# QoS Optimization for Cloud Service Composition Based on Economic Model

Hisham A. Kholidy[1,5], Hala Hassan[2(✉)], Amany M. Sarhan[3], Abdelkarim Erradi[1], and Sherif Abdelwahed[4]

[1] Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha, Qatar
{hkholidy, erradi}@qu.edu.qa
[2] Department of Computer Engineering and Systems, Faculty of Engineering, Mansoura University, Mansoura, Egypt
Hala_h62@yahoo.com
[3] Faculty of Engineering, University of Tanta, Tanta, Egypt
amany_m_sarhan@tanta.edu.eg
[4] Electrical and Computer Engineering, Mississippi State University, Starkville, MS, USA
sherif@ece.msstate.edu
[5] Faculty of Computers and Information, Fayoum University, Fayoum, Egypt

**Abstract.** Cloud service composition is usually long term based and economically driven. Services in cloud computing can be categorized into two groups: Application services and Computing Services. Compositions in the application level are similar to the Web service compositions in Service-Oriented Computing. Compositions in the computing level are similar to the task matching and scheduling in grid computing. We consider cloud service composition from end users perspective. We propose Genetic Algorithm-based approach to model the cloud service composition problem. A comparison is given between the proposed composition approach and other existing algorithms such as Integer Linear Programming. The experiment results proved the efficiency of the proposed approach.

**Keywords:** Cloud service composition · Cloud computing · Genetic algorithm · Quality of service

## 1 Introduction

Cloud computing is emerging as the new paradigm for the next-generation distributed computing. Big companies such as Amazon, Microsoft, Google and IBM are already offering cloud computing solutions in the market. A fast increasing number of organizations are already outsourcing their business tasks to the cloud, instead of deploying their own local infrastructures [1]. A significant advantage of cloud computing is its economic benefits for end users and service providers.

Services in cloud computing can be categorized into application services and computing services [2]. Almost all the software/applications that are available through the Internet are application services, e.g., flight booking services. Computing services


R. Giaffreda et al. (Eds.): IoT360 2014, Part I, LNICST 150, pp. 355–366, 2015.
DOI: 10.1007/978-3-319-19656-5_48

are software or virtualized hardware that supports application services, e.g., virtual machines, CPU services and storage services. Service compositions in cloud computing include compositions of application services and computing services. Compositions in the application level are similar to the Web service compositions in SOC. Compositions in the computing level are similar to the task matching and scheduling in grid computing. Cloud service composition is usually long-term based and economically driven. Traditional QoS (Quality of Service)-based composition techniques usually consider the qualities at the time of the composition [3]. This is fundamentally different in cloud environments where the cloud service composition should last for a long period.

In this paper, a genetic-algorithm-based cloud service composition approach is proposed. We focus on the selection of composition plans based solely on non-functional or QoS attributes. The comparisons between the proposed approach and other existing ones show the effectiveness and efficiency of the proposed approach. The rest of the paper is structured as follows: Sect. 2 illustrates the preliminaries of service composition in cloud computing. Section 3 provides an overview of cloud service composition problem. Section 4 highlights the related work of the cloud service composition models. Section 5 describes in details the proposed composition approach. Section 6 tests and evaluates the proposed approach and presents the experiment results. Section 7 draws a conclusion and highlights the future work.

## 2 Preliminaries

In this section we present basic knowledge about cloud computing, service compositions in cloud computing and QoS model.

### 2.1 Cloud Computing System

Cloud computing provides two types of services, application and computing services. Application services are the most visible services to the end users (i.e. Google Apps), Cloud systems provide these services to the end users through the software providers in SaaS (Software as a Service) layer. Computing services are the hardware and system software in the datacenters that provide those services [2] see Fig. 1.

Some vendors use terms such as PaaS (Platforms a Service) or IaaS (Infrastructure as a Service) to describe their products. In this paper, PaaS and IaaS are considered together as Computing Services. PaaS are platforms that are used to develop, test, deploy and monitor application services. For example, Google has Google App Engine that works as the platform to develop, deploy and maintain Google Apps. IaaS services provide fundamental computing resources, which can be used to construct new platform services or application services. Computing Services include computation services, i.e., Virtual Machines (VMs); storage services, i.e., Databases; and network services. Computing Services Vendors are these companies or organizations that make their computing resources available to the public such as Amazon EC2.

