QoS-aware long-term based service composition in cloud computing

Shengcai Liu*, Yufan Wei*, Ke Tang**, A. K. Qin‡, Xin Yao†
*USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI),
School of Computer Science and Technology, University of Science and Technology of China
‡School of Computer Science and Information Technology
Royal Melbourne Institute of Technology, Melbourne, Australia
†Center of Excellence for Research in Computational Intelligence and Applications (CERCIA),
School of Computer Science, The University of Birmingham
⋆ Corresponding author
Emails: liuscyyf@mail.ustc.edu.cn, wyf724@mail.ustc.edu.cn, ketang@ustc.edu.cn, kai.qin@rmit.edu.au, x.yao@cs.bham.ac.uk

***Abstract*—Cloud service composition problem (CSCP) is usually long-term based in practice. A logical request is to maximize end users' long-term benefit. Thus, the overall long-term QoS properties of the composite service should be optimized and the users' requirements during the period should be satisfied. However, the benefit-maximization has not been considered under the background of long-term based CSCP in existing research yet. To fill this gap, in this paper, a new formulation LCSCP is proposed to define the long-term based CSCP as an optimization problem. Then, for the sake of efficiency, three meta-heuristic approaches (i.e, Genetic Algorithm, Simulated Annealing and Tabu Search) are studied. Comprehensive experiments are designed and conducted to test their various aspects of performance on different test sets with different workflows. Experimental results provide a basic perspective of how these three widely adopted meta-heuristic frameworks work on this new problem, which can be baseline work for further research.**

***Index Terms*—long-term, QoS-aware, cloud service composition, optimization**

I. Introduction

Cloud computing is quickly becoming the competitive alternative technology platform for conducting business in recent years [1], and it has attracted more and more attention in both academy research and industry [2]. Some cloud platforms such as Microsoft Azure by Microsoft, Amazon Web Service by Amazon and Google App Engine by Google have been cast into commercial use already. The importance of affordable access to reliable high-performance hardware and software resources and avoiding maintenance costs and security concerns have encouraged many large institution managers and stakeholders of information technology companies to migrate to cloud computing. They need not be concerned about overprovisioning for a service whose popularity does not meet their predictions in cloud computing, thus wasting costly resources, or underprovisioning for one that becomes wildly popular, thus missing potential customers and revenue [3].

Cloud service composition (CSC) is originally a mechanism to build up meta-services with simple function to a complex composite service functionally satisfying the end users' requirements. This can be quite simple when meta-service of each function is unique. However, the primary reality of the cloud world is that no cloud service stands alone [4]. There is a trend of cloud booming recently, and this impact brings out a challenge: more and more meta-service from different service providers have the same or overlapping functional characteristics but vary a lot in non-functional (i.e. Quality of Service, QoS) features. Thus, cloud service composition problem (CSCP): how to select among huge numbers of functionally overlapped cloud meta services in a QoS-aware environment to satisfy end users' non-functional requirement, is a critical issue in cloud computing [5]. Besides, given the on-demand nature of cloud computing, by modeling for the correlation between their Quality of Service (QoS) and cloud configurations/resources [6], cloud service providers only have to concern about CSCP.

Many approaches [7], [8], [9], [10] have been proposed to solve QoS-aware service composition problem in recent years. However, existing researches usually only consider the QoS performance of the composite service at the composing time while ignoring the instability in cloud environment, which does not comply with the situation in practice. Due to economical considerations, large companies and organizations usually are more willing to build long-term business relationship with cloud service providers [1], [11]. On the other hand, the end users' behaviour (i.e, QoS requirements) and the selected services' performance (i.e, QoS attributes) are both fluctuant in practice [12]. As a result, the composite service with the best current QoS performance is not necessarily the best after a period of time. This point was first considered in [12] in which an economical model is adopted to predict end users' QoS requirements in the future. Then in the following work [13], time series analysis is introduced and CSCP is transformed to a similarity search problem. However, finding the composite service that has the most similar QoS properties to end users' QoS requirements is not a benefit-maximization way. To meet users' actual needs better, CSCP should be solved by optimizing the QoS performance of the composite service in a long-term based mechanism, which has not been considered in existing researches yet.

In this paper, to fill the gap, the CSCP is modeled as an optimization problem based on a long term style. The contributions of this paper include: 1) A new formulation LCSCP is proposed to define the long-term based CSCP as an optimization problem; 2) Three widely used meta-heuristic approaches (i.e, Genetic Algorithm, Simulate Annealing and Tabu Search) are studied to tackle this new problem. Comprehensive experiments are designed and conducted to test their sensitivity to parameters and their performance on different test sets and different workflows. Experimental results provide a basic perspective of how these three frameworks work on this new problem, which can be a baseline work for further research.

The rest of the paper is organized as follow: Section II describes the related work. Section III elaborates the optimization formulation LCSCP. Section IV depicts the three meta-heuristic approaches. Section V evaluates the approaches and shows experimental results. Section VI concludes with a summary and possible future work.

II. Related Work

In service composition, end user's request is first decomposed into subtasks which need to be fulfilled according to a designed workflow. The subtask is also noted as abstract service (AS) which represents a virtual service with a specific functionality. For each abstract service, a candidate service (CS) needs to be selected from a set of services provided by service providers that have the equivalent corresponding functionality but vary a lot in non-functional properties, i.e, QoS (quality of service). Choosing different candidate services for abstract services will affect the QoS properties of the composite service, which are the optimization object and the constraints.

