QoS-Aware Peer Services Selection Using Ant Colony Optimisation

Jun Shen and Shuai Yuan

School of Information Systems and Technology
University of Wollongong, Australia
Wollongong 2522, Australia
jshen@uow.edu.au, shuai.yuan@hotmail.com

Abstract. Web services coordinated by computational peers can be aggregated to create composite workflows that provide streamlined functionality for human users or other systems. One of the most critical challenges introduced by Peer-to-Peer (P2P) based Web services is represented by Quality of Service (QoS)-driven services composition. Since many available Peers provide overlapping or identical functionalities, though with different QoS, selections need to be quickly made to determine which peers are suitable to participate in an expected composite service. The main contribution of this paper is a heuristic approach which effectively and adaptively finds appropriate service peers for a service workflow composition, and also some uncertainties in the real ad-hoc scenarios are considered by a proper re-planning scheme. We propose to adopt Ant Colony Optimisation (ACO) to tackle the QoS-aware Peers' composition problem in both static and dynamic situations, as ACO represents a more scalable choice, and is suitable to handle and balance generic QoS attributes by pheromones. The proposed approach is able to improve the selection performances in various service composition structures, and also can adaptively handle unexpected events. We present experimental results to illustrate the efficiency and feasibility of the proposed method.

Keywords: P2P, QoS, ACO, service selection, composition.

1 Introduction

Web services are autonomous software systems identified by URIs which can be advertised, located, and accessed through messages encoded according to XML-based standards (e.g., SOAP, WSDL, and UDDI [6]) and transmitted using Internet protocols [18]. In decentralised network, Peer-to-Peer [19] (P2P) based service computing inherits the foundational service-oriented features (e.g. service protocols, service discovery mechanism and QoS awareness, etc.), and therefore becomes more flexible and scalable, due to the capability of fully-distributed computing. Widely accepted and standardised Service-Oriented Architecture (SOA) makes it possible to realise larger scale computing paradigm like SaaS, so that P2P-based service systems, which depend upon peers' cooperation and share on services resources, can bring profound benefits and profits for decentralised service network (e.g. mobile commerce application and Location Based Services [14]). In fully-distributed P2P based service

W. Abramowicz and D. Flejter (Eds.): BIS 2009 Workshop, LNBIP 37, pp. 362–374, 2009.

systems, service composition is very inherent to occur as service peers are expected to work together, normally represented by a business process or scientific or engineering workflow, through cooperation/coordination to achieve desirable global goals. JXTA and BPEL [19] also allow peers to cooperate and automate business processes and reengineer their workflow structure, so as to rationally compose and make use of all resources in decentralised environment. In addition, P2P-based composite application is able to increase efficiency and reduce costs, as they are highly reusable and easily restructurable. The fast composition is essentially required to replan a service composition during the execution, sometimes because the actual QoS deviates from the estimated schedule and this may cause constraint violation, or sometimes even simply because some services might not be available on the fly. In this case, the re-composition time will influence the overall service response time, thus it should be kept as minimum as possible.

The scalability of decentralised service composition is a research challenge because the evaluation of all possible service peers can lead to a combinatorial explosion of solutions. In compositions involving large group of peers, an exhaustive search could be very time consuming. Moreover, it is particularly important for the QoS aware composition process itself to be fast. Especially for the synchronised or interactive systems, long delays may be extremely unacceptable. For example, the user of a ticket booking system might not want to wait for a long time while the system searches for candidate services offering flight tickets with the lowest booking fare. While peer composition is viewed as the ability to combine existing peers together in order to generate new functionalities or services, it usually becomes extremely hard to conduct when handling a large number of services. This is because that, to find the solution of such a QoS–aware composition problem is NP-hard. Although previous research efforts have looked at many pragmatic ways to effectively deal with the composition of Web services into executable workflows (e.g. [17, 20]), a remaining concern is how to optimise the service composition in terms of making system more effective. Thus, it is necessary to have an effective approach to handle the composition issue quickly, and also, adaptively.

In this paper we propose to adopt ACO (Ant Colony Optimisation) to enhance performance of service composition for P2P workflow and prove it is suitable to be used in QoS-aware service composition. The remainder of this paper is organised as follows. After related work is presented in Section 2, Section 3 details the proposed approach, i.e. the ACO based method for QoS-aware composition in P2P workflow. Section 4 reports and discusses the experiment results obtained in the simulations, and finally, Section 5 concludes the paper.

