. They presented three meta-heuristic approaches named Simulated Annealing, Genetic Algorithm and Tabu Search for solving long-term based cloud service composition problems. Chen et. al. [25] presented the effects of seeding strategies for improving the overall QoS of service composition. They proposed four seeding strategies, which provides knowledge of the problem to consolidate the MEOA for optimizing service composition . Mostafa et al. [26] proposed two meta-heuristic approaches to solve the problem of QoS-aware multi-objective service composition with conflicting objective. In the first approach, they applied reinforcement learning algorithm that deals with the uncertain and dynamic environment in solving multi-objective QoS problem while in the second approach, they presented single and multiple policy-based multi-objective service compositions. This approach usages a self-organization mechanism that exploits the problem structure, to derive the weights of different QoS objectives and find a set of Pareto optimal solutions, that satisfy the multiple QoS factors in the uncertain environment. Chen et al. [21] focused on QoS-dependency in service composition. They adopted a pruning algorithm for removing unpromising candidate services from the search space and then, they applied Vector Ordinal Optimization techniques for finding Pareto optimal solutions.

In recent years, researchers proposed few approaches on the selection and composition of services in multi-tenant SaaS. To name of few, He et al. [4] proposed MSSOtimizser (Multi-tenant SaaS Optimizer) that provides effective and efficient service selection in multi-tenant SaaS environment. They considered SaaS provider's optimization goals and applied Integer Programming (IP) approach for finding optimal services that meet different users QoS requirements including SaaS optimization goals. Wada et al. [5] presented optimization framework named E^3 for solving SLA-aware cloud service composition problem. E^3–MOGA (Multi-Objective Genetic Algorithm) finds optimal solutions equally distributed in objective space and select any one of them based on end-user SLA requirements. E^3–MOGA supports three different categories of the users namely silver, gold and platinum for generating optimal service composition plans. Wang et al. [7] proposed recommendation based service selection approach for multi-tenant SaaS. They used the K-mean clustering technique for grouping the services and tenants based on similarity in terms of QoS requirements. However, these approaches support service provisions of equivalent functionalities with varying QoS. Tenants may have different functional and QoS requirements in the SaaS cloud. Consequently, these approaches are not suited for the dynamic and adaptive provision of variant execution plans, where each may offer a customized plan for a given tenant. Unlike existing approaches, each execution plan may need to support a distinct variant of functionality, QoS and cost requirements. Additionally, scaling is another limitation that needs to be addressed algorithmically. This is important for the technique to be fit for real real-time dynamic and adaptive composition and recomposition scenarios. Our middleware addresses these limitations.

VII. Threats to validity

We carried out the evaluation and simulation of our proposed approach by implementing a multi-tenant middleware for dynamic service composition. In a simulation study, the middleware supports only two different type of users request for a service (e.g., professional and enterprise service). The simulation, however can be extented to support more users; they have different functional and QoS requirements.

Though the evaluation uses the WS Dream datasets [18], as opposed to application logs, the use was beneficial as it compasses wide variation of behavior over time that may be tedious to accumulate over time and in scenarios characterized by stability and observes in a deployed setting. Middleware could be used in a real environment where web service QoS capacity can be measured based on the infrastructure

and allocated resources (e.g., CPU). In addition, we only considered two non-functional requirements (e.g., throughput, response time and cost constraints) submitted by the user for a service. However, the QoS model can be extended to include more non-functional requirements (e.g., availability, latency, and service reputation).

VIII. Conclusions

Existing approaches for dynamic services composition tend to be limited when addressing multitenant execution plans. This is due to the fact that they are not fundamentally designed for the dynamic and adaptive provision of variant execution plans, each offering a customized plan for a given tenant with its functionality, QoS and cost requirements. Additionally, these approaches can fail to scale with the number of tenants, their varying functionalities, QoS and cost, rendering them unfit for real-time dynamic and adaptive composition and re-composition scenarios. The problem is acknowledged to be an NP-hard problem. To address this problem, we proposed multi-tenant middleware that goes beyond the state-of-art to envision scenarios, where variants of functionalities can be supported. We model service composition as an evolutionary optimization problem with a novel encoding representation and fitness evaluation strategy. The middleware has the capability to react proactively, if changes would occur during the execution of a composite service (e.g., partner service failure, uncertain request workload, QoS fluctuation, and environmental changes etc.) and re-compose the service execution plan that ensures the QoS required by the tenant. For evaluating the effectiveness of our approaches, we conducted several experiments and our results showed that the MOEA/D-STM outperforms NSGA-II in terms of performance and quality of solutions.

References

[1] A. Jula, E. Sundararajan, and Z. Othman," Cloud computing service composition: A systematic literature review", Expert Systems with Applications, vol. 41, no. 8, pp. 3809 – 3824, 2014.