QoS-aware service composition problem can be modeled as a multidimensional, multi-choice knapsack problem (MMKP) [8] which has been proved NP-hard. Many approaches have been proposed to solve it during last decade. In [7] the composition problem is transformed to a integer linear programming (ILP) whose constraints and objective function are linear, and then adopts specific software solver such as LPSolve [14] to solve it. However, the integer program approach needs exponential time as the problem scale increases. In order to overcome the time cost shortcoming, some approaches [9], [15] have been proposed. Genetic algorithm-based approaches are employed in [9], [15] to quickly find near-optimal solutions, in which each gene of the chromosome represents the abstract service of a composite service and its value represents a candidate service. Besides, there are alternative ways to resource allocation in clouds, e.g., in [10] an economics-inspired approach is proposed which not only aims to achieve efficiency, but also takes into consideration the adaption and resilience aspects.

Compared to traditional service composition, cloud service composition is usually long-term based in practice [1], [11]. In long-term based CSCP, the overall QoS performance of the composite service during the long period should be involved instead of the performance only at the composing time. This point was first considered in [12] in which Bayesian Network is adopted to predict end users' QoS requirements in the future, and an influence diagram is employed to model the long-term based CSCP. However, they assume conditional probability relationship among multiple QoS attributes, which does not comply with reality. As a result, the cloud service composition model adopted in [12] can not represent the general case in practice. In [13], time series group (TSG) is introduced to represent the long-term QoS properties and CSCP is transformed to a similarity search problem. However, finding the composite service that has the most similar QoS performance to end users' QoS requirements is not a benefit-maximization approach. In general, considering service composition as an optimization problem is more in line with users' need. However, the benefit-maximization has not been considered under the background of long-term based CSCP in existing research yet.

III. Problem Formulation

In this section, the new formulation LCSCP in which long-term based CSCP is considered as an optimization problem is proposed. First the basic concepts are introduced, and then the specified long-term based QoS model is explained in detail. Based on these notations, the constraints and objective function of the formulation is built finally.

A. Basic concepts

Cloud computing providers offer their services according to three fundamental models [16], i.e, *SaaS* (software as a service), *PaaS* (platform as a service), *IaaS* (infrastructure as a service). In *SaaS*, end users are provided access to application services such as flight booking services and hotel booking services. Cloud service providers manage the infrastructure and platforms that run the applications, which simplifies maintenance and reduces support costs for end users [17]. Furthermore, to make these applications work together smoothly, some utility services are needed. These cloud services are usually considered as platform and infrastructure, i.e, *PaaS* and *IaaS*. In this paper, we consider platform service and infrastructure service together as utility service.

In CSCP, to accomplish end user's request, the overall task is first decomposed into subtasks which need to be fulfilled according to a designed workflow, and each subtask is accomplished by an abstract application service (AAS). There are four basic control structures in workflow, i.e, sequential, parallel, optional and loop. Fig. 1 gives an example of a completely sequential workflow, the overall task T can be

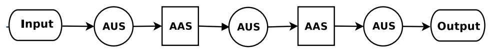


Fig. 1: a sequential example of workflow

accomplished by executing two $AASs$ sequentially, and the abstract utility services (AUS) provide the necessary infrastructure and data processing between adjacent $AASs$. Three kinds of utility services are involved in this paper, i.e, *network* transferring data between application services, *storage* storing the intermediate data and *cpu* adjusting the last application service's output data to the next application service's input data, denoted as AUS_1, AUS_2, AUS_3 respectively.

Given a workflow, assuming there are n $AASs$, denoted as set $A = \{AAS_i | 1 \leq i \leq n\}$, and $3 \cdot q$ $AUSs$, denoted as set $U_t = \{AUS_{t,i} | 1 \leq i \leq q\} (1 \leq t \leq 3)$, in which $AUS_{t,i}$ belongs to the corresponding $AUS_t (t = 1, 2, 3)$ type. For each $AAS_i \in A$, a set of k_i candidate application services (CAS) $CA_i = \{CAS_i^j | 1 \leq j \leq k_i\}$ are available to implement it. For all the $AUS_{t,i} \in U_t$, a set of x_t candidate utility services (CUS) $CU_t = \{CUS_t^j | 1 \leq j \leq x_t\}$ are available to implement them. That is to say, the candidate service set is the same for all the $AUSs$ that belong to the same utility type. This is reasonable since the utility services such as network and storage are underlying services in reality which can be used in application-independent scenario. Based on these notations, the workflow is denoted as a list wf:

$$wf = (AAS_1 .....AAS_n AUS_{1,1} .....AUS_{1,q} AUS_{2,1} .....AUS_{2,q} AUS_{3,1} .....AUS_{3,q}) \tag{1}$$

For each abstract service in wf, a concrete implementation needs to be selected from corresponding candidate service set. Once the selection is done, the composite service (cs) is constructed completely, denoted as follow:

$$cs = (a_1 .....a_n u_{1,1} .....u_{1,q} u_{2,1} .....u_{2,q} u_{3,1} .....u_{3,q}) \tag{2}$$

in which each element represents the index of corresponding selected candidate service in the candidate set. The whole search space is denoted as a set D:

$$D = \{cs | 1 \leq a_{ia} \leq k_{ia}, 1 \leq u_{t,iu} \leq x_t, 1 \leq ia \leq n, 1 \leq t \leq 3, 1 \leq iu \leq q\}. \tag{3}$$

The objective in CSCP is to find the cs with the best long-term QoS properties from D.