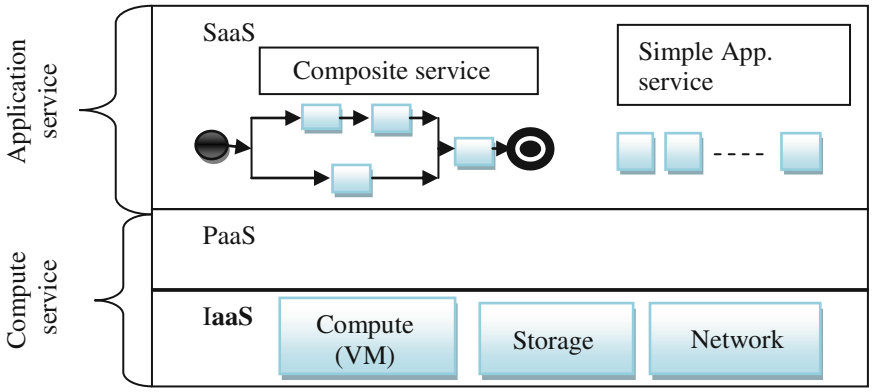


**Fig. 1.** Cloud system

## 2.2 Service Composition in Cloud Computing

A composite service is specified as a collection of abstract application services according to a combination of control-flow and data-flow. Similar to traditional service composition, cloud service composition is conducted in two steps. First, a composition schema is constructed for a composition request. Second, the optimal composition plan is selected. Control-flow graphs are represented using UML activity diagrams. Each node in the graph is an abstract application service. There are four control-flow patterns for defining a composite cloud service (composition schema), such as sequential, parallel, conditional and iterative (loop) patterns [8] (Fig. 2). Directed acyclic graphs (DAGs) are used to represent composition schema.

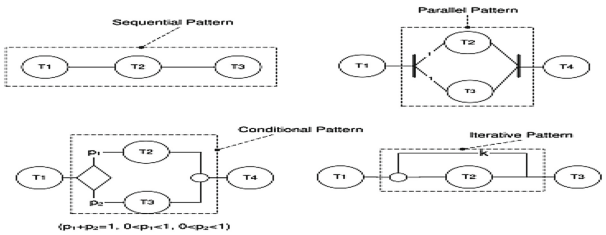


**Fig. 2.** Basic composition patterns

Any solution to a composition problem in cloud computing includes: (1) Map the abstract application services to concrete application services and corresponding compute service (VM, database and network services). (2) Schedule the execution order of the application services. In this research we introduce only the selection of the composition plan (abstract application services) based solely on user's QoS attributes.

## 2.3 QoS Model

QoS-based cloud service selection is a critical process that directly determines the QoS values of the composite cloud service. The ultimate goal of the selection process is to

find the best composition plan for the composite cloud service that provides the optimal choice to the end user. We use a QoS model that is applicable to all the SaaS and IaaS.

### 2.3.1 QoS Model for Elementary Services

We consider the three major QoS attributes which are: cost, response time, and throughput.

**Cost**: the amount of money that the end user has to pay to the service provider for using a cloud service S, it is denoted as $Q_{cost}(S)$. For SaaS provider, it is the execution cost for using a single request SaaS, for IaaS provider it is the computation cost for using a unit IaaS for one second.

**Response time:** the time interval of a cloud service S from request to response, it is denoted as $Q_{resp}(S)$. For SaaS provider SP, response time is the expected delay in seconds between the moment when a request is sent and the moment when the results are received. For IaaS provider IP, the response time is the number of CPU (network, storage) units used for processing a computation (data transfer, storage) request.

**Throughput:** the rate at which a cloud service S can process requests, it is denoted as $Q_{th}(S)$. For SaaS provider SP, the throughput is the number of requests the SaaS provider is able to process per second. For IaaS provider IP, the service rate is the number of CPU (network, storage) requests IaaS provider is able to process per sec.

