An New Ant-Based Approach for Optimal Service Selection with E2E QoS Constraints

Dac-Nhuong Le
Hanoi University of Science, Vietnam National University
Vietnam
Nhuongld@hus.edu.vn

Abstract: *In this paper, we study the dynamic service composition becomes a decision problem on which component services should be selected under the E2E QoS constraints. This problem is modeled is a nonlinear optimization problem under several constraints. We have two approaches: local selection is an event strategy with polynomial time complexity but cannot handle global constraints and the traditional global selection approach based on integer linear programming can handle global constraints, but its poor performance renders it inappropriate for practical applications with dynamic requirements. To overcome this disadvantage, we proposed a novel Min-Max Ant System algorithm with the utilites heuristic information to search the global approximate optimal. The experiment results show that our algorithm is efficiency and feasibility more than the recently proposed related algorithms.*

Keywords: E2E QoS constraints, QoS-aware, Min-Max Ant System

Received: 4 August 2014, Revised 9 September 2014, Accepted 12 September 2014

1. Introduction

Nowadays, the web services technologies are new solution to develope software foundation for next generation enterprise and Web-based systems. Services from different providers can be integrated into a composite service regardless of their locations, platforms, and/or execution speeds to implement complex business processes and transactions that can be provide a promising solution for the integration of business application [1]. With growing demands, many service providers have started to oer dierent QoS (Quality of service) service levels so as to meet the needs of different user groups, by offering different service qualities based on user needs or service cost [2]. For composite services, one of the QoS issues is to dene the service integration model and identify the optimal service selection to meet a users QoS requirement. Component services can be dynamically composed to form complex and value-added services so that the requirement of users be satisfised. Therefore, the end-to-end (E2E) services selection is an important part of the web service composition problem [3].

In this paper, we study the dynamic service composition becomes a decision problem on which component services should be selected under the E2E QoS constraints. This problem is modeled is a decision problem as a nonlinear optimization problem under several constraints. The rest of this paper is organized as follows: some related works will be analyzed and evaluated in section 2. Section 3 introduces the model system defined for service selection with E2E QoS constraints problem. Section 4 resents our new an ant-based approach to solve it. Section 5 is our simulation and analysis results based on performance compared, and finally, section 6 concludes the paper.

2. Related works

There are many have worked on this topic, such as: BPEL4WS (Business Process Execution Language for Web services), Microsofts XLANG, IBMs WSFL (Web service Flow Language) , WSCI (Web service Choreography Interface), BPML (Business Process Modeling Language), DAML-S (DARPA Agent Markup Language) are the industrial standard specifications for Web service composition. The SWORD project, FUSION framework, eFlow system, QoS-oriented framework WebQ, SELF-SERV platform are the centralized operator, engine, coordinator to control the execution of constructed composite services [1]. An quality driven approach to select component services during execution of a composite service presented in [4], the authors consider multiple QoS criteria such as price, duration, reliability and take into account of global constraints. The linear programming method is used to solve the service selection problem, which is usually too complex for run-time decisions. The E2E QoS model and management architecture for complex distributed real-time system [5].

The QoS-based service selection problem aims at finding the best combination of web services that satisfy a set of end-to end QoS constraints in order to fullfil a given SLA (Service Level Agreement), which is an NP-hard problem [6]. This problem becomes especially important and challenging as the number of functionally equivalent services offered on the web at different QoS levels increases exponentially [7]. As the number of possible combinations can be very huge, based on the number of subtasks comprising the composite process and the number of alternative services for each subtask. The local selection strategy in which, the candidate service who has the maximum QoS utility value is selected for each service class is very ecient but can not handle end-to-end QoS constraints [9]. D. Ardagna et al proposed exact search algorithms to perform an exhaustive search to nd the best combination that satifises a certain composition level SLA is impractical [8]. T.Yu et al proposed heuristic algorithms that can be used to find a near-optimal solution efficiently in [3]. M.Wu et al proposed the global optimization algorithm uses integer linear programming techniques to nd the optimal selection of component services for the composition [10]. However, the disadvantages of deterministic algorithms like linear programming are only applicable to small-scale problems due to their exponential time complexity.