B. QoS model

The QoS parameters considered in this paper are response time, cost, availability and reliability, denoted as q_1, q_2, q_3, q_4 respectively. Different from traditional service composition problem, the QoS performance of composite service needs to be measured from a long-term perspective in CSCP. In this paper, time series method is adopted to describe the long-term data. Time series is widely used in any domain of applied science and engineering which involves temporal measurements, and it is firstly introduced by [19] to CSCP. Here the QoS properties of the candidate services are represented as a time series group, which is a set of time series $\{Q_1, Q_2, Q_3, Q_4\}$, where each time series $Q_j = \{q_{i,j} | i = 1, 2, 3, ...L_j\}$, where $q_{i,j}$ is the value of QoS parameter q_j at the i^{th} sampling point assuming q_j is sampled for L_j times in total. To simplify the problem, the starting sampling point, sampling interval, and

	sequential	**parallel**	**optional**	**loop**
$T(q_1)$	$\sum_{i=1}^{m} t_i$	$\max_{i=1}^{m} t_i$	$\sum_{i=1}^{m} t_i * p_i$	$t * loop\_times$
$C(q_2)$	$\sum_{i=1}^{m} c_i$	$\sum_{i=1}^{m} c_i$	$\sum_{i=1}^{m} c_i * p_i$	$c * loop\_times$
$A(q_3)$	$\prod_{i=1}^{m} a_i$	$\prod_{i=1}^{m} a_i$	$\prod_{i=1}^{m} a_i * p_i$	a^{loop_times}
$R(q_4)$	$\prod_{i=1}^{m} r_i$	$\prod_{i=1}^{m} r_i$	$\prod_{i=1}^{m} r_i * p_i$	r^{loop_times}

TABLE I: Aggregation function for each structure. m is the sum number of both AAS and AUS

sampling number of each parameter are considered the same, i.e, $L_j (j = 1, 2, 3, 4) = L$. Thus the time series group can be denoted as a $L \times 4$ matrix $\boldsymbol{TSG}$:

$$\boldsymbol{TSG} = \begin{bmatrix} q_{1,1} & q_{1,2} & q_{1,3} & q_{1,4} \\ q_{2,1} & q_{2,2} & q_{2,3} & q_{2,4} \\ ..... & ..... & ..... & ..... \\ ..... & ..... & ..... & ..... \\ q_{L,1} & q_{L,2} & q_{L,3} & q_{L,4} \end{bmatrix} \tag{4}$$

For each candidate application service $CAS_i^j \in CA_i (1 \leq i \leq n)$, its QoS properties are represented as a time series group, denoted as $\boldsymbol{TSG}(CAS_i^j)$. However, it is different when representing the QoS of CUS. Although the CUS set is the same for all the $AUSs$ that belong to the same utility type for the reason that utility services are used application-independent, the QoS of the CUS are application-dependent. For example, the response time of a specific network service may be different when serving different transmission terminals [18]. More specifically, if $CUS_t^j \in CU_t (1 \leq t \leq 3)$ is selected and it connects CAS_a^b and $CAS_{a'}^{b'}$ in the composite service, the QoS properties of CUS_t^j are jointly determined by itself, CAS_a^b and $CAS_{a'}^{b'}$, denoted as $\boldsymbol{TSG}(CUS_t^j (CAS_a^b, CAS_{a'}^{b'}))$. In this way, the QoS properties of different candidate services are dependent.

The QoS properties of sc are also represented as a $L \times 4$ matrix, denoted as $\boldsymbol{TSG}(sc)$. Each entry in $\boldsymbol{TSG}(sc)$ is computed by aggregating the corresponding entries in the $\boldsymbol{TSGs}$ of the selected candidate services in sc. The aggregation functions used in this paper are derived from the ones adopted in [7] and have been slightly modified to be applied in CSCP. Tab. I describes how the aggregation functions work in different basic control structures.

C. Constraints

The constraints in long-term based CSCP are constructed based on users' requirements, which are represented as a time series group too, denoted as a $L \times 4$ matrix $\boldsymbol{TSG}(user)$. The QoS parameters considered in this paper are categorized in two types. Response time (q_1) and cost (q_2) are one type of parameters which need maximizing while availability (q_3) and reliability (q_4) are the other type of parameters which need minimizing. Thus users refer to the entries $\boldsymbol{TSG}(user)[i, j] (1 \leq i \leq L, j = 1, 2)$, i.e, the values of

q_1(response time) and q_2(cost), as the tolerate upper limit, and the entries $\boldsymbol{TSG}(user)[i,j](1 \le i \le L, j = 3,4)$, i.e, the values of q_3 (availability) and q_4 (reliability), as the tolerate lower limit. More specifically, the constraints are constructed as follows: Assuming $L_0 > 0$ and $L/L_0 = m \in \mathbb{N}^+$, we define two $m \times 4$ matrix $\boldsymbol{TSG'}(cs)$ and $\boldsymbol{TSG'}(user)$, in which each entry $[i,j]$ is computed as :

$$\boldsymbol{TSG'}(M)[i,j] = \frac{1}{L_0} \sum_{s=(i-1)\cdot L_0+1}^{i\cdot L0} \boldsymbol{TSG}(M)[s,j] \quad (5)$$

where M refers to cs or $user$. Based on these notations, constraints in long-term based CSCP are described as :

$$\begin{aligned} &\boldsymbol{TSG'}(cs)[i,j] - \boldsymbol{TSG'}(user)[i,j] \ge 0, j = 1,2 \\ &\boldsymbol{TSG'}(cs)[i,j] - \boldsymbol{TSG'}(user)[i,j] \le 0, j = 3,4 \end{aligned} \quad (6)$$

The user-defined L_0 represents the number of successive sampling points which are involved in calculating average value. Intuitively L_0 controls the intensity of the constraints. When $L_0 = 1$, the average value is calculated on all sampling points, which means the only constraint is that user's average requirements of whole period need satisfying. When $L_0 = L$, the average value is calculated at each sampling point, that is to say, the constraints request user's requirements at each sampling point should be strictly satisfied, which means the number of constraints are $4 \cdot L$.

