International Journal of Grid and Distributed Computing
Vol. 9, No. 6 (2016), pp.103-116
http://dx.doi.org/10.14257/ijgdc.2016.9.6.11

A Data Volume Aware Ant Colony Optimization Approach for Geographical Knowledge Cloud Service Composition

Xiaozhu Wu[1, 2], Chongcheng Chen[2] and Hongyu Huang[3]

[1]*College of Computer Science and Mathematics, Fuzhou University, Fuzhou 350012, China, wxz@fzu.edu.cn*
[2]*Key Lab of Spatial Data Mining & Information Sharing, MOE, Fuzhou University, Fuzhou 350002, China, chencc@fzu.edu.cn*
[3]*Key Lab of Spatial Data Mining & Information Sharing, MOE, Fuzhou University, Fuzhou 350002, China, hhy1@fzu.edu.cn*

Abstract

Geographical knowledge cloud service is a typical online service that provides big spatial data analysis with the function of knowledge discovery or decision-making. The composition of geographical knowledge cloud service imposes stricter requirements for better overall QoS and execution efficiency of the service chain. In this paper, we present a data volume aware ant colony optimization approach called DVA-MOACO algorithm for geographical knowledge cloud service composition. Our algorithm utilizes a multi-index service quality evaluation model, and improves the transition probability while considering the data transfer cost and other QoS constraints simultaneously when ant finding path. Our algorithm could reach the Pareto near optimal solution rapidly with better QoS performance and lower data transfer cost from numerous candidate solutions.

Keywords*: service composition; multi-object optimization; ant colony algorithm; geographical knowledge service; cloud computing*

1. Introduction

Geographical knowledge cloud service (GKCS) brings a new paradigm to deal with huge geospatial data by integrating web services, GIS, knowledge discovery and cloud computing technology. The aggregation of spatial data mining algorithms, spatial decision support models and geographical process simulation models makes GKCS capable of supplying users with on-demand spatial data knowledge discovery and decision support functions, which has cross areas, cross platforms and cross nodes ability properties.

The cloud computing technology is essential for GKCS to provide scalable and elastic running environment. With outstanding distributed data storage and parallel computing capability, cloud platform could host and aggregate great deal of spatial data services and data process services. The cloud platform with core functions such as geospatial data deep processing, distributed knowledge discovery and knowledge sharing (provided by GKCS) could be named as *geographical knowledge cloud* [1].

In applications, GKCS usually need to be composed to form a service chain to fulfill complex task. The GKCS to be composited also named as atom service, which inherits the cooperation capability and platform-independent characteristic from web services. With the number of atom services rapidly increasing all over the world, there are more and more GKCS available for composition while they have similar functions and quite different service quality (QoS). Hence the service chain shall search atom services from huge candidate services pool according to functional requirement and QoS constraints, which is a time-consuming procedure and not feasible in online cloud service scenario. Furthermore, the QoS constraints always contain multiple conflict objects, and these

ISSN: 2005-4262 IJGDC

objects could not be reached simultaneously. For example, users shall pay higher price for better cloud service with higher performance where the price and performance is a couple of conflict objects. The service composition optimization problem that needs to consider multiple conflict objects is known as Multi-objective Optimal Problem of Service Composition (MOPSC) [2]. The MOPSC has been proven to be a NP-hard problem that could not be resolved in linear manner [3]. For instance, assume that a user utilize one data preprocess service from A cloud and another spatial clustering algorithm service form B cloud to analysis 10GB geospatial data, and the data preprocess service's output is the input of the spatial clustering algorithm service. In such circumstance, the transmission of 10GB data and network condition between the two services will dramatically affect the efficiency of this service chain. Owing to the above reasons, the object of optimization of GKCS composition is to find the optimal candidate atom services, which have better QoS performance and higher overall execution efficiency.