### 2.3.2 QoS Model for Composite Services

In addition to the QoS attributes of each individual service, the composition pattern and corresponding aggregate functions should be also considered. The overall QoS can be defined as shown in Eq. 1.

$$QoS_{total}(S) = \left[Q_{cost}(S),\ Q_{resp}(S),\ Q_{th}(S)\right] \tag{1}$$

where S is a single path composite cloud service;$QoS_{total}(S)$ indicates the overall quality of a composed cloud service S. Each dimension is the aggregation of all services which is calculated by the aggregation functions (Table 1) [18].

**Table 1.** Aggregation functions

| | Sequential | Parallel | Conditional | Loop |
|---|---|---|---|---|
| Cost | $\sum_{i=1}^{n} Q_{cost}(i)$ | $\sum_{i=1}^{n} Q_{cost}(i)$ | $\sum_{i=1}^{n} (P(i) * Q_{cost}(i))$ | $K * Q_{cost}(i)$ |
| Response Time | $\sum_{i=1}^{n} Q_{resp}(i)$ | $\max_{i=1\ldots n} (Q_{resp}(i))$ | $\sum_{i=1}^{n} (P(i) * Q_{resp}(i))$ | $K * Q_{resp}(i)$ |
| Throughput | $\min_{i=1\ldots n} (Q_{th}(i))$ | $\min_{i=1\ldots n} (Q_{th}(i))$ | $\sum_{i=1}^{n} (P(i) * Q_{th}(i))$ | $Q_{th}(i)$ |

# 3 System Model

In this section, we highlight the service composition model, the problem formulation, and the cloud service composition problem.

## 3.1 Cloud Service Composition Model

The service composition model introduced in this paper is similar to the one given in [6]. In this model we identify four components: IaaS (Infrastructure as a Service) Providers, SaaS (Software as a Service) Providers, Composer and End users, see Fig. 3. Platform as a Service (PaaS) layer is omitted because we assume that this layer is included in the IaaS layer. The IaaS Providers supply IaaS, i.e., CPU services, storage services, and network services, to SaaS providers and end users. The SaaS providers supply SaaS to end users. The end users are usually large companies and organizations, e.g., universities, governments. The Composer is the proposed composition model that acts on behaves of the end users to form composite services that contain services from multiple SaaS providers and IaaS providers. The main functions of composer are: (1) construct composition schema. (2) Select optimal composition plan. Since the main concern in this research is the selection of optimal composition plan based solely on user's preferences (QoS attributes). We assume that existing composition techniques such as the one introduced in [7] to generate composition schema.

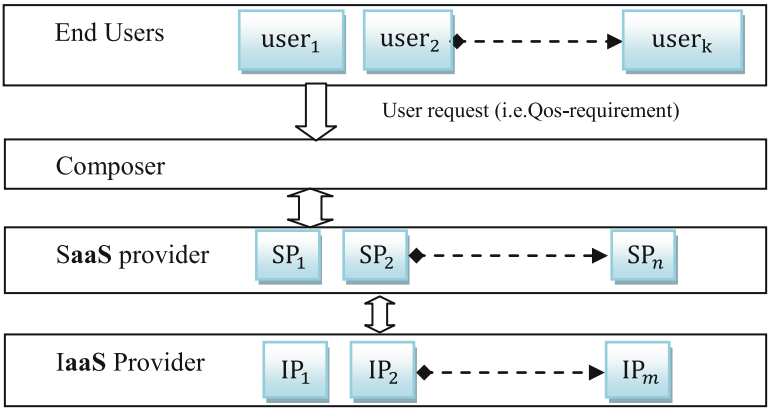


**Fig. 3.** The proposed composition model

## 3.2 Problem Formulation

This section introduces the cloud service composition problem as follow:

Given

1. A set of abstract cloud services (or tasks) T involved in a cloud service composition, where $T = \{T_1, T_2, \ldots..., T_n\}$ and n is the total number of abstract cloud services in the composition.