D. Objective function

In order to evaluate the solution quality of cs, firstly the average value of each parameter is calculated, denoted as $\overline{q} = (\overline{q_1}, \overline{q_2}, \overline{q_3}, \overline{q_4})$:

$$\overline{q_j} = \frac{1}{L} \sum_{i=1}^{L} \boldsymbol{TSG}(cs)[i,j], j = 1,2,3,4 \quad (7)$$

Since the scales of different QoS parameters are different, the Simple Additive Weighting (SAW) method is adopted to normalize $\overline{q_i}$, similar to the one used in [7]. The normalized $\overline{q_i}(1 \le i \le 4)$ is computed so that the best values are equal to 0, the worst equal to 1. Thus, lower normalized values indicate better solution quality. The normalized values are denoted as: $\overline{Q} = (\overline{Q_1}, \overline{Q_2}, \overline{Q_3}, \overline{Q_4})$. Based on these notations, the evaluation of solution cs is defined as:

$$E(cs) = \sum_{i=1}^{4} w_i \times \overline{Q_i} \quad (8)$$

where w_i is the weight of parameter q_i, and $\sum_{i=1}^{4} w_i = 1$.

Given all the above notations and the aforementioned constraints and evaluation, we now arrive at the following new formulation of long-term based CSCP, denoted as LCSCP:

$$\begin{aligned} &\min_{cs \in D} E(cs) \\ &s.t. : \\ &\boldsymbol{TSG'}(cs)[i,j] - \boldsymbol{TSG'}(user)[i,j] \ge 0, j = 1,2 \\ &\boldsymbol{TSG'}(cs)[i,j] - \boldsymbol{TSG'}(user)[i,j] \le 0, j = 3,4 \end{aligned} \quad (9)$$

Different from existing research, the newly-proposed LCSCP is an optimization problem. The constraints and objective function are constructed based on the introduction of the long-term data, which brings about new problem characteristics. The size of search set D is $O(k^n x^{3q})$, in which q is the number of each type of AUS. Considering the structure of cs, similar to traditional service composition problem, the LCSCP can be easily transformed to a MMKP problem which has been proved NP-hard. However, due to the introduction of the long-term data, the objective function is more complicated and the problem has much more constraints. Besides, different from traditional service composition, the QoS properties of different candidate services are dependent in LCSCP. Based on above two points, intuitively LCSCP is even harder than traditional service composition problem.

IV. Meta-heuristic Approach

With the formulation of LCSCP, in this section an exploratory attempt is made to tackle this new problem, which can be a baseline work for future research. Since there are no specified approaches designed for solving LCSCP at present, several approaches are considered in this paper, including a Genetic Algorithm (GA), a Simulated Annealing (SA) and a Tabu search (TS). The reasons for choosing these three meta-heuristic frameworks are: 1) Meta-heuristics make few assumptions about the optimization problem being solved, and so they may be usable for a variety of problems [19]; 2) With the increasing use of clouds, efficiency is becoming a crucial factor in cloud environment, and previous researches [9], [20], [21] have shown these three meta-heuristics are effective in finding near-optimal solutions for traditional service composition problems.

Although these three meta-heuristics adopt different strategies in searching process, some basic principles are shared. In the following subsections we will first elaborate the common settings, then the exclusive features of each approach will be introduced in detail.

A. Common Settings

1) *Encoding.* In GA, encoding is also known as chromosome, which typically takes the form of binary strings. However, in this paper integer string is adopted for a better description of LCSCP. Solution in all three meta-heuristic approaches is encoded as the definition of cs, denoted as sol

$$sol = (a_1.....a_n u_{1,1}.....u_{1,q} u_{2,1}.....u_{2,q} u_{3,1}.....u_{3,q}) \quad (10)$$

in which each element represents the index of corresponding selected candidate service in the candidate set.

2) *Mutation/Move.* Mutation is one of the operators (selection, crossover, mutation, etc.) for GA, and it can be used to generate an offspring based on current individual. The operator move is used for generating neighbors in both SA and TS. In this paper both of them take the same operation. They randomly select an element in sol, and then change it randomly.

3) *Initialization.* Completely random initialization is adopted in all three approaches, i.e, each element in initial solutions is randomly generated.