In [11], C.W.Zhang et al are proposed a Genetic Algorithm (DiGA) to solve the web service selection supporting QoS. Similiar, S. Liu et al in [12] are applied genetic algorithm for dynamic web services selection algorithm with QoS global optimal in web services composition. The advantages of genetic algorithm are intelligence and global convergence. Genetic algorithm which is inspired from the evolution of biological populations is a non-deterministic algorithm. Although very efficient in solving NP hard problem like service selection, the eect of genetic algorithm is very sensitive to its parameter setting. The Parameter-Adaptive Genetic Algorithm (PAGA) solved the service selection problem presented in [13]. In [14], K.Su et al suggested an ecient discrete invasive weed optimization algorithm for web services selection consists two steps. First, a set of randomly generated feasible solutions are transformed into decimal code. Second, they utilize Gaussian diffusion to guide the population tospread in the solution space. X.Q. Fan et al is presented a new cooperative evolution algorithm consisting of a Stochastic Particle Swarm Optimization (SPSO) [15] and a Simulated Annealing (SA). The Multipheromone and Dynamically Updating Ant Colony Optimization algorithm (MPDACO) put forward to solve this problem, which includes a global optimization process and a local optimizing process proposed by Y.M.Xia et al [16]. C.Zhange et al proposed the Clustering Based Ant Colony Algorithm for Service Selection (CASS) including its model and concrete algorithm in [17].

3. Service selection with E2E QoS constraints Problem

3.1 Notations and denitions

The concepts denition for service selection with E2E QoS constraints problem [18] can be dened as as follows:

1. **Service Class**: Service class $S_j = \{s_{1j}, s_{2j} \ldots\ldots s_{mj}\}$ is used to describe a set of functionality equivalent web services, where $s_{ij}(1 \le i \le m)$ represents the i^{th} component service in service class S_j .

2. **Abstract Composite Service:** An abstract composite service $CSabstract = \{S_1, S_1, \ldots\ldots S_n\}$ can be dened as an abstract representation of a composition request. Each task of the abstract composite service refers to a service class without referring to any concrete service.

3. **Concrete composite Service:** A concrete composite service *CS* is a instantiation of an abstract composite service. It can be gained by binding each service class $S_j \in C\ Sabstract$ to a concrete service s_{ij}.

4. **QoS Criteria:** which can be used to describe the quality of web services, is a set of non-functional properties. QoS criteria can be devided into several aspects such as runtime-related QoS, transaction support related QoS, configuration management

QoS, cost related QoS and security related QoS [19]. More generally, QoS criteria can include generic QoS like response time, price, reliability, availability etc, as well as domain specific QoS [18]

An extensible QoS computation model includes both the generic and domain specific criteria, where new criteria can be added and used to evaluate the QoS without changing the underlying model are presented in [18]. The QoS criteria of a concrete service s can be dened as a vector $Q(s) = [\, q_1(s)\,, q_2(s) \ldots\ldots\ldots\ldots\ldots q_r(s)\,]$ where $q_k(s)$ $(1 \le k \le r)$ represents the k^{th} QoS criterion of s. The QoS criteria of a concrete composite service *CS* can be defined as a vector $Q\,(CS) = [q'_1(CS), q'_2(CS), \ldots\ldots q'_k\,(CS)]$, where $q'_k\,(CS)$ $(1 \le k \quad r)$ represents the k^{th} QoS criterion of *CS* [18].

3.2 Web Service QoS Attributes

The QoS attributes of a business process are decided by the QoS attributes of individual services and their composition relationships. There are different ways that individual services can be integrated to form a business process. The QoS of a concrete composite service is decided by the QoS of its component services and the composition model. The four basic relationships are sequential, parallel, conditional and loop. Our composition model that we focus in this paper is sequential, due to the other models can be transformed to the sequential model by using some technologies like Critical Path Algorithm, Hot Path, and Loop Unfolding [4].

In this study, we are considered four quality attributes as part of the Web service parameters: response time, reliability, availability and throughput. The response time of the business processs is the sum each individual service s_j response time at the chosen service level. The reliability of the business processsis the product of each individual service s_j reliability at the selected service level. It is a non-linear function. In order to make all aggregation functions to be linear, we can transform it using an logarithmic function. The availability of the business processsis the product of each individual service sj availability at the selected service level. Same as the reliability attribute, we can use an logarithmic function to convert it to a linear formulation. The computation expression of the QoS of composite services are shown in Table 1.