2. For task $T_i$, a set of $K_i$ candidate SaaS providers can be used to implement the task, $SP_i = \{ SP_i(1), SP_i(2), \ldots, SP_i(K_i)\}$ . A set of $P_p$ candidate IaaS providers supply IaaS to composite services: $SP_0 = \{ SP_0(1), SP_0(2), \ldots, SP_0(P_p)\}$
3. A candidate composition plan (denoted as Plan $[\{SP_0(K_0), SP_1(K_1), SP_2(K_2) \ldots SP_0(K_0)\}]$ is formed by selecting certain SaaS providers and IaaS providers for the end user.
4. In the composition plan, the composite service is supported by the IaaS provider $SP_0(K_0)$. Task $T_i$ is implemented by SaaS provider $SP_i(K_i)$.
5. We assume sequential composition pattern is used for composition and user's QoS attributes are numbered from 1 to 3, with 1 = cost, 2 = response time and 3 = throughput. The QoS values for a composition plan using the aggregation functions stated above is denoted as: $QoS_{total}$(plan) = [Q(1), Q(2),Q(3)]
6. Since Service composition in cloud is long-term based economically driven user's QoS requirement, The end users would have different QoS requirements (cost, response time, throughput) on the composite service during long period, i.e., the end user may prefer composite service that has less cost, while in another period the end user may find that the cost is less important than decreasing the response time as much as possible.

The Composite Cloud Service (CCS) is the problem whose goal is to choose one or more composition plan (execution plan) that have a good performance on most of the QoS factors and meet user requirements. [19] The CCS problem based on multiple QoS criteria is a combinational optimization problem that is known to be an NP-Hard [12]. The solution to this problem should optimize (minimize/ maximize) the components of the vector $QoS_{total}$(plan).

When the composition system makes decisions on which concrete SaaS providers and IaaS providers should be selected for the end user, it has no idea about how will the ultimate composite service behave during a long period. To enable long-term cloud service composition, economic models are needed to predict the long-term preferences of the end users. An economic model is defined as "a theoretical construct that represents economic processes by a set of variables and a set of logical and quantitative relationships between them". [9].

The final objective is to find a fully cloud service composition that minimizes the cost and time, and improves the throughput. We model the cloud service composition problem as a multi-objective optimization problem (MOP).

### 3.3 Cloud Service Composition Optimization Problem

In this problem, we consider the three user objectives, the lowest cost, the shortest response time, and the highest throughput. This makes it infeasible to find an optimal composition as these objectives can conflict with each other. One way to address this problem is to convert the composition problem to a single-objective problem by asking users to give weights to each objective. The end user presents these preferences through a Score Function [3]. We denote the QoS requirements of the end user as: $W_a(t) = [w1(t), w2(t), w3(t)]$ , where $W_a(t)$ denotes the weight of QoS attribute $(a)$ for the composite service at period $t$. Each composition plan is associated with a score from the

end user's perspective. A commonly used score function is the weighted sum of QoS values of the composite service. The main objective is to find an optimal composition plan *S (plan)* that provides the maximum score value.

$$S\ (\text{plan})\ = W_1(t) * + W_2(t) * Q_2(t) + W_3(t) * Q_3(t) \tag{2}$$

## 4 Related Works

Service composition problem can be categorized into two groups. One group focuses on the functional composition among component services. The other group aims to make optimal decisions to select the best Component services based on non-functional properties (QoS).

Functional-driven service composition approaches typically adopt semantic descriptions of services. Examples of automatic approaches include Policy-based approach proposed by [14]. Other functional-driven composition approaches use AI planning methods. Most of them [15] assume that each service is an action which alters the state of the world as a result of its execution. The inputs and outputs parameters of a service act as preconditions and effects in the planning context. Users only need to specify the inputs and the outputs of the desired composite service, a plan (or a composite service) would automatically generated by the AI planners. Different users may have different requirements and preferences regarding QoS. Therefore, QoS-aware composition approaches are needed. QoS-aware service composition problem is usually modeled as a Multiple Criteria Decision Making [3] problem. The most popular approaches include integer linear programming and genetic algorithms. An Integer Linear Program (ILP) consists of a set of variables, a set of linear constraints and a linear objective function. After having translated the composition problem into this formalism, Specific solver software such as LPSolve [13] can be used. References [16, 17] use Genetic Algorithms (GA) for service composition.

Most of the existing composition approaches are not well suited for cloud environment [17]. They usually consider the qualities at the time of the composition [4]. The proposed composition approach considers the problem from a long-term perspective.