2 Related Work

QoS–aware service selection for composition attracts many interesting applications and trials of various methods and search strategies, mostly based on operation research (OR) or artificial intelligence (AI). In general, most of the accepted approaches for effective service composition are focusing on how to find the optimal/close-optimal combinations with the minimum cost (e.g. time, computation resource constraints), and they are usually dependent upon Integer Programming or

Genetic Algorithms, or mixed up with some other optimisation strategies. Recently ACO has been widely used for the problems such as allocating jobs in robot networks and searching the shortest path, but has rarely been considered for service selection issue. Here we would firstly discuss about those typical approaches used for service composition, and briefly compare with our proposed ACO solution.

The Integer programming (IP) solutions with regard to dynamically finding the best service combination have been proposed in some recent papers ([1], [20]). These works consider linearity of the constraints and the objective functions, and find the best combination of the concrete services. From the QoS point of view, they're conducted at run time. Zeng et al. [20] essentially focus on the cost, response time, availability and reliability attributes, where logarithmic reductions are used for the multiplicative aggregation functions, and the model is claimed to be extensible with respect to other similarly behaving attributes. The need to deal with more general constraint criteria, such as service dependencies or user preferences, is strengthened in the work presented by Aggarwal et al. [1]. However, they do not explain how the optimisation problem could be solved. In this paper, we adopt the ACO instead, as any kind of constraint could be handled herein, and we also provide some empirical data to assess the performance of our method versus the (linear) integer programming solution in the service composition setting, especially in terms of comparison of the computation time with the same number of tasks. An alternative to find a close-optimal solution could be to use nonlinear integer programming techniques, but the maturity of the available tools is questionable, and the application of these methods in our setting requires further investigation. A survey of some nonlinear techniques is contained in a paper by Grossmann [9].

Genetic Algorithms (GA) is viewed as a distinguishing heuristic method for seeking a close-optimal combination. With regard to service composition, Canfora et al. [2] utilised GA and focused on the situation where there may be more than one service candidate that provides identical functionality but has different QoS, to determine which service candidates should participate in a required composite service. When compared to widely-used linear integer programming methods, their genetic algorithm deals with non-linear functions while it scales well with an increase in the number of tasks. Technically, ACO is also qualified to deal with that situation adaptively, as it uses updated information from time to time to approach to the optimal result. A randomised heuristic method based on the general principles of genetic search strategy was described by Chockalingam et al. [4] to solve the mapping problem, in which parallel tasks are assigned to a multiprocessor to minimise the execution time. Similarly to a P2P-based service system, all peers are required to cooperate and work together to accomplish a designed composite task, but we would consider using ACO to save more time. The work conducted by Cao et al. [3] is a recent development on the application of GA to the service selection problem in the context of Web service composition. A service selection model defines a business process as a composition of many service agents. Each service agent corresponds to a set of multiple Web services, which are provided by different service providers to perform a specific task. Nevertheless, taking into account the uncertainty and self-adaptive capability, GA may be not as good as ACO to deal with dynamic situations or unexpected events (e.g. when agent breaks down or its quality is changed during iterations). Instead, ACO can take more advantage of heuristic information and adapt itself

to any changing circumstance quickly and easily, and also can facilitate to formulate the selection problem based on the nature of a service application effectively.

A lot of researches regarding the QoS in Web services focus on the development of QoS ontology languages and vocabularies, as well as the identification of various QoS metrics [11] and their measurements with respect to semantic Web services. For example, papers [10] and [13] emphasised a definition of QoS aspects and metrics. In [10], all of the possible quality requirements were introduced and divided into several categories, including runtime related, transaction support related, configuration management and cost related, and security related QoS. Both of the papers shortly present their definitions and possible determinants. In our previous work [15, 16, 19], we focused on P2P-based service selection and the extensible description of non-functional properties via OWL-S and WSMO. In [19], we presented a first sketch of QoS-aware service selection, however, with special attention to the extraction of the ontological description of services and the design of the selection process with OWL-S. With regard to the selection process, the prototype presented in that paper has the limitation in terms of dealing with multi-specifications, because it only considers "ResponseTime" as the selection criteria, by which the selection is not quite realistic for effective services composition. Therefore in [15, 16], we extended the description of non-functional properties via modelling-driven WSMO specification, and presented an algorithm for Peer coordinator to automatically identify the best peers through unifying qualities and properties. Nevertheless, based upon our earlier works, the main aim of this paper is to propose a quick and effective ACO-based approach for P2P-based QoS-aware service composition.