QoS Criteria	**Aggregation Function**
Response Time	$q'_{time}\,(CS) = \sum_{j=1}^{n} q_{time}\,(s_j)$
Reliability	$q'_{rel}\,(CS) = \prod_{j=1}^{n} q_{rel}\,(s_j)$
Availability	$q'_{av}\,(CS) = \prod_{j=1}^{n} q_{av}\,(s_j)$
Throughput	$q'_{thr}\,(CS) = \min\limits_{1 \le j \le n} q_{thr}\,(s_j)$

Table 1. Computation expression of the QoS of composite services

Criteria like reliability, availability and throughput are positive that means the higher the value is, the higher the quality is. On the contrary, response time is a negative criterion that means the higher the value is, the lower the quality is. We utilize the Simple Addtive Weighting [4] technique to map the QoS vector into a single real value for the purpose of evaluating the quality of composite services. The utility of composite service CS is computed as:

$$U\,(CS) = \sum_{k=1}^{1} w_k \; \frac{Q_{max}\,(k) - q'k\,(CS)}{Q_{max}\,(k) - Q_{min}\,(k)} \; + \sum_{k=2}^{4} w_k \; \frac{q'k\,(CS) - Q_{min}\,(k)}{Q_{max}\,(k) - Q_{min}\,(k)} \qquad (1)$$

In which, $Q_{min}(k)$ and $Q_{max}(k)$ represent the minimum and maximum value of the k^{th} *QoS* criterion of all possible instantiations of the given abstract composite service can be computed as follows :

$$Q_{min}\,(k) = \sum_{j=1}^{n} \max_{s_{ij} \in S} q_k\,(s_{ij}) \qquad (2)$$

$$Qmax\,(k) = \sum_{j=1}^{n} \max_{s_{ij} \in S} q_k\,(sij) \qquad (3)$$

and w_k $(1 \le k \le 4)$ is the weight assigned to each QoS criterion, which represents users priorities. We have the following constraints on w_k.

$$\sum_{k=1}^{4} w_k = 1, 0 \le w_k \le 1, w_k \in R \tag{4}$$

3.3 QoS-aware service composition

The process of QoS-aware service composition can be dened as follows:

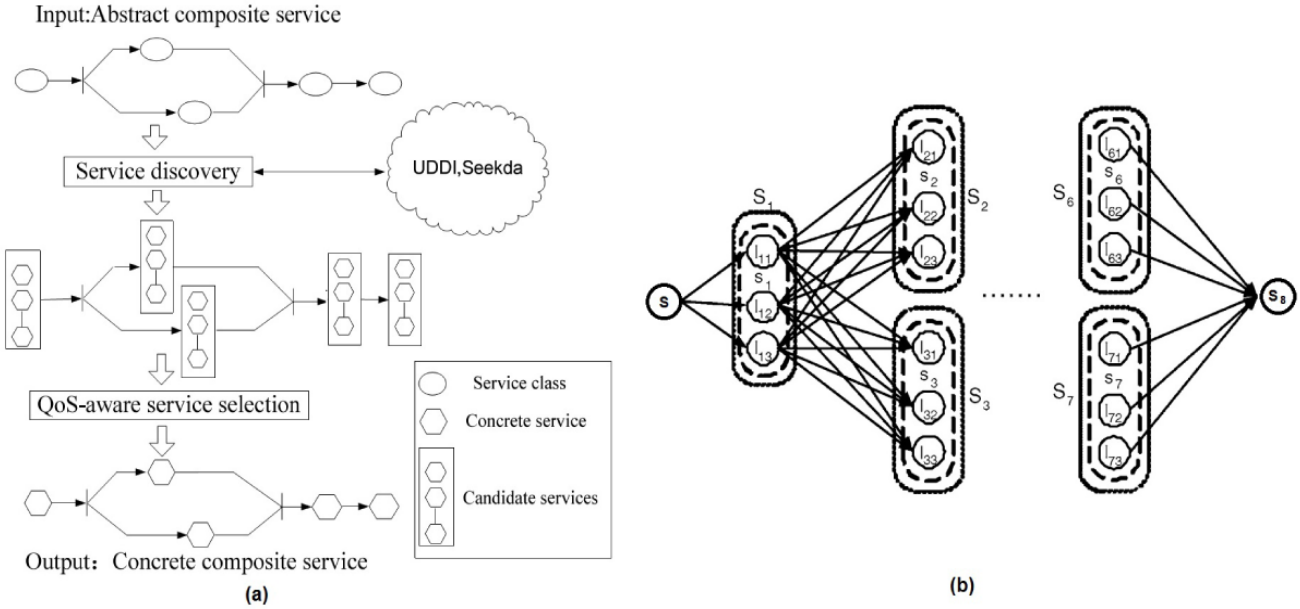

Figure 1. QoS-aware service composition and Graph construction

1. An abstract composite service can be stated in a workflow-like language like *Process Execution Language for Web Services* (BPEL4WS).

2. The discovery engine utilizes existing infrastructure like Universal Description Discovery and Integration (UDDI) or Seekda [20] to discover several concrete services for each task in the workflow by means of syntactic or semantic matching.