## 5 Genetic Algorithm – Based Approach

Genetic Algorithms (GAs) [12] are heuristic approaches that iteratively find the nearest optimal solution in large search solutions. Any possible solution to the optimization problem is encoded as a Chromosome (genome). A set of chromosomes is referred to as a Population. The first step of a GA is to derive an initial population. A random set of chromosomes is often used as the initial population. This initial population is the first generation from which the evolution starts. The second step is the selection process, each chromosome is eliminated or duplicated (one or more times) based on its relative quality. The population size is typically kept constant. The next step is the Crossover

process. Some pairs of chromosomes are selected from the current population and some of their corresponding components are exchanged to form two valid chromosome. After crossover, each chromosome in the population may be mutated with some probability. The mutation process transforms a chromosome into another valid one. The new population is then evaluated and each chromosome is associated with a fitness value, which is a value obtained from the objective function. The objective of the evaluation is to find a chromosome that has the optimal fitness value. If the stopping criterion is not met, the new population goes through another cycle (iteration) of selection, crossover, mutation, and evaluation. These cycles continue until the stopping criterion is met [4].

### 5.1 Algorithm Implementation

**The Proposed Approach Consists of the Following Process:**

1. **Define Chromosome:** For genetic algorithms, one of the key issues is to encode a solution of the -problem into a chromosome (individual). In our model, the chromosome is encoded by an integer array with a number of items equals to the number of distinct abstract services that compose our service. Each item, in turn, contains an index to the array of the concrete services matching that abstract service.
2. **Generate Initial Population:** a predefined number of chromosomes are generated to form the initial generation. The chromosome in a generation is first ordered by their fitness values from the best to worst.
3. **Apply Genetic Operator:** To apply this process we define the operator of each step as following:
   - **Selection Operator:** We use the binary tournament selection as the selection operator. The binary tournament selection runs a tournament between two individuals and selects the winner. In this way, the individuals that formed the next generation are determined. The population size of each generation is always P.
   - **Crossover Operator:** We use the single-point crossover as the crossover operator. The crossover point is a random value from 1 to $N_t$ (the number of genes in one chromosome).
   - **Mutation Operator:** We randomly select an abstract service and randomly replace the corresponding concrete service with another one among those available. Clearly, we select the abstract service for which only one concrete service is available.
4. **Evaluate the Chromosomes Using Fitness Function:** The fitness function should maximize some QoS attributes (i.e. throughput), minimize some other attributes (i.e. cost and response time). In addition, the fitness function must penalize solutions that do not meet the QoS constraints and drive the evolution towards constraints satisfaction. Let us suppose that the composite service has a set of constraints defined as $QC$ [10]. we define $D(c)$, the distance from the constraint satisfaction of a chromosome (solution) $\boldsymbol{c}$, as following :

$$\mathrm{D(c)} = \sum_{i=1}^{l} QC^i(c) * e_i * weight^i, e_i = \left\{ \begin{array}{l} 0 QC^i(c) \leq 0 \\ 1 QC^i(c) > 0 \end{array} \right\} \quad (3)$$

where,

- $weight^i$ indicates the weight of the QoS constraint.
- $l$ is number of constraints that composite service has for a specific chromosome (solution) $\boldsymbol{c}$.

The fitness function for a chromosome $c$ is then defined as:

$$\mathrm{F}(c) = \sum_{i=1}^{a} w^i * Q^i(c) + weight_{\mathrm{p}} * \mathrm{D}(c) \quad (4)$$

where,

- $w^i$ is the weight corresponding to each QoS attribute $i$.
- $weight_{\mathrm{p}}$ is the penalty factor.
- A is number of Qos attributes which is 3 in our case.

The stop criterions of the proposed approach are:

(1) Iterate until the constraints are met (i.e. D(c) = 0).
(2) If this does not happen within a defined maximum number of generation, '*MAXGEN'*, then iterate until the best fitness value remains unchanged for a given number '*MAXGEN'* of generations.
(3) If neither (1) nor (2) happened within '*MAXGEN'* generations, then no solution will be returned.