3 The Methodology

Ant Colony Optimisation is a well-established optimisation technique based on the principle that real ants are able to find the shortest paths between their nest and a food source [8]. This mechanism works on the basis of pheromones, some kind of biochemical scent, which is left behind by the ants. Other ants are attracted by these pheromones and always walk in the direction with the highest pheromone concentration. This natural behaviour was first adopted as an optimisation technique by Dorigo, Colorni, and Maniezzo [5, 7]. Their Ant System provides the foundation of the optimisation method presented here. This behaviour is the basis for a cooperative interaction, and this manner can be applied not only to solve discrete optimisation problems but also to solve both static and dynamic combinational optimisation problems [12].

In this work an artificial ant is an agent which moves from node to node on a composition graph. It chooses the nodes to move by using a probabilistic function of both trail accumulated on links, and a heuristic value, which was chosen here to be a function of the number of services within a workflow. By a probability, artificial ants prefer nodes which are connected by links with relatively more pheromone trail. The following are the three heuristic hints from natural ant behaviours that we have translated to our artificial ant colony: 1. the preference for paths with a high pheromone level; 2. the higher rate of growth of the amount of pheromone on ideal paths; 3. the path-mediated communication among ants.

For the purpose of selection processes, ResponseTime, Cost, Availability and Reputation are utilised often as QoS factors, and they can be described as a peer's non-functional properties with WSMO. Assume a composite service "*l*", we define that *RT(l)*, *C(l)*, *A(l)* and *R(l)* are the respective normalised QoS factors (ResponseTime, Cost, Availability and Reputation) in the interval [0,1]. These normalised QoS attributions can be computed by applying the rules described in Table 1, which is based on a simple way of normalisation. The composite service "*l*" contains "*n*" atomic Web services, and we assume that the number of artificial ants is "*m*" at each iteration (after an iteration, ideally every ant can find a combination of possible candidate peers for composite service "*l*").

Table 1. Normalising QoS Attributions

Response Time *(RT)*	$RT(l) = \frac{ReponseTime(l)}{\sum_{i=1}^{m} ReponseTime(l)}; \quad ReponseTime(l) = \sum_{i=1}^{n} ReponseTime(ws_i)$
Cost *(C)*	$C(l) = \frac{Cost(l)}{\sum_{i=1}^{m} Cost(l)}; \quad Cost(l) = \sum_{i=1}^{n} Cost(ws_i)$
Availability *(A)*	$A(l) = \frac{Availability(l)}{\sum_{i=1}^{m} Availability(l)}; \quad Availability(l) = \prod_{i=1}^{n} Availability(ws_i)$
Reputation *(R)*	$R(l) = \frac{Reputation(l)}{\sum_{i=1}^{m} Reputation(l)}; \quad Reputation(l) = \prod_{i=1}^{n} Reputation(ws_i)$

We define a cost function for evaluating QoS of a service composition *l* as follows:

$$Q(l) = \frac{w_1 \cdot RT(l) + w_2 \cdot C(l)}{w_3 \cdot A(l) + w_4 \cdot R(l)} \tag{1}$$

Where: w_1, w_2, w_3 and w_4 are weights which indicate the importance of the QoS factors for service integrator (or user). Q(*l*) represents cost function of composite service *l*, and the objective is to minimise the cost function value of composite service "l". The reason why we define the function in this way is that we need to differentiate the preferences of QoS properties, such as ResponseTime (is preferred as low as possible) and Availability (is preferred as high as possible).