3. The service selection middleware selects one component service for each task. The component services selected should maximize the utility of composite service without violating the E2E QoS constraints

We use binary decision variables x_{ij} to represent whether the candidate service s_{ij} is selected or not (x_{ij} = 1 means s_{ij} is selected, and otherwise). We can re-writing formula (1) to include the decision variables and the *QoS* computation expression of composite services defined in Table 1.

The problem of QoS-aware service selection can be formulated as a maximization problem of the utility value [16] given by:

$$F = max\left(w_1 \frac{Q_{max}(1) - \sum_{j=1}^{n}\sum_{i=1}^{m} q1(s_{ij})\, x_{ij}}{Q_{max}(1) - Q_{min}(1)} + \sum_{k=2}^{3} w_k \frac{\prod_{j=1}^{n}\sum_{i=1}^{m} q_k(s_{ij})\, x_{ij}}{Q_{max}(k) - Q_{min}(k)} Qmin(k) + w_4 \frac{\min_{j}\left(\sum_{i=1}^{m} q4(s_{ij})\, x_{ij}\right) - Q_{min}(4)}{Q_{max}(4) - Q_{min}(4)} \right)$$

Subject to:

$$\sum_{i=1}^{m} x_{ij} = 1, 1 \le k \le n \tag{6}$$

$$\sum_{j=1}^{n}\sum_{i=1}^{m} q1\,(s_{ij})\,x_{ij} \le C_1 \qquad (7)$$

$$\prod_{j=1}^{n}\sum_{i=1}^{m} q_k\,(s_{ij})\,x_{ij} \le C_k, k=2,3 \qquad (8)$$

$$\min_{j}\left(\sum_{i=1}^{m} q_4\,(s_{ij})\,x_{ij}\right) \le C_4 \qquad (9)$$

Where C_1, C_2, C_3, C_4 represent the end-to-end constraints on response time, reliability, availability and throughput respectively. (6) means that only one concrete service should be selected for each task. From the above discussion, we can nd that QoS-aware service selection problem is a nonlinear optimization problem under several constraints. Our approach for this problem is presented in the next section. QoS-aware service selection problem is proved to be a NP- Complete problem. Deterministic algorithms like linear programming are only applicable to small-scale problems due to their exponential time complexity.

4. Our approach

The *Min-Max Ant System* (MMAS) based on several modifications to *Ant System* (AS) which *aim* (*i*) to exploit more strongly the best solutions found during the search and to direct the ants search towards very high quality solutions and (ii) to avoid premature convergence of the ants' search [21].

4.1 Construst Ant solutions

In the rst step, we generated the construct graph for services. Each service level in every individual service is represented by a node in the graph, with a cost and a benefit value in Figure 1.

1. Each service level of each individual service is represented as a node in the graph.

2. If service s_i is connected to service s_j, all service levels in s_i are connected to all service levels in s_j.

3. Set link cost, delay and benefit: To remove parameters from graph nodes, we add the service time and cost of the receiving node to all incoming links to the node. That is, if data is sent from s_i to s_j, we will add service time $e\,(s_j\,;\,l\,)$ and cost $c\,(s_j\,,\,l\,)$ to the link connected to s_j. So for the link from service level l_a of s_i to service level l_b of s_j, its delay is set to $d_{ij} + e\,(s_j\,,\,l_b)$ and its cost is $c_{ij} + c\,(s_j\,,\,l_b)$. The benet of the link is set to $b\,(s_j,\,l_b)$.

4. Suppose there are *k* service classes in the execution plan, we add a source node (S_0) and a sink node ($S_k + 1$) to the graph as shown in Figure1. For all nodes that have no incoming edges, add links from the source node to them; the delay, cost and benefit of these links are set to 0. For all vertexes that have no outgoing edges, connect them with the sink node. The delay and cost of these links are set to be the transmission delay and cost between the service (that provides the start node of the link) and user. The benefit of these links are set to 0.

5. For each link, according to formula (5), we can compute its utility *F* based on the benefit and cost of the link.

4.2 Update pheromones

Let τ_{min} and τ_{max} are lower and upper bounds on pheromone values. We initialize all pheromone trails to τmax and choose rst service s_i has been randomly chosen. For each candidate service s_j, the pheromone factor $\tau_{Sk}\,(s_j)$ is initialized to $\tau(s_{i,}\,s_j)$. Each time a new service s_l is added to the solution S_k, for each candidate service s_j, the pheromone factor $\tau S_k\,(s_j)$ incremented by $\tau(s_l,\,s_j)$. And the highest pheromone factor can be dened as

$$\tau sk\,(s_{ij}) = \sum_{s_j \in s_k} \tau\,(s_i\,s_j) \qquad (10)$$

When the algorithm starts, ants can perform search in order of group that impose the lowest restriction on the next choices. The key point is to decide which components of the constructed solutions should be rewarded, and how to exploit these

rewards when constructing new solutions. A solution is a set of selected objects $S = \{ s_{ij} \mid x_{ij} = 1\}$. Say $x_{ij} = 1$ is the mean service s_{ij} is selected and the corresponding decision variable x_{ij} has been set to 1. We must constructed solutions $S = \{x_{i1j1}$ ..... x_{injn}, pheromone trails are laid on each objects selected in *S*. So pheromone trail τ_{ij} will be associated with object .