## 6 Experiments and Evaluation

We run our experiments on a Dell laptop with 2.3 GHz Intel Core i7 processor and 6G Ram under Windows 7 operating system. We have used Jmetal [11] which is Java framework for implementing multi-objective and single-objective algorithms. We implement the Genetic Algorithm-based approach in this framework that allows us to define each problem by defining each chromosome (variable) to point to a cloud service candidate. After that, we define the fitness functions for all defined objectives and choose the algorithm that solves the problem. We first conduct one experiment to show the effect of changing available concrete services for each abstract service on the execution time, Fig. 4 shows that the execution time increases quickly at the beginning of the experiment, but it keeps nearly constant when the number of concrete services for each abstract service becomes larger than 200. We conduct the experiment that shows the effect of selection operator on reaching to the optimal fitness value. Figure 5 shows the comparison between the tournament selection operator used in the proposed approach and the random selection operator. As shown in Fig. 5, the proposed approach will always reach an optimized fitness value while random selection seldom

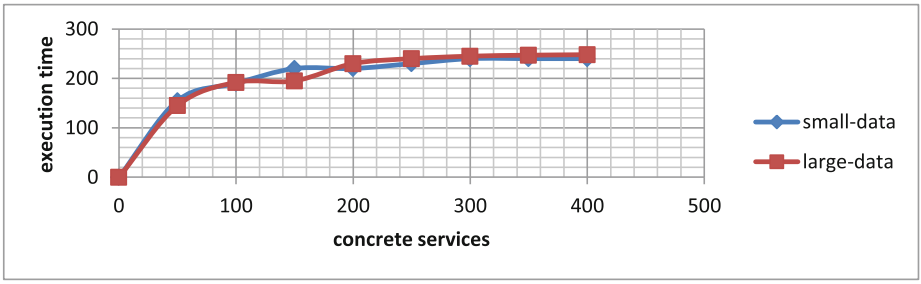


**Fig. 4.** Concrete services against execution time

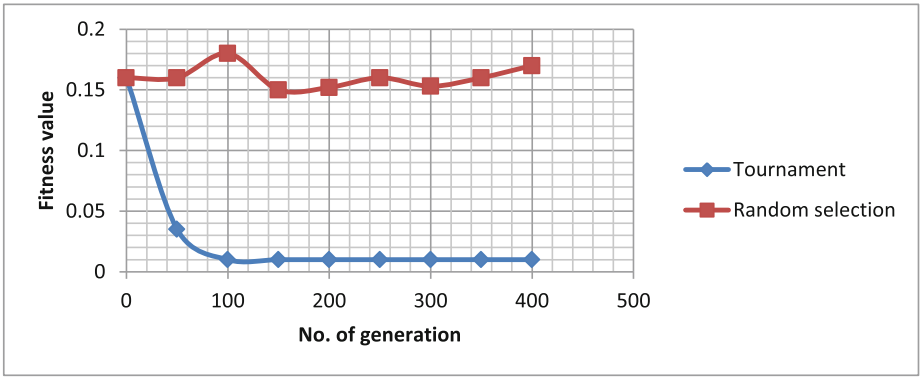


**Fig. 5.** Tournament selection against random selection

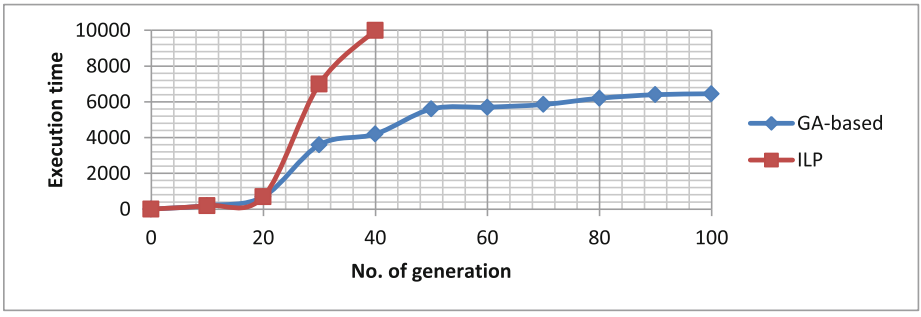


**Fig. 6.** GA-based against ILP

converges. The proposed GA based approach will always reach an optimal fitness value and the converged point becomes very close to the actual optimal point. We conduct a comparison experiment with another approach such Integer Programming. The Integer Programming (IP) approaches have been proposed to solve QoS-aware service composition. The IP approaches implemented using LPSolve [13], which is an open source integer programming system. Figure 6 shows that the IP approach performs as good as

the GA based approach at the beginning. Notice that, when the number of abstract services becomes more than 40, the execution time cost of the IP approaches will increase exponentially to solve composition problems.