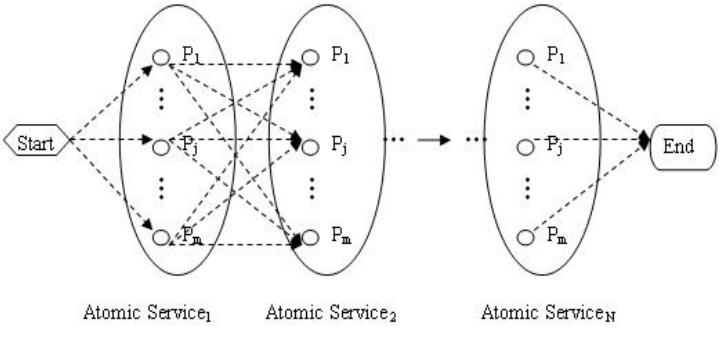

Fig. 1. Composition graph for services workflow

In Figure 1, we may assume that there are 'm' candidate peers $\{P_1, .., P_m\}$ and 'N' atomic services {Atomic Service$_1$, …, Atomic Service$_N$} in a service composition process. That is to say, there are 'N' groups of peers, and each group is $\{P_1, .., P_m\}$. Clearly, the goal is to find right peers from each group for each atomic service. Hence, by applying the ACO based approach, *m* artificial ants are initially placed on Atomic Service$_1$'s nodes (from P_1 to P_m). Ideally, the more ants, the better, but that would consume much longer time for iteration. Hence, in a balanced way, we normally set the number of ants is the number of peers. For any step *i*, ants move to newer possible nodes (the peer group for Atomic Service$_i$) until it reaches the end node (the peer group for Atomic Service$_N$). When all the ants have completed a path, the ant that made the path with the lowest cost function value *Q* would modify the links belonging to its path by adding an amount of pheromone trail. In each iteration, each ant generates a feasible composite service by choosing the nodes according to a probabilistic state transition rule.

For the selection of a node, ant uses heuristic factor as well as the pheromone factor. The heuristic factor denoted by η is a heuristic desirability about an ant moving from a current node to another, and the pheromone factor denoted by τ is an indication of how many ants have visited the link. The probability of selecting a next node (from Atomic Service$_i$ to Atomic Service$_j$) is given by:

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{h \in H} \tau_{ih}^{\alpha} \cdot \eta_{ih}^{\beta}} & if \ \rightarrow \ i \in H \\ 0 \rightarrow Otherwise \end{cases} \tag{2}$$

where P^k_{ij} is the probability with which ant *k* chooses a next node to go. *H* is the set of nodes that have not been passed by any ant for Atomic Service$_j$, and *h* is a node (i.e. a peer in the group for Atomic Service$_j$) which has not been visited yet by ant *k*. The parameters α and β control the relative importance of the pheromone versus the heuristic information η_{ij}, which is the heuristic desirability of the ant on node *i* moves to node *j*, is given by:

$$\eta_{ij} = \frac{1}{Nj} \tag{3}$$

where N_j is the number of Web services between Atomic Service$_j$ and Atomic Service$_n$. Actually, for service peer composition, η can be viewed as a heuristic stimulus which can keep ants moving from source towards destination covering all atomic services.

The whole path's pheromone update is applied at the end of each iteration by only one ant, which can be the iteration-best or the best-so-far, and the update formula is defined as follows:

$$\tau_{ij} = \begin{cases} (1-\rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau, if \ (i, j) \in L \\ \tau_{ij} \rightarrow Otherwise \end{cases} \tag{4}$$

where $\rho \in (0, 1]$ is a parameter that controls the speed of evaporation of pheromone, and $\Delta\tau = 1/Q(L)$, here L is the current best path with the lowest Q value.

To facilitate the peers' service composition and deal with dynamical situations, the proposed ACO algorithm for service composition is presented as follows. Algorithm 1 is designed to quickly find the close-optimal combination for a composite service, and Algorithm 2 is responsible for monitoring uncertainties and then generating replacement solution for the remaining part of composition in case one or more peer(s) becomes unavailable. In Algorithm 1, each atomic service on a peer is regarded as a virtual node of a potential path that might be found by ants. After each iteration, a possible path would be found by ants, and then the combination QoS is calculated and recorded, afterwards the visited nodes' pheromones would be updated. Finally, the nodes with more pheromones would be more possibly consisted of an eventual path, i.e. a close-optimal combination would be generated. For dynamical changes and uncertainties during execution, the Algorithm 2 is set as a supplement to reasonably cope with peer's breakdown or if there are any other needs.