Let S_k be the set of the selected objects at the k^{th} iteration. The heuristic factor $S_k(s_{ij}) = q_k(s_{ij})\, x_{ij}$. We have unique idea of a separate pheromone trail for services to save the ordering of services that lead to a good solution. The service pheromone trail also follow a MMAS approach and initialized to the max pheromone value. Once each ant has constructed a solution, pheromone trails laying on the solution objects are updated according to the ACO [22]. First, all amounts are decreased in order to simulate evaporation. This is done by multiplying the quantity of pheromone laying on each object by a pheromone persistence rate $(1- P)$ such that $0 \le P \le 1$. Then, pheromone is increased for all the objects in the best solution of the iteration. When constructing a solution, an ant starts with an empty solution. At the first construction step an ant selects a group randomly and at all the latter steps, groups are selected according to their associated pheromone value. After selecting a group, the algorithm removes all the bad Candidates that violates resource constraints. The local heuristic information of the remaining candidate objects of the services and selects an object updated following probability equation:

$$Ps_k(s_{ij}) = \frac{[\tau s_k(s_{ij})]^{\alpha}\,[nS_k(s_{ij})]^{\beta}}{\sum_{sij \in Candidates} [\tau S_k(s_{ij})]^{\alpha}\,[\eta S_k(s_{ij})]^{\beta}} \qquad (11)$$

in which, *Candidates* are all items from the currently selected service which do not violate any resource constraints; τS_k is the pheromone factor of the dynamic heuristic information S_k; ηS_k is the desirability of s_{ij} .

$$\eta S_k(s_{ij}) = \frac{1}{q_k(s_{ij}) x_{ij}} \qquad (12)$$

The influence of the pheromone concentration to the probability value is presented by the constant α, while constant β do the same for the desirability and control the relative importance of pheromone trail versus local heuristic value.

$$\tau S_k(s_{ij}) \leftarrow (1-\rho)\,\tau s_k(s_{ij}) + \Delta\tau S_k(s_{ij})^{best} \qquad (13)$$

Let $S_{internalbest}$ be the best solution constructed during the current cycle. The quantity of pheromone increased for each object is defined by

$$G(S_{internalbest}) = \sum_{sij \in\ Sinternalbests} \frac{1}{\sum_{j=1}^{n}\sum_{i=1}^{m} q(s_{ij})\, x_{ij}} \qquad (14)$$

After each iteration, service pheromone values have much better than the original pheromone trail for ants. The best solution updates the group pheromone trail. All the adjacent groups get the highest amount of pheromone value that gradually diminishes as the distance between services increases. The Candidate-groups data structure maintains a list of feasible candidate groups which can be considered next. After each ant has constructed a solution, the best solution of that iteration is identified and a random local search procedure and a random item swap procedure is applied to improve it. Then pheromone trail is updated according to the best solution. Also it maintains a database of top *k* solutions. After each iteration a small amount of pheromone is deposited in the pheromone trails of the objects belonging to the top *k* solutions. The motivation behind this strategy is to ensure quick convergence on good solutions and to explore better areas more thoroughly. The algorithm stops either when an ant has found an optimal solution, or when a maximum number of cycles has been performed. After pheromone update, $\tau S_k(s_{ij})$ is set in $[\tau_{min}, \tau_{max}]$; $\forall s_{ij}$ defined by:

$$\tau S_k(s_{ij}) = \begin{cases} \tau_{max}\ if\ \tau S_k(s_{ij}) > \tau_{max} \\ \tau_{max}\ if\ \tau S_k(s_{ij}) \in [\tau_{min}, \tau_{max}] \\ \tau_{min}\ if\ \tau S_k(s_{ij}) < \tau_{min} \end{cases} \qquad (15)$$

Algorithm 1 MMAS algortihm for Service Selection with E2E QoS Constraints

Initialize pheromone trails to τ_{max} for both item and group;

$Top_{ksolution} \Leftarrow$ {hold topmost k solutions};

Repeat

$S_{globalbest} \Leftarrow \oslash$;