4) *Fitness Evaluation.* It is also referred to as fitness function in GA. In this paper the following formula is used to calculate the fitness of each sampled solution in search process

$$fitness(sol) = E(cs) + \lambda \times penalty \tag{11}$$

in which $E(cs)$ is computed as stated in Eq.8, λ is the penalty factor parameter and the $penalty$ item is calculated based on a $L_0 \times 4$ matrix $\boldsymbol{cp}$, which is calculated as:

$$\boldsymbol{cp}[i,j] = \overline{\boldsymbol{TSG'}}(user)[i,j] - \overline{\boldsymbol{TSG'}}(cs)[i,j] \tag{12}$$

where $\overline{\boldsymbol{TSG'}}$ is the normalization result of $\boldsymbol{TSG'}$, and $\boldsymbol{TSG'}$ is calculated as stated in Eq.5. Since lower normalized values indicate better solution quality, in $\boldsymbol{cp}$ each entry greater than 0 means a constraint is violated. Then $penalty$ item is computed based on $\boldsymbol{cp}$:

$$penalty = \sum_{\boldsymbol{cp}[i,j]>0} \boldsymbol{cp}[i,j] \tag{13}$$

B. Specialized Algorithms

Since the attempts made here is exploratory and original, simple and primary operators are adopted in these algorithms. In this way, the results may provide a basic understanding of the characteristics of LCSCP and how the basic versions of these meta-heuristics work on this new problem. Based on this work, LCSCP can be solved better in future research.

1) Simulated Annealing: The pseudo code of SA approach is presented in Alg. 1. Temperature T is initialized according to the geometric cooling schedule [22]. It is computed as the difference between the maximum and the minimum fitness. Since the fitness of each solution is calculated as Eq.11, Eq.12 and Eq.13, in which $E(sol) \in [0,1]$ and each entry $\boldsymbol{cp}[i,j](1 \leq i \leq L_0) \in [0,1]$, theoretically the minimum fitness is 0 and the maximum fitness is $L_0 \cdot 4 + 1$. Thus the initial temperature is:

$$inital\_T = L_0 \cdot 4 + 1 \tag{14}$$

In the search process, the generated neighbor sol' is accepted as base solution for the next iteration either if it is better than sol or if the Metropolis criterion [22] is satisfied:

$$rand(0,1) < exp(-(fitness(sol') - fitness(sol)))/T \tag{15}$$

in which $rand(0,1)$ generates a random number in $(0,1)$. At the end of each iteration, temperature T is updated following the geometric cooling schedule again, it is updated as: $T = T \cdot \alpha, \alpha \in [0,1)$.

2) Tabu Search: The implementation of TS approach is presented in Alg. 2. Tabu list is adopted to avoid entrainment in cycles in search process. It records the recently operated moves so that these moves are not allowed when neighbors are generated. Each move on Tabu list will be reserved for a fixed time, i.e, Tabu tenure. Moreover, aspiration rule is adopted that a move which was recorded by Tabu list will be allowed if it will generate a solution better than the currently-known best one.

Algorithm 1 Simulated Annealing

```
1:  procedure SA_APPROACH
2:      Initialize temperature $T$
3:      Generate initial solution $sol$ randomly
4:      Evaluate $sol$
5:      $bestsol$ = $sol$
6:      while stop criterion not met do
7:          Generate $sol'$ by operating move $m$ on $sol$
8:          Evaluate $sol'$
9:          if Accept $sol'$ then
10:             $sol$ = $sol'$
11:             if $sol$ is better than $bestsol$ then
12:                 $bestsol$ = $sol$
13:             end if
14:         end if
15:         Update T
16:     end while
17:     return $bestsol$
18: end procedure
```

Algorithm 2 Tabu Search

```
1:  procedure TS_APPROACH
2:      Generate initial solution $sol$ randomly
3:      Initialize $TabuList$
4:      Evaluate $sol$
5:      $bestsol$ = $sol$
6:      while stop criterion not met do
7:          $nextsol$ = $null$
8:          for each neighbor $sol'$ do
9:              Evaluate $sol'$
10:             if move($sol \to sol'$) is not banned then
11:                 if $sol'$ is better than $nextsol$ then
12:                     $nextsol$ = $sol'$
13:                 end if
14:             else
15:                 if $sol'$ is better than $bestsol$ then
16:                     $nextsol$ = $sol'$
17:                 end if
18:             end if
19:         end for
20:         Record move($sol \to nextsol$) on $TabuList$
21:         Update $TabuList$
22:         $sol$ = $nextsol$
23:         if $sol$ is better than $bestsol$ then
24:             $bestsol$ = $sol$
25:         end if
26:     end while
27:     return $bestsol$
28: end procedure
```

3) Genetic Algorithm : The implementation of GA approach is presented in Alg. 3. Elitism selection is used in population survival to prevent loss of best found solution. Random selection scheme is used when selecting members for mutation and crossover. The parameter s is the population size; γ is the fraction of the population to be replaced by crossover in each iteration, and μ is the mutation rate. One-point crossover is adopted in this implementation.

Algorithm 3 Genetic Algorithms

```
1: procedure GA_APPROACH
2:     Initialize s members randomly as population $P$
3:     Evaluate each member in $P$
4:     $P'$ = NULL
5:     while stop criterion not met do
6:         Select best $(1-\gamma)*s$ members from $P$ and insert into $P'$
7:         Randomly select $\gamma * s$ members from $P$ with no duplicate
8:         Pair them up and operate one-point crossover
9:         Insert the offspring into $P'$
10:        Randomly select $\mu * s$ members from $P'$
11:        for each selected member $i$ do
12:            Operate mutation
13:        end for
14:        $P$ = $P'$
15:        Evaluate each member in $P$
16:    end while
17:    return $bestmember$
18: end procedure
```

