CHAPTER 2

2 REVIEW OF LITERATURE

2.1 OVERVIEW

Service compositions build new services by orchestrating a set of existing services. In the Internet of Services there may be many functional similar services, but with different Quality of Service (QoS). Thus, a significant research problem in service compositions is how to select the composition's composite services that the overall QoS of the composition is being maximized. This chapter summarizes, classifies and evaluates significant research efforts in this area and gives an overview of further open research questions.

2.2 PROLOGUE

The vision of the future internet is to transform the web of information to a web of services. The goal is to develop a so-called Internet of Services (IoS), where software modules of different complexity, analogous to the paradigm software-as-a-service, are provided on network servers and consumed via the internet.

In most cases, a single service is not sufficient to fulfill customers' complex requirements. In that case, service compositions are needed. A service composition orchestrates a set of services to one to solve a complex goal successfully [21]. The most well-known example of service composition is the travel planer that invokes a flight, hotel and car booking service in sequence.

Building a composition requires two steps [22]. At first, the functionalities required by the composite services (namely the tasks of the composition) and their interactions, the control and data flow, are identified.

Secondly, for each task, an appropriate implementation is selected and bound to the task. We call this set of service implementations execution plan of the composition.

As soon as there are at least two service implementations for one task, a selection of the numerous execution plans for the service composition has to make. In most cases, the execution plan selection is not done arbitrarily, but to maximize and/or limit some individual properties of the composition. One of the most important properties is the Quality of service (QoS). QoS is a measure for how well a service serves the customer.

Classical QoS properties are response time, availability or reputation. The selection of the optimal execution plan that maximizes the composition's end-to-end QoS is a well-known NP-hard problem [23], called QoS-aware service composition. This chapter summarizes, classifies and evaluates significant research efforts on QoS-aware service composition and gives an overview of current trends and still open research questions.

2.3 NOMENCLATURE OF THE QOS METRICS

2.3.1 Web Service Quality Model

The web services providing services of the same type may categorize as a community where the selection of a web service from a community may be done based on their nonfunctional properties such as differences of pricing, etc. The non-functional properties known as Quality of Service (QoS) characteristics are the base for differentiating and selecting a web service. A quality model dependent on quality criteria set for Web service can be applied where the criteria is the non-functional properties such as pricing, reliability, etc. of the WS. Any criteria additionally may incorporate into the model over the existing criteria of selection without any modifications.

2.3.2 The Criteria of Quality of Services:

2.3.2.1 Execution Price:

For a specific service (s), the fee paid by a requestor of a service for executing of an operation (op) is the execution price $q_{pr}(s, op)$ **.**

2.3.2.2 Execution Duration:

The execution duration $q_{dr}(p)$ is computed is based on the Critical Path Algorithm (CPA) applied similarly to a project digraph on the execution path W_e of an execution plan (p). A project digraph critical path is a path from the initial state to the final state having the nodes labeled with the longest total sum of weights. A node relating to a task t in W_e, and its weight is the service operations execution time executed by t, which is, $q_{du}(sv_p(t), op(t))$, where $sv_p(t)$ represents the service allotted to task t in plan p, and $op(t)$ represents the operation performed by task t. A critical path related task is known as an essential task, whereas a critical path related task to which a service is allotted is known as a critical service.

The Figure 2.1 offers an instance of a critical path representing an execution path depicted as a project digraph, and the related execution plan p is depicted as, ***p = { <t1;*** *s*1 ***>;<t2; s2 >;<t3; s3 >;<t4; s4 >;<t5; s5> }.***

The execution duration is shown next to each service in the Figure 2.1. The project digraph has two project paths, wherein project path 1 is < ***t1; t4; t5*** > and project path 2 is < ***t2; t3; t4; t5*** >. Project path 1 shows execution time of 37 seconds, and project path 2 has execution time of 62 seconds. Hence path 2 is the critical path where the plan has an execution time of 62 seconds. The critical tasks are tasks, t2, t3, t4 and t5 whereas the critical services are the services s2, s3, s4, and s5.