For each *ant* $k = 1:::N_{Ant}$ **do**

$S_{internalbest} \Leftarrow \oslash$;

Candidategroups $\Leftarrow$ all the groups;

While Candidategroups $\neq \oslash$ **do**

$C_g \Leftarrow$ Select a group from Candidategroups according to group pheromone trail;

Candidates $\Leftarrow$ $\{s_{ij} \in C_g$ that do not violate resource constraints in (6)-(9);

Update local heuristic values by equation (13);

Choose an object $s_{ij} \in$ Candidates with probability computed by equation (11);

$S_k \Leftarrow S_k \cup s_{ij}$; Remove C_g from Candidategroups;

Endwhile

If $F(S_k) > F(S_{internalbest})$ **then** $S_{internalbest} \Leftarrow S_k$;

Endfor

If $F(S_{globalbest}) < F(S_{internalbest})$ **then** $S_{globalbest} \Leftarrow S_{internalbest}$;

Update database and pheromone trails lower and upper bounds by τ_{min} and τ_{max};

Until maximum number of cycles reached (N_{max}) or optimal solution found;

The comparison time complexity of our algorithm with the recently proposed related algorithms are shown in Table 2.

Algorithm	Time complexity
Integer linear programming	$O(2^{MN})$
Exhaustive search	$O(M^N)$
Monte Carlo	M^q (where q is an integer smaller than N)
DiGA	$O\,(iter_{max} \bullet P_{max} \bullet W_{max} \bullet (N + log\,(P_{max} \bullet W_{max}))$
PAGA	$O\,(iter_{max} \bullet N \bullet L + iter_{max} \bullet L \bullet logL)$
MMAS	$O\,(N_{ant} \bullet N_{max} \bullet N \bullet M \bullet k)$

Table 2. The comparison time complexity of the dierence algorithms considered

5. Experiments and Results

5.1 Dataset and parameter implementation

There are three factors that determine the size of the QoS-aware service selection problem: the number of tasks in composite service N, the number of candidate services per class M and the number of QoS constraints k ($1 \leq k \leq 4$). In the our experiment, we present an experimental evaluation of our algorithm, focusing on its efficiency in terms of execution time and the quality in terms of the obtained best solution fitness value, and compare it with the recently proposed related algorithms Exhaustive search [8], Monte Carlo, DiGA [11], PAGA [13], SPSO [15], MPDACO [16],and CASS [17] on different scale test instances. We conducted experiments using the QWS real dataset which includes measurements of 9 QoS attributes for 2507 real web services [23]. These services were collected from public sources on the web, including UDDI registries, search engines, and service portals, and their QoS values were measured using commercial benchmark tools [7]. The different scale test instances are created and shown as in Table 3.

Intances	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15
N	5	5	5	5	5	5	5	5	5	5	10	10	10	10	10
M	4	8	12	16	20	24	28	32	36	40	50	100	150	200	250
w_k	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25

Table 3. Main features of test instances

All algorithms are implemented in C language and executed on a Chipset DuoCore 3.0GHZ, 4GB RAM computer. The colony size of our algorithm is set as $N_{Ants} = 50$, and the other parameters are set as follows: $k = 10$, $\alpha = 1$, $\beta = 5$, $P = 0{:}01$, $\tau_{min} = 0.01$, $\tau_{max} = 8$, $\rho = 0.5$, $N_{Max} = 500$. The parameters of the other three compared algorithms are set as the same as the original papers [8], [11], [13], [15], [16], [17].

5.2 Case study 1

In the first experiment, we implemented and evaluated the execution time and the quality in terms of the obtained best solution tness value of the our algorithm and compare it with Exhaustive search [8], Monte Carlo, DiGA [11], PAGA [13], SPSO, MPDACO, CASS algorithms on instances #1 to #10. In Figure 2(a) shows the optimality of algorithms with different number of candidate services change from 4 to 40. The ratio between the optimal QoS utility value obtained by the given approach and the optimal QoS utility value obtained by Exhaustive search approach are shown in Figure 2(b) .