To cope with the challenge of MOPSC, there have been many attempts to find near optimal solutions by utilizing novel optimization algorithms such as heuristic algorithm, evolutionary algorithm, bionic algorithm and integer programming method [4]. In despite of the difference of basic idea, these researches always adopt local optimization strategy or global optimization strategy to achieve the optimal or near optimal solutions [5][6]. The local optimization strategy aims to seek best candidate service for each atom service and without considering the overall quality of service chain. This strategy is easy to be implemented but not necessarily leading to global optimal solution. In contrast, the global optimization strategy tries to get global optimal solution, which takes into account all objects. Most of researches apply global optimization strategy to solve MOPSC and focus on how to accelerate optimization procedure and get better solutions.

As the structure and overall data transfer cost of service chain will affect the optimization result greatly, in this paper, we propose a new ant colony optimization approach for geographical knowledge service composition that takes advantage of parallel computing capability of cloud platform. The proposed method evaluates the QoS of complex service chain which consists of sequence, switch, loop and parallel structure and considers the data transfer cost simultaneously. Compared with previous ant colony based methods for MOPSC, the proposed method could produce solutions with better performance by balancing QoS with data transfer volume of service chain.

This paper is organized as follows. In Section 2 we introduce relate works. The QoS evaluation model for GKCS is presented in Section 3. In Section 4, the detail of DVA-MOACO algorithm is depicted. The algorithm evaluation is described in Section 5. We conclude this paper and provide suggestions for future work in Section 6.

2. Relate Works

Towards the problem of QoS-based service composition, various methods have been proposed which consist heuristic algorithm, evolutionary algorithm, bionic algorithm and integer linear programming methods.

Luo Yuan-Sheng pointed out that the execution time of heuristic algorithm of multi constrained optimal path (HMCOP) will increase with the significant growth of service search space; to improve the efficiency, Luo apply the cross entropy as the heuristic information to the HMCOP algorithm to guide the search process, and achieved good results [7]. Klein proposed a QoS service composition method by the application of linear programming and heuristic algorithm. The first step of the method is to use linear programming method to reduce the search space, the second step is to use the hill climbing method to explore solution space; this method can reduce the search time limited, but the quality of the solution depends on the reduced solution space [8].

Particle swarm optimization algorithm and genetic algorithm is the outstanding evolutionary algorithm. Huang Lican proposed an improved particle swarm optimization

algorithm name as IDPSO for the optimization of services composition, which can change particle's marching step according to different situation; when the particles is far from the optimization target, the step size becomes large, otherwise small. This method can accelerate the convergence speed and avoid getting into local optimal [9]. Tang Maolin presents a hybrid genetic algorithm to find the optimal service composition with best QoS that apply fitness function, penalty function and local optimizer to realize population optimization [10]. Huang ChunChe also proposed a genetic algorithm considering multi constraints for the service composition problem [11].

Due to the ant colony algorithm (ACO) has the property of decentralized, flexibility, robustness and self-organized, researchers pay attention to use ACO to tackle the problem of service composition [12]. In order to reduce the search cost, Agostino *etc*. proposed similarity-based a self-organizing framework which places services in same P2P node that often work together, and a P2P-based parallel ant colony algorithm is implemented on the framework[13]. Wang ZhiJian merge max-min ant algorithm (MMAS) with cultural algorithm framework and presents a new optimization algorithm of C-MMAS, which adopts the new state transition probabilities and pheromone update method to achieve good performance[14].

Other researches focus on the problem of service composition in data intensive applications [15]-[22]. Pennec *etc*. [15] introduce a service-based workflow manager to describe data-intensive service composition such as image processing in grid environment. This work could efficiently process the resulting computations by transparently exploiting different parallelism levels, while the problem of dynamic service selection in order to reduce data transfer cost is unhandled. Similar service composition models presented in [16][17][20] proposed composition language and dynamic reconfiguration methods. Papers [18][19][21][22] consider the QoS model of composed service compromise with data transfer cost and utilized genetic algorithm, GP-Tabu algorithm *etc*. to cut down the overall data transfer cost.