Algorithm 1. ACO-based Approach for Service Peer Selection

Step 1. Initialisation:
An initial population of ant colony individuals Ant_k, k=1, 2,…, m (m can be the number of candidate nodes), is initialised in this step. N is the number of atomic Web services. Format the composite nodes list of each ant to provide the following steps to record the ant composition history. Set a small amount of pheromone on each node as initial condition for each link. Set iteration counter I=1, and initialise the maximum iteration number $I_{max.}$ (When I reaches I_{max}, Stop.)
Step 2. Starting travel:
for ant k=1 to m **do**
  Place Ant_k on candidate node of the first atomic service.
end for
Step 3. Searching for next node:
Repeat until the composition list is full
for k=1 to m **do**
  if ant Ant_k is already reached the final node of composition
    Then Break and **do** Ant_{k+1}.
  else
    Choose the candidate nodes and calculate the probability P^k given in formula (2).
  end if
  if the ant Ant_k is stuck at a dead node or reach the maximum transport action number N.
    Then move the ant back to the start node, and clear the current composition list of Ant_k.
  end if
end for
Step 4. Calculate the composite service's QoS value for each ant's journey:
for k=1 to m **do**
  Compute the Q value according to Q(l) in formula (1) based on the visited nodes
end for
Step 5. Update the pheromone factor:
For each path's nodes update the pheromone value according to formula (4)
Find the best composition path and update the best composition history.
Step 6. Check to stop criterion:
if ($I<I_{max}$)
  Then record the best composition list, and **Goto** Step 2.
else
  Return the best composite service and stop.
end if

Algorithm 2. Monitoring and Replanning

```
While (1)
    if any breakdown/events happen
        Put all incomplete atomic Web services into an arraylist L;
        Delete the dead peers and refresh the peer group;
        Do Algorithm 1 for replanning the left composition;
    end if
end while
```

4 Experimental Results and Evaluation

In this section, we evaluate performance and feasibility of the proposed ACO method. To verify the proposed approach, experiments are conducted based on a few different scenarios. Essentially, we conduct a few experiments to prove the proposed method: 1. apply ACO in a general composite workflow, and compare ACO with optimisation method in static environments, in terms of computation time; 2. test the re-planning performance by randomly setting a few peers dead in dynamical environments.

In simulation, we established the environment with Matlab, and all experiments were conducted on a PC with Intel Pentium4 3.0GHz, 2MB; 800 MHz FSB; DDR2 1GB @ 667MHz; and MATLAB 2008a. All QoS data (i.e. ResponseTime, Cost, Availability and Reputation) of service peers are randomly generated in Matlab.

4.1 Quick Composition and Comparison

As shown in Figure 2, a general workflow, which contains structures of split and flow for service composition, are depicted, with 7 atomic Web services (AWSs) involved. As a matter of fact, for peer selection process, now we do not consider about differences between simple sequential composition pattern and other mixed patterns, as our emphasis is on selecting an appropriate peer for each atomic service amongst service peers. Therefore, in this work, the selection method we proposed can be extensible for service selection in all sorts of composition patterns, even though different composition patterns may impact the calculation of cost functions.

In this experiment, we assume that there are 10 service candidate peers, and utilise the proposed method to select appropriate service peers which are able to conduct those 7 atomic Web services and provide better overall service quality. The major goal of this method is to identify better peers with pheromones and probabilities for atomic services. In the selection process, there is little difference between a purely

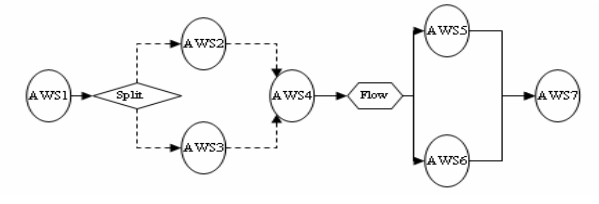

Fig. 2. A general composition workflow

sequential workflow and the mixed one, while using ACO method to search for proper candidate peers. Ants will exploit their own heuristics at join or split nodes.