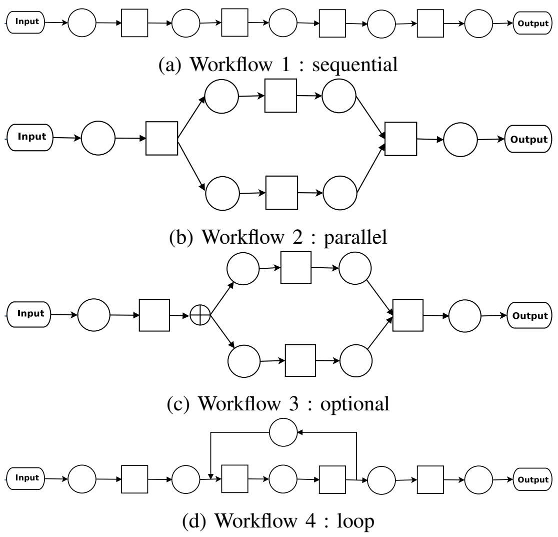
V. EXPERIMENTAL STUDIES

A. Overview

We design and conduct two groups of experiments to test the performance of the three meta-heuristic approaches on LCSCP. Group 1 is designed to test whether the approaches are sensitive to parameters, and group 2 is aimed at comparing the best performance of each approach.

B. Data Sets

Four workflows are constructed and adopted in this experiment. As a baseline data set, each workflow contains one basic control structure aiming at observing the behaviours on elemental workflows of the three approaches. As shown in Fig. 2, square represents AAS and circle represents AUS. In workflow 3 the probability of each branch is set to 0.5. In workflow 4 the loop structure will run for twice.



(a) Workflow 1 : sequential

(b) Workflow 2 : parallel

(c) Workflow 3 : optional

(d) Workflow 4 : loop

Fig. 2: Adopted worklfows in experiment

TABLE II: Proper ranges of after-aggregation QoS for each workflow

	workflow 1	workflow 2	workflow 3	workflow 4
$T(q_1)$	4300, 5260	3342, 4152	3243, 4041	6768, 7866
$C(q_2)$	460.0, 492.0	534.3, 564.7	358.1, 384.7	725.6, 762.2
$A(q_3)$	0.206, 0.442	0.260, 0.278	0.263, 0.272	0.178, 0.264
$R(q_4)$	0.206, 0.442	0.260, 0.278	0.263, 0.272	0.178, 0.264

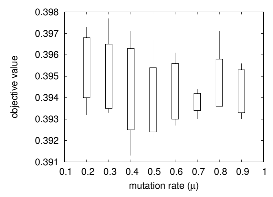
The number of sampling points L is set to 70 while the user-defined L_0 is set to 10. For each AUS, the size of the candidate set is the same and it is set to 10. Assuming there are n $AASs$, for each of them, the size of the candidate set is the same, denoted as k, k ranges from 60 to 300 at a step size of 30. For each CAS, the QoS parameters of the entries in $\boldsymbol{TSG}$ are randomly generated with uniform distribution from the following intervals: response time (q_1) $[100-400]$, cost (q_2) $[20-30]$, availability (q_3) $[0.90-1]$ and reliability (q_4) $[0.90-1]$. Note the QoS properties of CUS are jointly determined by itself and the $CASs$ it connects, in experiment, for each AAS, the candidate set CA is divided into $k/10$ groups and each group has 10 $CASs$. Thus for each CUS, the values in $\boldsymbol{TSG}$ are determined by the CUS itself and the groups that the $CASs$ it connects belong to, and each entry in the $\boldsymbol{TSG}$ is generated in the same way as CAS. Additionally, we generate users' requirements $\boldsymbol{TSG}(user)$ in a structural way to guarantee the existence of feasible solutions. Through experiment observation, for the entries in $\boldsymbol{TSG}(user)$, the proper range of each QoS parameter with each workflow is listed in Tab. II. It can be proved that the after-aggregation QoS values of $60\%-80\%$ of $cs \in D$, i.e, entries in $\boldsymbol{TSG}(cs)$, are within these ranges. The QoS parameters of the entries in $\boldsymbol{TSG}(user)$ are randomly generated within these ranges.

C. Protocol

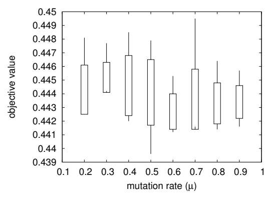
Stop criterion is set to the number of sampled solutions being 250000. The weight of each QoS parameter $w_1 = w_2 = w3 = w_4 = 0.25$. In each test the tested algorithm will run for ten times.

The experiment consists of two groups. Group 1 is designed to test each algorithm's sensitivity to parameters. We have tested the performance of each algorithm with different parameter settings. For convenience, for each algorithm, only one parameter which has the most significant impact is selected to be tested in group 1. For SA approach, parameter α ranges from 0.15 to 0.9 with a step size of 0.15. For TS approach, parameter $Tabu\ tenure$ ranges from 20 to 100 with a step size of 10. For GA approach, mutation rate μ ranges from 0.2 to 0.9 with a step size of 0.1. Empirical values are adopted for other parameters here: penalty factor λ is set to 50, in GA, population size s is set to 100 and γ is set to 0.8, here the size of CA are fixed to 180.

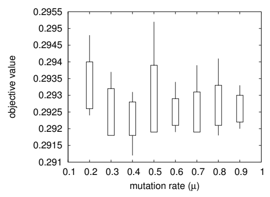
In group 2 the best performance of each approach is tested on each workflow and each size of test set. The best parameter settings summarized from group 1 are used in this group. Experimental results are shown in subsection V-D.