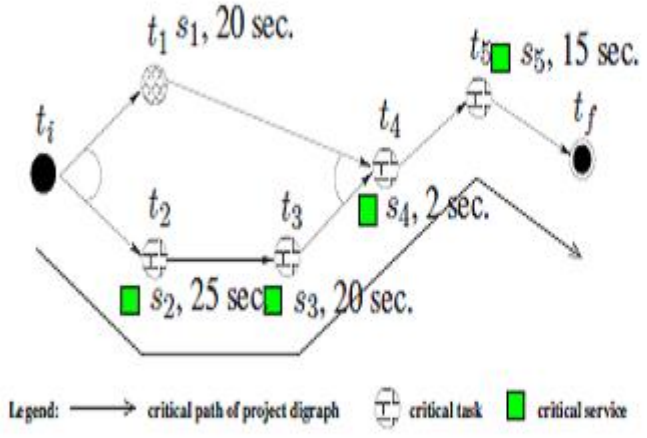

Figure 2.1: Example of Critical Path

2.3.2.3 Reputation:

The reputation $q_{rep}(p)$ for an execution plan **(*p*)** is computed as the average of the services reputations involved in ***p***.

$$q_{rep}(s) = (\sum_{i=1}^{N} q_{rep}(S_i)) / N \qquad \text{(Eq 2.1)}$$

2.3.2.4 Successful Execution Rate (Reliability):

$q_{rat}(s)$ is the probability of appropriately responding to a request within a frame of maximum expected time specifying a depiction of a web service.

$$q_{rat}(s) = N_c(s) / N \qquad \text{(Eq 2.2)}$$

$N_c(s)$ is the no. of times of finishing the service with success within a frame of maximum expected time where ***k*** is the total number of executions.

2.3.2.5 Availability:

$q_{av}(s)$ is the probability of the accessibility of the service.

$$q_{av}(s) = T_a(s) / \theta \qquad \text{(Eq 2.3)}$$

$q_{av}(s)$ is the total measure of time (in sec) for which the service (s) in the last θ seconds, is available.

2.4 CONTEMPORARY AFFIRMATION OF THE RECENT LITERATURE

QoS-aware service composition problem is described as an NP-hard problem [24]. Several techniques have developed to tackle QoS-aware service composition problem. Some have deal with service composition problem under non-varying QoS environment, while others have dealt with the issue in varying QoS environments (runtime environments). We therefore classify current approaches according to whether not they take QoS changes into account. This classification is necessary as it is currently missing in literature. In this section, techniques are classified into static and adaptive approaches.

2.4.1 Static approaches

This category of techniques performs web service composition using prior knowledge of web service QoS values. Also, they do not consider dynamic changes in QoS. Hence, they are not ideally suitable in runtime environment where web service QoS values are constantly changing over time. We further classify techniques here into four categories namely; (1) Local maximization approaches; (2) Linear optimization approaches; (3) Approximation approaches and (4) Pareto optimization approaches.

2.4.1.1 Local maximization approaches.

One of the reasons why there is difficulty in dealing with a QoS service composition problem is the presence of both local QoS constraints (applied to each web service) and global QoS constraints (applied to the whole composite service). Ideally, the solution would have to satisfy both the requirements before being called an optimal solution, which can be difficult to achieve. A method current studies have used in reducing complexity of the problem is to consider only local constraints in selecting best candidate services for each subtask. This way, only local constraints will need to be satisfied before solution is executed. The process is known as local maximization. For example, instead of finding the composite service that has total response time less than 20ms, local maximization finds the composite service whose candidate services each have response times less than 20ms. In this case, the best solution may not necessarily have total response time less than 20ms. Using the method, an optimal solution can reach in the little amount of time. A popular local approach is Dynamic Programming [25], [26] which breaks down an execution plan into separate divisible and indivisible parts. It computes an optimal solution for each divided part while relying on a recursive branch-and-bound algorithm in finding optimal solutions for the inseparable parts. Another local approach is a technique proposed in [27]. The method uses a learning-depth-first search (LDFS) algorithm that combines bound depth-first searches with learning iteratively. Some other technology for local maximization is one based on Simple Additive Weighing (SAW) [28]. This method assigns scores to each candidate services by multiplying their QoS values with a user-defined weight value. The weight value signifies the importance of satisfying a given QoS constraint specified by the user. The method then selects the web service with the best score for each task. An advantage of local approaches is that their running time scales well with an increase in the number of services. However, they suffer from loss of accuracy because of the pre-selection process involved.

2.4.1.2 Linear optimization approaches

Linear approaches refine the NP-hard problem by assuming a linear objective function. An objective function is a measure of how good the QoS value of a composite service is. It is determined by combining QoS property values of all the candidate services contained in the composite service into a single aggregate value. Linear optimization of web service composition is achieved using Linear Integer Programming (LIP) techniques [29], [26]. LIP can find the best composite service without necessarily building all possible compositions. It can also integrate both functional and non-functional service properties [30] into the composition process. LIP is usually useful when dealing with a small set of services. Although, it can take a lot of time to find the best composition in large service environments.