Figure 3 shows the nature of ACO regarding best composition value and average value at each iteration for this experiment. These composition values are based on calculated Q values. At every iteration, the change of best value indicates that ants found a better path of composition than the previous iteration. As shown in left part of Figure 3, the best composition values were getting better and better from the beginning to the 9^{th} iteration, and afterwards there was no changes happened since ants could not found better solution. In the right part of Figure 3, the situation of average composition values changed obviously during all the 20 iterations. Here, the average value is the mean value of all possible compositions found by ants at each iteration, and the changes of average value are caused by the probabilities of ants' choices and the amount of pheromones. However, the trend of the best composition value is led well by heuristic information, as shown in the left part of Figure 3.

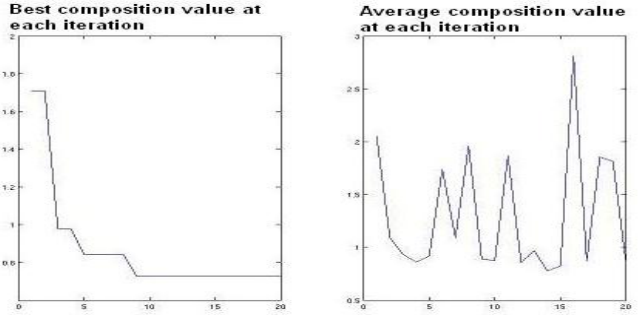

Fig. 3. The Q values in ACO iterations

Table 2. Calculated Q Values of Peers for Atomic Services without ACO Approach

	AWS 1	AWS 2	AWS 3	AWS 4	AWS 5	AWS 6	AWS 7
Peer 1	0.873	0.398	0.780	**0.008**	0.455	0.950	**0.144**
Peer 2	**0.142**	0.413	0.554	Inf	0.869	0.630	0.394
Peer 3	0.682	0.963	Inf	0.677	0.219	0.378	0.259
Peer 4	Inf	**0.075**	0.751	0.217	0.466	0.471	0.338
Peer 5	0.781	0.839	0.532	0.054	0.387	0.887	0.853
Peer 6	0.946	0.248	**0.009**	0.029	Inf	**0.130**	Inf
Peer 7	0.212	0.729	0.683	0.236	0.437	0.442	0.145
Peer 8	Inf	0.695	0.040	0.292	0.372	0.658	0.798
Peer 9	0.990	0.229	0.153	0.229	0.449	Inf	0.162
Peer 10	0.458	0.333	0.830	Inf	**0.193**	0.419	0.465

Table 3. Selection Results with Q Values by ACO Method

AWS1	AWS2	AWS3	AWS4	AWS5	AWS6	AWS7
Peer 2 (0.142)	Peer 4 (0.075)	Peer 6 (0.009)	Peer1 (0.008)	Peer 10 (0.193)	Peer 6 (0.130)	Peer 1 (0.144)

Table 2 shows each service peer's Q value that is calculated based on formulas in Table 1, without ACO approach, and each data represents the relationship (Q value) between a candidate peer and an atomic service rather than a composition service. The infinite (Inf) value of a peer means that the service peer can not fulfil the corresponding atomic service, and Q values of selected peers are calculated based on a set of random data which are representing relevant peers' non-functional properties. Table 3 then lists the results from using ACO method. From the two tables, we can see that the combination results by ACO method are exactly the same selections with the best Q value for atomic services in Table 2. It suggests that ACO based approach can also generate the results without completely knowing Q values of each peer. Actually, exhaustively calculating all peers' Q values in this way (e.g. Table 2) is a simple optimisation method to select service peers for service composition, as it can easily find the best peers combination after knowing each peer's Q values for atomic services. Hence, the major cost of computation by the optimisation method is the calculation on all possible combinations, and when the number of atomic services and peers increases, the time cost would greatly increase as well. Given the computation cost, ACO based approach does not need to calculate all service candidates' Q values beforehand, since the calculation is only needed to conduct when a service peer is chosen with a certain probability. In other words, ACO approach can save much more computation time than the optimisation method, particularly for a large number of possible combinations.