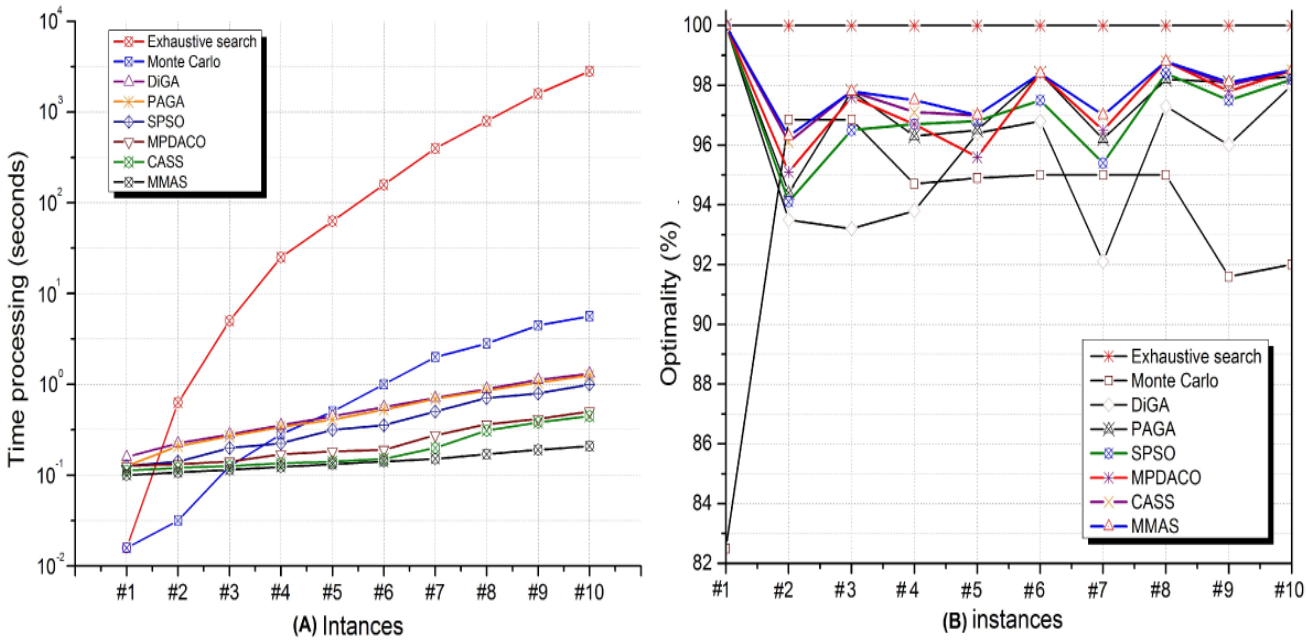

Figure 2. Comparison time processing and optimality of algorithms considered

The results show that our algorithm is able to achieve above 98% optimality, which outperforms other approaches in Figure 2(a). Therefore, from Figure 2(b), we can s that MMAS algorithm is faster than the other three compared algorithms for the test instances.

5.3 Case study 2

In the second experiment, we run each DiGA, PAGA, SPSO, MPDACO,CASS and MMAS algorithms twenty times on each test instance #11 to #15. The termination condition for all the algorithms on each test instance is set based on the maximum candidate evaluation number, which is set as 3 104. The obtained best, worst, and average fitness values are given in Table 4 belows. In which, the optimal value in the best, worst, and average case is shown in bold. The experimental results show that the MMAS algorithm achieved good quality and stable solution than the previous approach.

	DiGA			PAGA			SPSO		
Instance	*Best*	*Worst*	*Average*	*Best*	*Worst*	*Average*	*Best*	*Worst*	*Average*
#11	**0.1219**	0.1239	0.1231	**0.1219**	0.1256	0.1229	**0.1219**	0.1241	0.1228
#12	0.1321	0.1365	0.1345	0.1319	0.1351	0.1342	0.1316	0.1347	0.1339
#13	**0.1455**	0.1614	0.1555	0.1459	0.1582	0.1548	0.1468	0.1573	0.1536
#14	0.1595	0.1798	0.1677	0.1563	0.1611	0.1594	0.1546	0.1561	0.1549
#15	0.1474	0.1637	0.1562	0.1427	0.1532	0.1482	0.1469	0.1519	0.1476
	MPDACO			**CASS**			**MMAS**		
Instance	*Best*	*Worst*	*Average*	*Best*	*Worst*	*Average*	*Best*	*Worst*	*Average*
#11	**0.1219**	0.1237	0.1227	**0.1219**	0.1234	0.1225	**0.1219**	**0.1231**	**0.1223**
#12	0.1316	0.1342	0.1335	**0.1312**	**0.1337**	0.1332	**0.1312**	**0.1337**	**0.1331**
#13	**0.1455**	0.1501	0.1481	**0.1455**	0.1529	0.1481	**0.1455**	**0.1497**	**0.1473**
#14	**0.1521**	0.1562	0.1539	**0.1521**	0.1559	0.1533	**0.1521**	**0.1559**	**0.1531**
#15	0.1426	0.1503	0.1464	**0.1415**	**0.147**	0.1447	**0.1415**	**0.147**	**0.1446**

Table 4. Comparison of the results obtained by the dierence algorithms considered