## 7 Conclusions and Future Work

This paper discusses the cloud service composition problem as multi-objective optimization to satisfy the requirements of user's QoS requirements and presents an approach to solve the multi-objective problem by converting it to a single objective problem. We have proposed an approach that uses the Genetic Algorithm to solve this problem. The experiment results proved the efficiency of the proposed approach.

For the future work, we intend to solve the multi-objective composition problem using multi-objective algorithms such NSGA II (Non-dominated Sorting Genetic Algorithm), SPEA-II (Strength Pareto Evolution Algorithm) [20] and other meta-heuristics algorithm which is based on swarm intelligence such as Particle Swarm Optimization (OMOPSO) [21] and compare the efficiency of these algorithms.

**Acknowledgments.** This work was made possible by NPRP grant # 7 - 481-1 - 088 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

## References

1. Youseff, L., Butrico, M., Da Silva, D.: Toward a unified ontology of cloud computing. In: Grid Computing Environments Workshop (2009)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: a Berkeley view of cloud computing. Technical report, February 2009
3. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Trans. Softw. Eng. **30**(5), 311–327 (2004)
4. Srinivas, M., Patnaik, L.: Genetic algorithms: a survey. Comput. **27**(6), 17–26 (1994)
5. Canfora, G., Di Penta, M., Esposito, R., Villani, M.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1069–1075. ACM, New York (2005)
6. Ye, Z., Bouguettaya, A., Zhou, X.: QoS-aware cloud service composition based on economic models. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) Service Oriented Computing. LNCS, vol. 7636, pp. 111–126. Springer, Heidelberg (2012)
7. Medjahed, B., Bouguettaya, A., Elmagarmid, A.: Composing web services on the semantic web. VLDB J. **12**(4), 333–351 (2003)
8. Wu, B., Chi, C., Chen, Z., Gu, M., Sun, J.: Workflow-based resource allocation to optimize overall performance of composite services. Future Gener. Comput. Syst. **25**(3), 199–212 (2009)
9. Baumol, W., Blinder, A.: Economics: Principles and Policy. South-Western Pub, Mason (2011)

10. Canfora, G., Di Penta, M., Esposito, R., Villani, M.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1069–1075 (2005)
11. Durillo, J., Nebro, A.: jMetal: a java framework for multi- objective optimization. Adv. Eng. Softw. **42**(10), 760–771 (2011)
12. De Jong, K., Spears, W.M.: Using genetic algorithms to solve NP complete problems. In: Proceedings of the Third International Conference on Genetic Algorithm, pp. 124–132. Morgan Kaufman, Los Altos, CA (1989)
13. Berkelaar, M., Eikland, K., Notebaert, P., et al.: lpsolve: Open source (mixedinteger) linear programming system. Eindhoven U. of Technology
14. Chun, S.A., Atluri, V., Adam, N.R.: Using semantics for policy-based web service composition. Distrib. Parallel Databases **18**(1), 37–64 (2005)
15. Wu, D., Parsia, B., Sirin, E., Hendler, J., Nau, D.S.: Automating DAML-S web services composition using SHOP2. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 195–210. Springer, Heidelberg (2003)
16. Canfora, G., Di Penta, M., Esposito, R., Villani, M.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1069–1075 (2005)
17. Ye, Z., Zhou, X., Bouguettaya, A.: Genetic algorithm based QoS-aware service compositions in cloud computing. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part II. LNCS, vol. 6588, pp. 321–334. Springer, Heidelberg (2011)
18. Lie Q., Yan, W., Orgun, M. A.: Cloud service selection based on the aggregation of user feedback and quantitative performance assessment. In: Services Computing (SCC). IEEE (2013)
19. Jula, A., Sundararajan, E., Othman, Z.: Cloud computing service composition: a systematic literature review. Expert Syst. Appl. J. **41**, 3809–3824 (2014). Elsevier
20. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)
21. Sierra, M.R., Coello, C.A.: Improving PSO-based multi-objective optimization using crowding, mutation and $\varepsilon$-dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005)