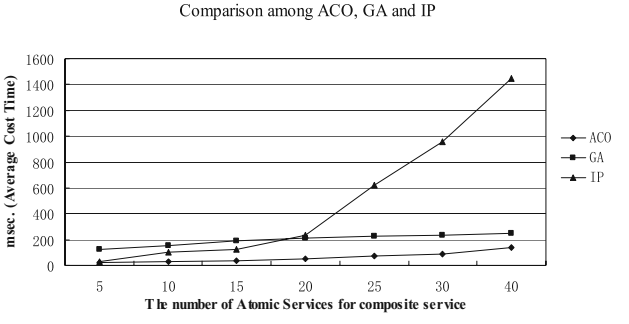

Fig. 4. Comparison of ACO, GA and IP

As shown in Figure 4, ACO is the least time consumer through all the experiments. When the number of concrete services is small, say, less than 15, IP outperforms GA. For about 20 atomic services the performances of ACO and IP tend to be the same. Then with more atomic services, while the ACO and GA are able to keep their computation time performance almost constant, this is not the case for IP based optimisation method, where we see an exponential growth due to the corresponding increment of the number of variables needed to represent the problem. In this experiment, we assumed a set of workflows consisting of 5 to 40 atomic services, and the same numbers of service peers are involved in the selection for each atomic service. The experimental settings of GA were based on [2], where GA is with crossover probability of 0.7, a mutation probability of 0.01 and the number of generation is 100. The parameters of the ACO in the simulations were set as: α, β are 1 and 5, respectively, ρ (speed of evaporation) is 0.1, and 20 iterations (as few changes occur after 20 runs in the test) for each trial.

4.2 Effective Re-planning

In reality, composite services are often conducted in scenarios that are changing dynamically and unexpectedly, due to real environments' uncertainties. In this case, peers need to take actions quickly for the re-planning and/or re-adjustment if any events occur. For example, some of the selected service peers for a composite service might crash down during execution, and then the expected workflow would be stuck forever until a replacement was given. Hence, it is necessary for the ad-hoc execution of a whole workflow to be affiliated with a replanning scheme, i.e., the ability to adaptively find a replacement in order to resume the remaining incomplete tasks. Figure 5 depicts 6 scenarios: two with "5 atomic Web services and 20 peers", two with "10 atomic Web services and 30 peers" and two with "15 atomic Web services and 40 peers". During the composite service execution, we assume some service peers to be dead unexpectedly, e.g., 1, 2 and 4 service peers can be suddenly out of order in each scenario. In the simulation, consumed time for replanning vary from case to case, the maximum replanning time is 45 (msec.), and the minimum is 7 (msec). "F" means that there is no available peer(s) for the required atomic services. The amount of consumed replanning time is actually dependent on the dead peer's virtual location in workflow process where a number of atomic services have not completed when the event occurred.

Based on all experimental results, we conclude that the ACO-based approach leads to effective and adaptive QoS-aware composition with less computation time and reasonable quality of solution.

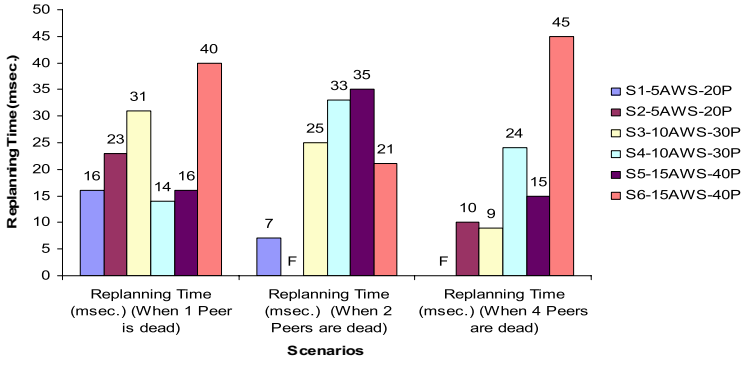

Fig. 5. Replanning Performance of ACO in Dynamical Environments

5 Conclusion and Future Work

In this paper, the ACO-based approach is proposed to tackle the QoS-aware peers' composition problem, in terms of determining a set of peers to be bound to atomic services. Those services are contained in an orchestration, and candidate peers are heuristically selected based on QoS aspects for the orchestration by the means of ACO. With our experimental results, ACO allows a P2P based e-service system to handle QoS attributes with non–linear aggregation functions adaptively and

efficiently, and it is a balanced practice to apply ACO to quickly finding the close-optimal solution. The main contribution of this paper is a heuristic approach to effectively determine and select appropriate service peers for composition, and also the uncertainties in the real service execution scenarios are considered with a proper re-planning scheme.