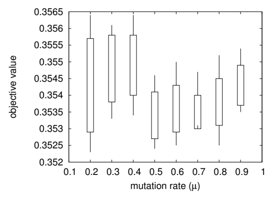
(a) GA Approach workflow=1



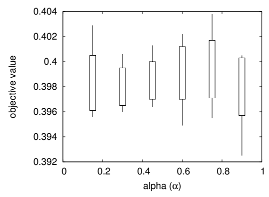
(b) GA Approach worklfow=2



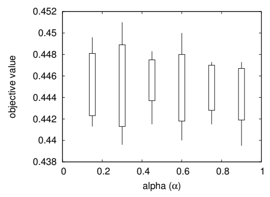
(c) GA Approach workflow=3



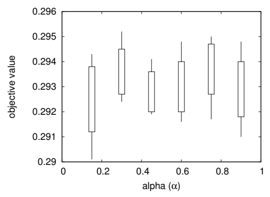
(d) GA Approach workflow=4



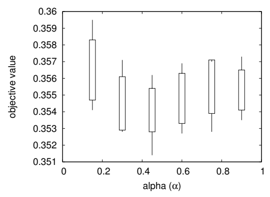
(e) SA Approach worklfow=1



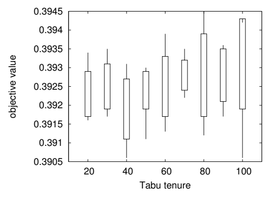
(f) SA Approach worklfow=2



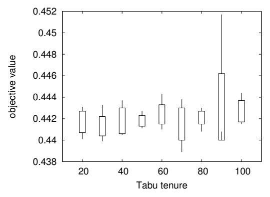
(g) SA Approach worklfow=3



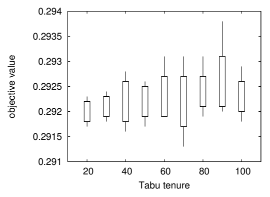
(h) SA Approach workflow=4



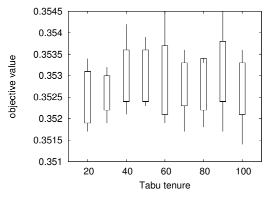
(i) TS Approach worklfow=1



(j) TS Approach worklfow=2

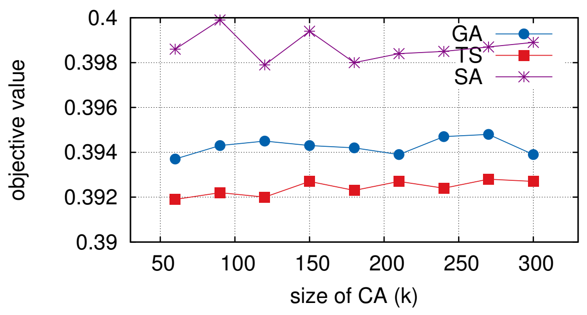


(k) TS approach worklfow=3

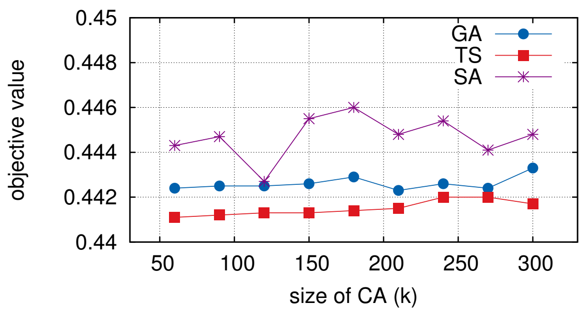


(l) TS Approach worklfow=4

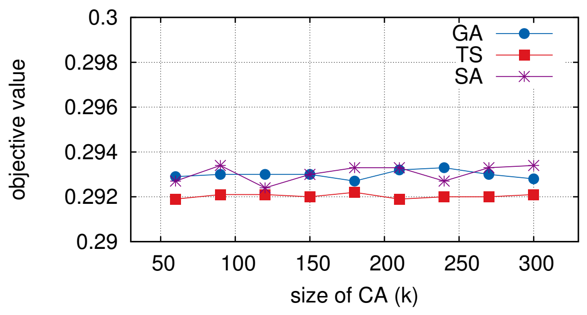
Fig. 3: Performance of each approach on each workflow with different parameter settings. For certain parameter setting and approach, the minimum/maximum of the whisker represents the minimum/maximum objective value of the 10 runs, and the minimum/maximum of the box represents mean value $-/+$ standard deviation of the 10 runs.



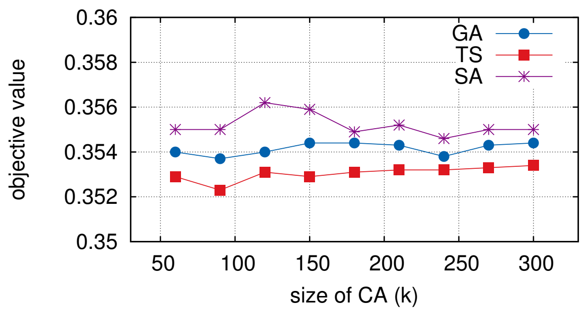
(a) performance with workflow 1



(b) performance with workflow 2



(c) performance with workflow 3



(d) performance with workflow 4

Fig. 4: Objective value of each approach on each test size with each workflow. Objective value is the evaluation of solution, i.e $E(sol)$, which does not contain the penalty item. For certain test size and approach, the corresponding point represents the mean objective value of the 10 runs.