3. QoS Evaluation Model

3.1. QoS Evaluation Model for Atom Service

QoS evaluation model aims to assess the overall quality of atom service or service chain according to multiple quality indexes and it is a key factor in candidate service selection. Traditionally, the QoS of atom service can be calculated by weighted summation of all indexes, this method could be summary as following formula:

$$Q(S) = w_1 * qos_1 + w_2 * qos_2 + \ldots + w_i * qos_i$$

where $Q(S)$ is the overall quality of atom service S, w_i is the weight of the ith QoS index, and qos_i denotes the normalized QoS value of ith index. This method has the advantage of low computation cost and convenient logical. Otherwise, the QoS value produced by this method could not objectively express the quality of atom service in that the setting of weight is extremely subjectively. In order to eliminate the artificially interference, we use the following method:

$$Q(S) = \sqrt{\sum_{i=0}^{m} \left| \frac{qos_i^{best} - qos_i}{qos_i^{best}} \right|^2} \quad (1)$$

In the above formula, qos_i^{best} denotes the best value of ith index which decided by the user or the best value of candidate services in ith index, qos_i represents the QoS value of ith index of atom service S, and m is the number of index. If we use

$Q^{best}(S)=\{qos_1^{best},qos_2^{best},\ldots,qos_m^{best}\}$ to denote the best performance of atom service S, then the formula (1) actually means the gap between real performance with the best performance $Q^{best}(S)$. Owing to some of the QoS indexes may be mutual conflict, there may not one atom service can reach the best value on all indexes. Hence $Q^{best}(S)$ means the ideal performance that there is no one candidate service can achieve. In such circumstance, the smaller of gap in formula (1) means the better performance.

3.2. QoS Evaluation Model for Service Chain

3.2.1. Service Chain Splitting Algorithm: The above QoS evaluation model could only apply to single service and not suitable for service chain that contains multiple atoms services and complex structure. Obviously, the QoS of service chain cannot compute by cumulating QoS of atom services directly. In this paper, we propose a QoS evaluation model for service chain which firstly split the service chain into several segment according to structure, then calculate QoS value of service chain in each QoS index respectively, finally treat service chain as an atom service and use formula (2) to get its' QoS value.

Consider a service chain as depicted in Figure 1. This service chain has six abstract atom services {AS1, AS2, AS3, AS4, AS5, AS6} and each atom service has multiple candidate services. Suppose this service chain consists of {S11, S21, S31, S41, S51, S61} and each atom service's QoS value of response time is {0.5, 0.4, 0.3, 0.6, 0.5, 0.8}. The overall response time of service chain can be different according to its structure. Consider the three paths after AS1, which are {AS2, AS5,}, {AS3} and {AS4}, have different relations. In case they are parallel, then the overall response time of service chain is 0.5+Max {(0.4+0.5),0.3,0.6}+0.8=2.2. In case they are optional paths and each to be executed in possibility of {0.4, 0.3, 0.3}, then the overall response time of service chain is 0.5+((0.4+0.5)*0.4+0.3*0.3+0.6*0.3)+0.8=1.93.