2.4.1.3 Approximation approaches:

Current studies have also focused on techniques for finding approximate solutions as they are faster to reach than optimal ones. These approaches are usually heuristic in nature as they achieve optimization by performing various computations and iterations on possible solutions concerning a given degree of quality. Heuristics are capable of arriving at solutions while making little or no assumptions about the problem. They are also capable of rigorously covering huge search spaces in relatively small amount of time. Several approximation approaches have introduced. One popular method is Particle Swarm Optimization (PSO) technique. PSO simulates the natural behavior of birds when searching for food particles, each particle representing a possible solution (composite service). All particles within the swarm are assigned parameters such as position(X), velocity (V), best local position (Xbi), and best global position (Xgi). In the study [29] first investigated the application of Particle Swarm Optimization (PSO) in the context of service composition.

They introduced a discrete PSO approach called DPSO to finding approximate solutions. Their technique uses best global position (Xgi) to update location and speed parameters for all particles in the swarm after each iteration;

$$X_{i+1} = X_i + V_{i+1} \ (3) \qquad \text{(Eq 2.4)}$$

$$V_{i+1} = V_i + C1R1(Xb_i - X_i) + C2R2(Xg_i - X_i) \ (4) \qquad \text{(Eq 2.5)}$$

Where vectors Xi is a particle position, Vi represents particle velocity, Xbi represents particle's best-recorded position, Xgi is the global best particle position, C1 and C2 are called social parameters, while R1 and R2 are randomly generated numbers between 0 and 1. The technique finds good quality solutions in considerable amount of time, but suffers from been trapped into local optima more often than not. An improved PSO approach is presented in [31]. This approach incorporates adaptive mutation and learning factor ability into the basic PSO technique. Here, adaptive mutation mechanism prevents particles from being trapped in the local maxima by letting them hop out during early stages of computation, while learning factor tunes C1 and C2 values to improve population density of particles during computation.

Another popular approximation technique is one based on Genetic Algorithms (GA). GAs is capable of evolving members of a generation (genomes) according to a set of rules up to a point where fitness value is optimized. Each genome can represent a possible composite service and is usually encoded in the form of array of numbers. GA initiates the optimization process by building an initial generation of genomes which are subsequently to find the ones with best fitness values. These best individuals are then placed into a mating pool where crossover and mutation operators alter them to improve their fitness. Song et al. [32] studied how GA can use in solving the service composition problem. In their approach, they encode candidate services (WSij) in the form of an integer array inside each gene of a genome. Their technique aims to find a combination of array elements that achieve the best fitness value while meeting QoS constraints. Their approach finds good quality solutions most of the time, however, it often traps into local optima. Authors in [33] present an improved GA that reduces the likelihood of trapping into local optima. They perform two functions; an improved mutation function controls mutation process via mutation probability parameter, the other function is a partial initialization function that stores set of best genomes which will later use in performing crossover with other genomes.

Literature indicators suggest that approximation approaches are more efficient than linear approaches as they can rapidly eliminate large numbers of possible execution plans in a relatively small amount of time. Furthermore, they can handle a large number of services better than linear methods. However, they suffer from lack of ability to find optimal solutions most of the time. It is possible that the best solution was discarded during the elimination process using these approaches.

2.4.1.4 Pareto-optimization approaches

These approaches find solutions that are optimal concerning one or more objective functions, but not all. One method that yields Pareto solutions is a two-stage conversion of a multi-objective problem into a single objective problem as proposed in [34]. This involves normalizing an objective function's range, and then using Simple Additive Weighting (SAW) technique to weight each objective function. In [35] a multi-objective optimization algorithm that relies on cost of service failure to rank Pareto-optimal solutions concerning a decision maker's preferences is proposed. The algorithm, however, cannot distinguish which Pareto-optimal solution is the better one. The study in [36] suggests a genetic algorithm based on non-dominated sort (NSGA-II) that map genomes to their respective fronts to sort and compare Pareto-optimal solutions. In [37] predicts a different comparison technique based on binary quality indicators that analyze Pareto-optimal solution sets. Pareto-optimization approaches have observe to provide better performance and good quality solutions than other methods.