[2] Gartner Inc. "Gartner Forecasts Worldwide Public Cloud Revenue to Grow 21.4 Percent in 2018", https://www.gartner.com/newsroom/id/3871416, [Online; accessed 20-July-2018], 2018.

[3] M. Cremene, M. Suciu, D. Pallez, and D.DDumitrescu, "Comparative analysis of multi-objective evolutionary algorithms for QoS-aware web service composition", Applied Soft Computing, Elsevier, pp. 124-139, 2016.

[4] Q. He, J. Han, Y. Yang, J. Grundy, and H. Jin, "QoS-driven service selection for multi-tenant SaaS", IEEE International Conference on Cloud Computing,pp. 566–573, 2012.

[5] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E3: A multiobjective optimization framework for SLA-aware service composition", IEEE Transactions on Services Computing, vol.5, no. 3, pp. 358-372, 2012.

[6] S. Liu, Y. Wei, K. Tang, A. K. Qin, and X. Yao," QoS-aware longterm based service composition in cloud computing", IEEE Congress on Evolutionary Computation, pp. 3362-3369, 2015.

[7] Y. Wang, Q. He, and Y. Yang, ""Qos-aware service recommendation for multi-tenant saas on the cloud", IEEE International Conference on Services Computing, pp. 178-185, 2015.

[8] Z. Ye, X. Zhou, and A. Bouguettaya, "Genetic Algorithm Based QoS-Aware Service Composition in Cloud Computing", International Conference on Database Systems for Advanced Applications, pp. 321-334, 2010.

[9] Salesforce. "Sales Cloud Edition Report". https://www.salesforce.com/products/sales-cloud/pricing/. [Online; accessed 20-July-2017], 2017.

[10] C. Jatosh, G.R. Gangadharan, U. Fiore, R. Buyya, "QoS-aware Bigh service composition using MapReduce based evolutionary algorithm with guided mutation", Future Geneation Computer System, Elsevier, pp. 1008-1018, 2018.

[11] L. Zeng, B. benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Service Composition", IEEE Transactions on Software Engineering, vol. 30,pp. 311-327, 2004.

[12] A. Ramrez, J. A. Parejo, J. R. Romero, S. Segura, and A. R. Corts," Evolutionary composition of QoS-aware web services: a many-objective perspective", Expert Systems with Applications, Elsevier, vol. 72, pp. 357–370, 2017.

[13] A. E. Yilmaz and P. Karagoz, "Improved genetic algorithm based approach for QoS aware web service composition". IEEE International Conference on Web Services, pp. 463–470, 2014.

[14] F. Chen, R. Dou, M. Li, and H. Wu, "A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing", Computers and Industrial Engineering, Elsevier, vol. 99, pp. 423–431, 2016.

[15] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang," Stable matching-based selection in evolutionary multiobjective optimization" , IEEE Transactions on Evolutionary Computation, vol. 18,no. 6, pp. 909923, 2014.

[16] Q. Zhang and H. Li," MOEA/D: A multiobjective evolutionary algorithm based on decomposition", IEEE Transactions on Evolutionary Computation , vol. 11, no. 6, pp. 712–731, 2007.

[17] M. Suciu, D. Pallez, M. Cremene, and D. Dumitrescu, " Adaptive MOEA/D for QoS-based web service composition", In European Conference on Evolutionary Computation in Combinatorial Optimization, pp. 73–84, 2013.

[18] Z. Zheng, Y. Zhang, M.R. Lyu, "Investigating QoS Real-world Web Services", IEEE Transation on Service Computing, vol. 7, no. 1, pp. 32-39, 2014.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, " A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.

[20] K. Bringmann and T. friedrich, "Approximation Quality of the Hypervolume Indicator", Artificial Intelligence, Elsevier, pp. 265-290, 2013.

[21] Y. Chen, J. Huang, C. Lin, and X. Shen, "Multi-Objective Service Composition with QoS Dependencies", IEEE Transactions on Cloud Computing, 2016.

[22] C. Jatoth, G.R. Gangadharan, and R. Buyya," Computational intelligence based QoS-aware web service composition: A systematic literature review", IEEE Transactions on Services Computing, vol. 10,no. 3, pp. 475–492, 2017.

[23] Y. Feng, L. D. Ngan, and R. Kanagasabai,"Dynamic service composition with service-dependent QoS attributes", IEEE 20th International Conference on Web Services, pp. 10–17, 2013.

[24] X. Liang, A. K. Qin, K. Tang, and K. C. Tan," QoS-aware Web Service Composition with Internal Complementarity", IEEE Transactions on Services Computing, 2016.

[25] T.Chen, M.Li, and X.Yao, "On the Effects of Seeding Strategies: A Case for Search-based Multi-Objective Service Composition",The Genetic and Evolutionary Computation Conference, pp. 1419-1426, 2018.

[26] A. Mostafa and M. Zhang, "Multi-objective service composition in uncertain environments", IEEE Transactions on Services Computing, 2015.