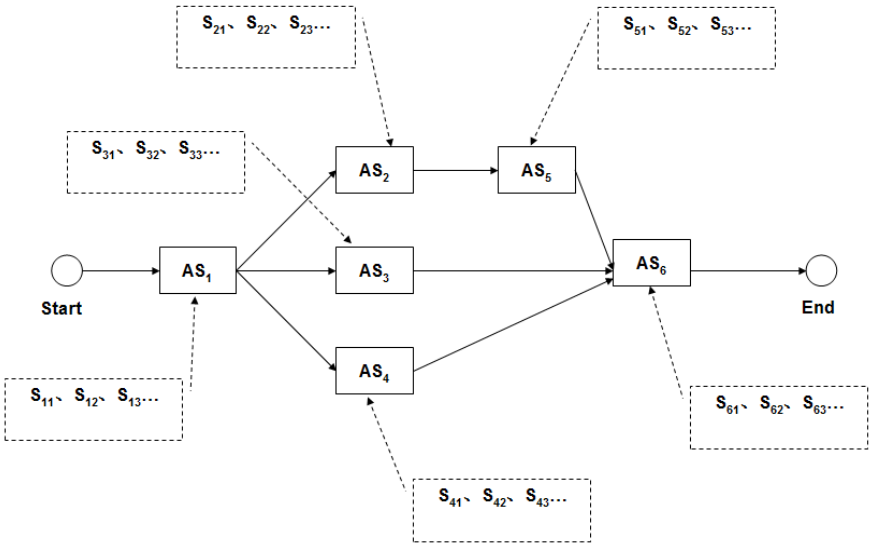

Figure 1. The Example of Service Chain

Owing to it is easier to compute the overall QoS value of service chain that contains only sequence structure. It is an ideal method to reduce the complexity by splitting service chain into multiple sequential segments that have simple structure. The idea of service chain splitting is to split on joint nodes and forking nodes until each segment has no joint nodes or forking nodes. If some segments contain joint nodes or forking nodes, then the

segments shall be split recursively until only sequence structure remains. The joint node is defined as the node that has more than one out going paths such as AS1 in Figure 1. And the forking node is defined as the node that has more than one incoming paths such as AS6 in Figure 1. Hence the service chain in Figure 1 can be split as segment1={start, AS1}, segment2={{AS2, AS5}, {AS3}, {AS4}} and segment3={AS6, end}. The three segments form as sequence structure so that the overall QoS value of service chain can be directly sum from them according to different QoS indexes. The service chain splitting algorithm is described as follows:

Table 1. Service Chain Splitting Algorithm

Input：	Service Chain, SC={N, E}
Output：	Split Service Chain, SSC={segment$_1$, segment$_2$….}

```
1.  function decomposite(SC)
2.  {
3.      SSC=NULL;
4.      segment1=segment2=segment3=NULL; // initial all segment to be NULL
5.      ifFoundSplittedNode=false;
6.      segment1.add(SC.N.START_NODE);
7.      //find succeed nodes
8.      tempNode= findNextNode(ACS.N.START_NODE);
9.      //do while if did not reach end node and not find joint node or forking node
10.     while(tempNode!= SC.N.END_NODE && ifFoundSplittedNode==false)
11.     {
12.         //if current node is a joint node or forking node
13.         if(tempNode.nodeType==SPLITT_NODE)
14.         {
15.             ifFoundSplittedNode=true;
16.             //find the corresponding joint node or forking node
17.             aggregationNode=findAggregationNode(tempNode);
18.             //find the aggregation node's succeed nodes
19.             nextNodes=findNextNode(tempNode);
20.             //for each path after the aggregation node, do splitting recursively
21.             //all split segments will be take as sub-segments of segment2
22.             for each node in nextNodes
23.                 segment2.add(decomposite(getSubSC(SC,
24.                                 node,aggregationNode)));
25.             end for each
26.             // for each path between the aggregation node and END node, do splitting
27.             // recursively, all split segments will be take as sub-segments of segment3
28.             segment3=decomposite(getSubSC(SC,
29.                     aggregationNode,SC.N.END_NODE))
30.         }
31.         segment1.add(tempNode);
32.     }
33.     SSC.add(segment1);
34.     SSC.add(segment2);
35.     SSC.add(segment3);
36.     return SSC;
37. }
```

This algorithm splits service chain recursively. Firstly, the algorithm will separate service chain into three segments according to first forking node and last joint node that are pair nodes. The first segment starts form START node and end with first forking node. The third segment starts from last joint node and end with END node. And the remaining belongs to second segment. Because there could be joint and forking nodes in second and third segment, so it is need to continue splitting second and third segment until there is no

joint and forking node exists. All segments produced by the algorithm are ordinal relation and any atom service can only stay in one segment. Previous service chain decomposition algorithm presented in paper [23] produce all of paths that start from START node and end with END node, which could cause that same atom service exist in different segments and make it difficult to compute over all QoS value of service chain. Although the research in paper [24] could produce segments that do not overlap with each other, but the relationship between segments is still too complicate to compute QoS value.