2.4.2 Adaptive approaches

In the previous section, techniques covered that focus on finding optimal solutions to the NP-Hard problem in static environments, i.e., environments where QoS values of web services are already known prior to generating the task workflow. Some current studies have extended the service composition problem to finding optimal solutions in situations where web service QoS values are not known prior to generating the workflow. Such examples reflect a more realistic scenario where runtime web service QoS values vary from values advertised by service providers. Adaptive approaches can adapt to changes in QoS values of the service environment. Adaptive service composition is a very active area of research attracting much of attention in recent years. It is divided into two types [38]; internal composition adaptation and External composition adaptation techniques. The former approach reacts to environmental changes by rebuilding a composition either from ground up or from the point of fault within the composite service. On the other hand, the latter approach uses adjustable adapters that bridge the gap between the service workflow and the dynamically changing service environment.

There are several internally adaptable service composition approaches, most of which have focused on small service environments. Amongst these approaches are AI Planning based techniques [39], [38]. In [39] the authors present an AI planning approach that relies on a moderately accurate Case-Based Reasoning (CBR) technique to build service compositions on-the-fly. CBR is used to obtain solutions from a set of composition cases gathered from experiences. If such solutions do not exist, then an AI planner builds composition solutions from ground up. The other study [38] presents a self-adaptive service composition that is based on AI planning graphs called Graph plan repair. Their approach reconfigures compositions during runtime with the aid of a greedy search algorithm used to explore the planning graph for possible service combinations that can achieve the user's goal. The algorithm scouts through the planning graph to find and repair services that don't meet user goal due to their unexpected change. If such services do exist, then new services are added to the graph to satisfy user goals.

Afterwards, the whole process starts all over until all services satisfy user goals. The method presented in [40] uses AI planning to map user requirements to service workflows dynamically. The approach is goal-oriented, thus any changes to user requirements at runtime will ultimately apply to the workflow structure. The major drawback of AI planning-based techniques is that they do not perform well in situations where the number of participating web services is large. Also, they are susceptible to sub-optimal solutions in such environments.

Several internal adaptation solutions focus on using Reinforced learning (RL) techniques to solve adaptive service composition problem. Work in [41] proposes an adaptive RL method based on Markov Decision Process (MDP) that finds optimal solution at runtime. MDP builds a model for obtaining compositions consisting of multiple aggregated workflows. RL method takes over in finding optimal solution (or Pareto-optimal solutions) by acquiring MDP policy with the optimal QoS. Any change in service environment will prompt a change of MDP policy for the sake of continuing the learning process. In [42] an extended MDP method called Semi Markov Process (SMP) has introduce to forecast QoS and network efficiency of web service environment during composition. The output from SMP then determines which web services need to replace because of poor QoS. In [43] the authors propose a method that re-plans composition once it predicts a difference between estimated

QoS and runtime QoS values. In their approach, the authors utilize a proxy-based model to achieve runtime binding of web services. In [44], it proposes an improved RL approach that utilizes Reuse Strategy to enhance performance and stability of RL. Work has extended in [45] which introduce a randomized RL technique that considers multiple QoS and non-QoS criteria like cost, reputation, deadline and user preferences to obtain optimal solutions while adjusting to runtime changes in availability of service environment. RL-based approaches enhance the service composition process with exploration and exploitation capabilities, making the composite services more reactive to environmental changes.

However, they impose high computational cost, especially in large service environments. Additional studies [46] have modeled the adaptive service composition problem as the shortest path problem. Here, the shortest path algorithm (CSP) is used to ascertain a faulty web service and come up with an alternative path to a backup web service. This approach often traps into local optima and doesn't always produce the best results. In [47], this paper focused on using heuristic approach to tackling adaptive service composition. Their method makes use of a K-means algorithm to find Pareto-optimal solutions. The algorithm works by firstly normalizing QoS constraints then a local classification is made to group candidate services into clusters concerning their QoS levels, upon which a heuristic algorithm performs global optimal selection.

Adaptation is performed by using a utility threshold τ to tune selection of clusters based on environmental constraints such as time or service density. With the aid of its classification technique, the approach performs efficiently in environment where the number of web services is large.

Some external composition adaptation techniques have proposed in recent studies. These techniques use composition methods, policies or protocols to continually regenerate and update optimal service workflows according to changes in environment. Social network analysis techniques have recently introduced to tackle adaptation in service composition. They are methods used to map and measure the relationship between web services in a social network.