In the future work, we would investigate some probabilistic models, e.g. Partially Observable Markov Decision Process (POMDP), to cope with more complicated uncertainties within actual service compositions and executions, and also we would implement an essential P2P based service prototype and testify the approach to see how sound it can suit the real applications.

References

[1] Aggarwal, R., Verma, K., Miller, J., Milnor, W.: Constraint driven Web service composition in METEOR-S. In: Proceedings of the 2004 IEEE International Conference on Services Computing, pp. 23–30. IEEE Computer Society, Los Alamitos (2004)

[2] Canfora, G., Penta, M.D., Esposito, R., Villani, M.L.: An approach for QoS-aware service composition based on genetic algorithms. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, New York, USA, pp. 1069–1075 (2005)

[3] Cao, L., Li, M., Cao, J.: Using genetic algorithm to implement cost-driven Web service selection. Multiagent and Grid Systems 3(1), 9–17 (2007)

[4] Chockalingam, T., Arunkumar, S.: Genetic algorithm based heuristics for the mapping problem. Computers and Operations Research 22(1), 55–64 (1995)

[5] Colorni, A., Dorigo, M., Maniezzo, V.: Distributed Optimisation by Ant Colonies. In: Proceedings of the European Conference on Artificial Life, Paris, France, pp. 134–142. Elsevier Publishing, Amsterdam (1991)

[6] Curbera, F., et al.: Unraveling the Web Services: An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing 6(2), 86–93 (2002)

[7] Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man and Cybernetics - Part B 26(1), 1–13 (1996)

[8] Goss, S., Aron, S., Deneubourg, J.-L., Pasteels, J.M.: Self-Organized Shortcuts in the Argentine Ant. Naturwissenschaften 76(12), 579–581 (1989)

[9] Grossmann, I.: Review of nonlinear mixed-integer and disjunctive programming techniques. Optimization and Engineering 3(3), 227–252 (2002)

[10] Lee, K., Jeon, J., Lee, W., Jeong, S., Park, S.: QoS for Web services: Requirements and Possible Approaches. W3C Working Group Note 25 (2003), http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/

[11] Liu, Y.T., Ngu, A.H.H., Zeng, L.Z.: QoS computation and policing in dynamic Web service selection. In: Proceedings of International Conference on World Wide Web, pp. 165–176. IEEE CS Press, New York (2004)

[12] Lorpunmanee, S., Sap, M.N., Abdullah, A.H., Chompoo-inwai, C.: An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment. International Journal of Computer and Information Science and Engineering 1(4), 207–214 (2007)

[13] Ran, S.: A model for Web services Discovery with QoS. ACM SIGecom Exchanges 4(1), 1–10

[14] Shen, J., Krishna, A., Yuan, S., Cai, K., Qin, Y.M.: A Pragmatic GIS-Oriented Ontology for Location Based Services. In: The 19th Australian Software Engineering Conference (ASWEC 2008), Perth, Australia, pp. 562–569. IEEE Computer Society Press, Los Alamitos (2008)

[15] Shen, J., Yuan, S.: Adaptive E-Service Selection in P2P-based Workflow with Multiple Property Specifications. In: Ting, I., Wu, H. (eds.) Book Web Mining Applications in E-commerce & E-services, pp. 153–168. Springer, Berlin (2009)

[16] Shen, J., Yuan, S.: Modelling Quality and Spatial Characteristics for Autonomous e-Service Peers. In: The 20th International Conference on Advanced Information Systems Engineering (CAiSE 2008), Forum, Montpellier, France, June 2008, vol. 344, pp. 49–52. CEUR-WS (2008) ISSN: 1613-0073

[17] Vanrompay, Y., Rigole, P., Berbers, Y.: Genetic algorithm-based optimization of service composition and deployment. In: Proceedings of the 3rd international workshop on Services integration in pervasive environments, pp. 13–17 (2008)

[18] Web Services Architecture Requirements Working Group (2004), http://www.w3.org/TR/wsa-reqs

[19] Yuan, S., Shen, J.: Mining E-Services in P2P-based Workflow Enactments. special issue Web Mining Applications in E-commerce and E-services of Online Information Review 32(2), 163–178 (2008)

[20] Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for Web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)