3.2.2. QoS Evaluation Method for Service Chain on Single Index

The QoS evaluation of service chain on single index is the second step to compute overall QoS value on all indexes. The evaluation procedure is affected by structure of service chain and characteristic of index. After splitting, the influence of structure can be ignored and only the characteristic of index needs to be considered. For example, the response time of service chain shown in Figure 1 is the sum of all segments' and the reliability can be multiplied all segments'. In the light of how the QoS index affects QoS evaluation result, QoS index can be classified as three types.

1) Cumulated type

The overall QoS performance on this type of QoS indexes can be cumulated from atom services, such as price and response time.

$$Q_{ct}(SC) = \sum_{i=1}^{n} Q_{ct}(S_i) \tag{2}$$

where $Q_{ct}(SC)$ denotes the overall QoS performance of service chain on index of cumulated type and $Q_{ct}(S_i)$ is the ith atom service's QoS value on index of cumulated type.

2) Multiplying Type

The overall QoS performance on this type of QoS indexes can be multiplied from atom services, such as accessibility, availability and reliability.

$$Q_{mt}(SC) = \prod_{i=1}^{n} Q_{mt}(S_i) \tag{3}$$

3) Extremal Type

The overall QoS performance on this type of QoS indexes depends on the best or worst atom service such as reputation, confidence and interestingness.

$$Q_{et}(SC) = \min \mid \max(Q_{et}(S_i)), 1 \le i \le n \tag{4}$$

3.2.3. QoS Evaluation Method for Service Chain on Multiple Indexes

The final step is to produce the QoS value of service chain on all indexes. After the processing of step 2, a service chain can be viewed as an atom service with multiple QoS indexes. Hence we can use the same method presented in section 3.1 to evaluate the performance of service chain. Here, qos_i^{best} denotes the best performance that the service chain can achieve by calculate directly from atom service and without considering the influence of structure. qos_i denotes QoS value of service chain on ith index. Here we present an example to demonstrate the evaluation method.

4. Data Volume Aware MOACO Algorithm

Multiple objective ant colony optimization algorithm (MOACO) is a proven technique to conquer multi-objective optimal problem that developed from basic ant colony optimization algorithm. In this section, we present an improved MOACO algorithm named as DVA-MOACO to optimize the performance of service chain by leading ants to march into paths that can reduce data transferring cost.

4.1. The Improved Rule of State Transition

The basic idea of ACO algorithm is to select path from which have been visited by ants and have greater possibility. The paths with greater possibility will attract more ants to travel to and accordingly more phenomenons will be left on these paths. In ACO, the choice possibility of path is named as state transition possibility. Therefore, the definition of rule of state transition will decide how ants perform routing. In order to describe how ants consider data volume as selecting path, the following two definitions are given:

1) Data transferring cost, referred as DTC. DTC describes the cost of transferring data between two nodes that depends on the data volume (DV) and transmission speed (TS). DTC can be calculated by

$$DTC_{ij} = f(DV_{ij}, TS_{ij}) = \frac{DV_{ij}}{TS_{ij}} \tag{5}$$

2) The best DTC of edge, referred as DTC_{ij}^{best}. Assume that $e_{ij} = \{AS_i, AS_j\}$ is a directed edge of service chain, where AS_i and AS_j are abstract nodes. Suppose that the size of data to be transferred on e_{ij} is DV_{ij}, the candidate service set of AS_i is $\{S_{i1}, S_{i2}, ..S_{in}\}$ and the candidate service set of AS_j is $\{S_{j1}, S_{j2}, ..S_{jm}\}$, then DTC_{ij}^{best} can be calculated by

$$DTC_{ij}^{best} = \min(\frac{DV_{ij}}{TS_{ik,jl}}) \tag{6}$$

where $TS_{in,jm}$ denotes the transmission speed of atom service S_{ik} and S_{jl} $(1 \le k \le n, 1 \le l \le m)$, which decided by distance and network state between atom services. Obviously, DTC_{ij}^{best} refers to the DTC of pair atom services with shorter distance and fasted network transfer capability.