D. Results

Fig. 3 shows the performance of each approach with different parameter settings for each workflow separately. Solutions returned in group 1 are all feasible. It can be seen the performance is close with different parameter settings, even when the range of the corresponding parameter is large enough. It can be concluded that, on LCSCP, these three meta-heuristic approaches are not sensitive to those parameters considered in this paper.

Fig. 4 depicts the performance of each approach on different sizes of test sets for each workflow. All the three approaches can find feasible solutions when the stop criterion is met. Intuitively, the rankings of the three approaches are TS approach, GA approach, SA approach. The search space D is too large compared to the limit of the number of the sampled solutions. As a result, the most greedy neighbor generation of TS approach works better in LCSCP. However, the quality of the solution returned by each approach is still very close.

Additionally, scale of the problem makes little influence on each approach. For the problem itself, a bigger test size means more feasible solutions and better continuity of the search space. In this way, when computational resources are adequate, qualities of the best solutions returned will not get worse as the test size increases. On the other hand, with a limit of the number of sampled solutions, a bigger test size means more infeasible solutions, which results to a bigger risk of spending computational resource on not promising search area. However, all of three approaches still can ensure the quality of the solutions not fall as the test size increases. In conclusion, all of them are of good scalability on LCSCP. Moreover, on different workflows, performance of each approach is stable, which indicates these three approaches may be effective on more complicated workflows.

VI. Conclusion

In this paper, the optimization formulation of long-term based cloud service composition problem, has been proposed as LCSCP. It is novel in the sense that it (i) fills the gap between the academic research and the industry situation; (ii) contains an integrated QoS model which takes both application services and utility services into consideration. Based on this formulation, specialized versions of three meta-heuristic approaches are proposed and elaborated. Comprehensive experiments are designed and conducted to get a basic understanding of LCSCP.

Our future work will explore LCSCP in a real-world case in which the correlation among QoS parameters should be considered and it would be solved online. Additionally, We will also study the unified way to evaluate the cloud service composition approaches in both functional and nonfunctional aspects.

VII. Acknowledgment

This work was supported in part by the 973 Program of China under Grant 2011CB707006, the National Natural Science Foundation of China under Grants 61175065 and 61329302, the Program for New Century Excellent Talents in University under Grant NCET-12-0512, and the European Union Seventh Framework Programme under Grant 247619.

References

[1] H. R. Motahari-Nezhad, B. Stephenson, and S. Singhal, "Outsourcing business to cloud computing services: Opportunities and challenges," *IEEE Internet Computing*, vol. 10, 2009.

[2] D. M. Smith, "Hype cycle for cloud computing, 2011," *Gartner Inc., Stamford*, 2011.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] T. J. Bittman. (2014, October) Hybrid Clouds and Hybrid IT: The Next Frontier. Gartner Inc. [Online]. Available: http://my.gartner.com/

[5] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications*, vol. 45, pp. 134–150, 2014.

[6] T. Chen, R. Bahsoon, and X. Yao, "Online qos modeling in the cloud: A hybrid and adaptive multi-learners approach," in *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*. IEEE, 2014, pp. 327–336.

[7] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, 2004.

[8] M. C. Jaeger, G. Muhl, and S. Golze, "QoS-aware composition of Web services: a look at selection algorithms," in *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, 2005.

[9] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition based on genetic algorithms," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 1069–1075.

[10] P. R. Lewis, F. Faniyi, R. Bahsoon, and X. Yao, *Markets and clouds: Adaptive and resilient computational resource allocation inspired by economics*. Taylor & Francis, 2013.

[11] Y. V. Natis, "Gartner aPaaS Report Card: Choose Your Cloud Application Platform Wisely," Gartner Inc., Tech. Rep., February 2012. [Online]. Available: https://www.gartner.com/doc/1930015/gartner-apaas-report-card-choose

[12] Z. Ye, A. Bouguettaya, and X. Zhou, "QoS-aware cloud service composition based on economic models," in *Service-Oriented Computing*. Springer, 2012, pp. 111–126.

[13] ——, "QoS-aware cloud service composition using time series," in *Service-Oriented Computing*. Springer, 2013, pp. 9–22.

[14] M. Berkelaar, K. Eikland, P. Notebaert *et al.*, "lpsolve: Open source (mixed-integer) linear programming system," *Eindhoven U. of Technology*, 2004.

[15] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "A framework for QoS-aware binding and re-binding of composite web services," *Journal of Systems and Software*, vol. 81, no. 10, pp. 1754–1769, 2008.

[16] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[17] R. A. Leo, "Monitoring Tools For Actionable Intelligence," University of Houston Clear Lake, Supremus Group, Tech. Rep., July 2013. [Online]. Available: https://www.sentinelagent.com/

[18] A. Klein, F. Ishikawa, and S. Honiden, "SanGA: A self-adaptive network-aware approach to service composition," *IEEE Transactions on Services Computing*, no. 3, pp. 452–464, 2014.

[19] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[20] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for qos-aware web service composition," in *Web Services, 2006. ICWS'06. International Conference on*. IEEE, 2006, pp. 72–82.

[21] F. Rosenberg, M. Muller, P. Leitner, A. Michlmayr, A. Bouguettaya, and S. Dustdar, "Metaheuristic optimization of large-scale qos-aware service compositions," in *Services Computing (SCC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 97–104.

[22] P. J. Van Laarhoven and E. H. Aarts, *Simulated annealing*. Springer, 1987.