6. Conclusion and Future Works

In this paper, we study the E2E QoS constraint issue for composite business processes that are built using the Web service framework. We have presented and analysised service selection algorithms. The objective of the algorithms is to maximize user-defined service utilities while meeting the E2E performance constraint. We propose a MMAS algorithm to solve the problem. We evaluated our algorithm, focusing on its efficiency in terms of execution time and the quality in terms of the obtained best solution tness value, and compare it with the recently proposed related algorithms Exhaustive search [8], Monte Carlo, DiGA [11], PAGA [13], SPSO [15], MPDACO [16],and CASS [17] on different scale test instances. The computational results showed that my approach is currently among the best performing algorithms for this problem. Service selection algorithms under dynamic QoS dataset is our next research goal.

References

1. Yu, T., Lin, K. J. (2005). Service selection algorithms for Web services with end-to-end QoS constraints, *In*: Proceeding of ISeB, Springer, p.103-126.

2. Ma, X., Dong, B. (2009). AND/OR Tree-based Approach for Web Service Composition, *Journal of Computational Information Systems* 5 (2) 727-728.

3. Yu, T., Zhang, Y., Lin, K. J. (2007). Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints, ACM Trans on the Web 1: p.1-26.

4. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z. (2003). Quality Driven Web Service Composition. *In:* Proc. 12th International WWW Conference.

5. Shankar, M., DeMiguel, M., Liu, J. W. S. (1999). An end-to-end QoS management architecture, *In:* Proceeding of 5th Real-Time Technology and Applications Symposium.

6. Michlmayr, A. et al. (2010). E2E support for QoS-aware service selection, binding, and mediation in VRESCo, *IEEE Trans on Services Computing*, 3 (3) 193-205.

7. Masri, E.A.et al. (2008). Investigating web services on the world wide web, *In*: Proceedings of the International Conference on World Wide Web, p.795-804, China, 2008.

8. Ardagna,D., Pernici, B. (2007).Adaptive service composition in exible processes, *IEEE Transon Software Engineering*, 33 (6) 369-384.

9. Zeng, L., Benatallah, B. (2004). QoS-Aware Middleware for Web Services Composition, *IEEE Transactions on software engineering* 30 (5) 321-322.

10. Wu, M. et al. (2011). Qos-driven Global Optimization Approach for Large-scale Web Services Composition, *Journal of Computers*, 6 (7) 1452-1456.

11. Zhang, C. et al. (2007). DiGA: population diversity handling genetic algorithm for QoS-aware web services selection, Computer Comm, 30 (5) 082-1090.

12. Liu, S.L. et al. (2007). A Dynamic Web Services Selection Algorithm with QoS Global Optimal in Web Services Composition, *Journal of Software*, 18 (3) 651-655.

13. Su, K., Liangli, M., Xiaoming, G., Yufei, S. (2014). An Efficient Parameter-adaptive Genetic Algorithm for Service Selection with End-to-end QoS Constraints, *Journal of Computational Information Systems* 10 (2) 581-588 .

14. Su,K. et al. (2014). An Ecient Discrete Invasive Weed Optimization Algorithm for Web Services Selection, Journal of software, 9 (3) 709-715.

15. Fan,X.Q. et al. (2011). Research on Web service selection based on cooperative evolution, Expert Systems with Applications, 38 (8) 97369743.

16. Xia,Y.M. et al. (2012). Optimizing services composition based on improved ant colony algorithm, *Chinese Journal of Computers*, 35 (2) 270281.

17.Zhang, C., Yin, H., Zhang, B. (2013). A Novel Ant Colony Optimization Algorithm for Large Scale QoS-Based Service Selection Problem, Discrete Dynamics in Nature and Society Vol.2013, p.1-9.

18. Liu, Y., Ngu, A.H.H., Zeng, L. (2004). QoS Computation and Policing in Dynamic Web Service Selection, In Proceedings of the International World Wide Web Conference, p.66-73.

19. Ran, S. (2003). A Model for Web Services Discovery With QoS, ACM, p.6-9.

20. Seekda. http://webservices.seekda.com/

21. Stutzle et al, (2000). Max-min ant system. Future Generation Computer Systems, p.889-914, 2000.

22. Dac-Nhuong Le. Evaluation of Pheromone Update in Min-Max Ant System Algorithm to Optimizing QoS for Multimedia Services in NGNs, Advances in Intelligent Systems and Computing, Vol.338, pp.9-17, Springer 2014.

23. Al-Masri, E., Mahmoud, QH, The qws dataset, http://www.uoguelph.ca/ qmahmoud/qws/index.html

24. http://randdataset.projects.postgresql.org/