An example is proposed in [18] which applied link analysis and page ranking to rank services based on their success and failure popularity. To obtain such information, snapshots of the whole service registry are taken at regular intervals so that popularity changes can reflect upon service workflows accessible to the user at runtime. A modified page rank approach namely service rank is presented in [47] which applied link analysis and page ranking to rank services based on their success and failure popularity. In this approach, web services were ranked based on connectivity and invocation history. The method combines ranking score with QoS score for composition ranking. Social network analysis techniques are more efficient than internal adaptation approaches since the information required for the adaptation process is obtained prior to the commencement of service composition. Its only drawback is that it needs more computational resources to acquire information that will be useful for adapting composite services.

The contemporary research also focused on service composition based on meta-heuristic approaches which offer QoS awareness in the composition. The strategy for offering composite services which include features of QoS, based on the greedy approach is given by Yu et al. in the paper [48]. In this approach, the search time for a solution is minimized, and with an adaptive strategy, the performance efficiency is achieved. In the paper [49], a web service composition technique WSMO (web service modeling ontology) by Xiangbing et al., proposes to achieve, in the least search time, performance effectiveness for determining a solution that is of optimal value. An approach [50] for QoS-aware web service composition by Zhao et al. utilizes the immune optimization algorithmic for enhancing the strategy for finding the local best value and an algorithm based on PSO is used to determine the global optimized value, to minimize the search time and maximize scalability. In the paper, [51] for achieving service quality optimization in the runtime composition of QoS-aware web services, Parejo et al. design a technique GRASP, and an algorithm for path re-linking. These techniques have benefits in terms of the optimal results offered, however the disadvantage is that each technique considers only a single QoS metric or maximum two metrics based on which it is not possible to model the scalability of the service composition. Another disadvantage is that the services involved in a composition proliferate with the increase of the tasks and the associated services, where O(n2) is the complexity of computing the service composition. The research on the existing and inherent complications of composing a web service focused on methods such as linear programming, integer programming, genetic algorithm programming, or dynamic programming. A linear integer programming (IP) method for overcoming the problems of QoS aware composition is proposed in the paper [22]. The service composition is implemented based on the global planning strategy and the objective function as well as the constraints as input is in a linear fashion in the linear integer programming. This technique effectively limits the execution plans possible due to the associated vast number of tasks and related web services. The disadvantage of the method is that it uses the functions of linear aggregation for various QoS attributes and in case of non-linear integer programming is used, difficulties in scalability are experienced. The Genetic algorithm (GA) is effective in addressing the complications in the optimization process of combination services [52]. Also, the impact on a genetic algorithm performance [53] varies according to the different parameters. A Genetic Algorithm (GA) based technique for web service composition that is QoS aware is proposed in the paper [54]. The authors compared the GA based technique with the linear IP technique and state that in case of workflows of limited size, the linear IP method is more effective. AGA based QoS-aware approach for services selection [55] studies the impact on the genetic algorithm performance of optimization due to various parameters, such as mutation rate, fitness function, etc. A comparison of the GA technique with the branch-and-bound techniques and exhaustive search shows the adverse impact on the efficiency of the genetic algorithm in identifying close to optimal results because of parameters such as, setup values, the goals specified for optimization, and due to constrains involved. The model devised in [56] defined set of QoS factors to predict available services. Many of existing quality-aware service selection strategies aimed to select the best service among multiple services available. The model devised in [22] considering the linear programming to find the linear combination of availability, successful execution rate, response time, execution cost and reputation, which is regarding find the optimal service composition towards given business operation. The model devised in [57] is considering the temporal validity of the service factors. The authors in [58] modeled a mixed integer linear program that considers both local and global constraints. The model devised in [59] is selecting services as a complex multi-choice multi-dimension rucksack problem that tends to define different quality levels to the services, which further considered towards service selection. All these solutions are depended strongly on the positive scores given by users to each parameter. However, it is not scalable to establish them in prospective order. Though the QoS strategies defined are used in service composition the factor fault-proneness of the service composition is usual.

Regarding this, a model devised in [60] explored a mechanism for fault proliferation and resurgence in dynamically connected service compositions. Dynamically coupled architecture outcomes in further complexness in need of fault proliferation between service groups of a composition accomplished by not depending on other service groups. In a gist, it can conclude that almost all the benchmarking service quality assessment models are attributed specific, user rating specific or both. Hence the importance of attributes is divergent from one composition requirement to other, and contextual factors influence the user ratings, and another important factor is all these benchmark models are assessing services based on their performance, but in practice, the functionality of one service may influence by the performance of another service. Henceforth here in this chapter, we devised a statistical approach that estimates the impact scale of service composition towards fault proneness, which is based on a devised metric called composition support of service compositions and service descriptors.