The overall DTC and DTC^{best} of the service chain is the sum of DTC and DTC_{ij}^{best} of all edges respectively. But the overall DTC^{best} usually cannot be reached for that there is other QoS constraints need to be satisfied simultaneously. Therefore, the rule of state transition shall be adjusted to make ant colony try to choose paths that the overall DTC could be close to DTC^{best} as far as possible and have better QoS performance.

In service chain, there are some edges will suffer more heavy data transfer cost than other edges. The ant colony shall pay more attention to DTC than other QoS indexes when travel over these edges. In order to measure which path will be affected more greatly, the following formula is utilized,

$$\omega_{ij} = f(DTC_{i,j}) = \frac{DTC_{i,j}}{\sum DTC_{i,j}}, 0 \le i, j \le |V| \tag{7}$$

where $|V|$ is the number of nodes contained in service chain, ω_{ij} is a numerical value between 0 and 1 and denotes the weight of $path_{ij}$ according to DTC. The greater of ω_{ij} means heavier DTC on $path_{ij}$ than other paths.

Base on formula (7), the state transition rule can be adjusted as following formula,

$$p_{ij}^{k}(t)=\begin{cases}0, if \quad node_j \in tabu_k \\ e^{-\omega_{ij}} \bullet \left(\dfrac{[\tau_{ij}(t)]^{\alpha} \bullet [\eta_{ij}(t)]^{\beta}}{\sum_{s \subset (nodes-tabu_k)} [\tau_{is}(t)]^{\alpha} \bullet [\eta_{is}(t)]^{\beta}}\right)^{1-\omega_{ij}}, if \quad node_j \in (nodes-tabu_k)\end{cases} \tag{8}$$

where $p_{ij}^{k}(t)$ t. $tabu_k$ are nodes which have been visited by ant and can not be reached from $node_i$ where current ant stay. $\tau_{ij}(t)$ is the pheromone density on $path_{ij}$ at time t and it is determined by how many ants have traveled on this path and updated by following method,

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{9}$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{10}$$

In formula (9) and (10), ρ is volatilization rate of pheromone and $(1-\rho)$ is residual rate of pheromone that can prevent a large amount of pheromone that may accumulate on some paths. $\Delta\tau_{ij}(t)$ is the increment of pheromone and accumulated from all ants that travel on $path_{ij}$ at time t. $\Delta\tau_{ij}^{k}(t)$ is the increment of pheromone on $path_{ij}$ laid by kth ant at time t,

$$\Delta\tau_{ij}^{k}(t)=\frac{\theta}{QoS_{ij} * \omega_{ij}} \tag{11}$$

Where θ is a constant; QoS_{ij} is the QoS performance of $node_i$ and $node_j$ that associate to $path_{ij}$ and calculated by formula (1). According to the definition of formula (1) that the lower of QoS_{ij} means the better performance, hence the formula (11) indicates that ants will lay more pheromone if they found better path.

$\eta_{ij}(t)$ is the heuristic information decided by QoS performance of path which found by kth ant at current loop. It can be calculated as $(QoS_{ij})^{-1}$. α is a constant and reflects the importance of history experience of ants. β is another constant to represent the importance of heuristic information. If α is greater than β, the ants will be inclined to choose path with more pheromone (history experience); In contrast, ants will prefer the path with better QoS performance for the moment.

In reference to formula (8) the state transition rule is now determined by the product of DTC and QoS performance. In case of some paths fall under greater DTC, ants will decrease the effect of QoS and focus on DTC

4.2 The Implementation of DVA-MOACO Algorithm

To optimize multiple objects simultaneously, DVA-MOACO algorithm will utilize $(m+1)$ ant colonies where m equals the number of objects and each ant colony has unique pheromone matrix. Each of the m ant colonies performs optimization aims to one of the m objects respectively so that each object has the chance to be optimized. The last one ant colony synthesizes the outcome of m ant colonies and use the method previously mentioned to do optimization on all of m objects.

Because each of the m ant colonies only focuses on one object, so that ants in these colonies will take the idea of single object optimization to choose path. In this case, these ant colonies will use following method to perform state transition,

$$p_{ij}^{k}(t)=\begin{cases}0, if \quad node_j \in tabu_k \\ \dfrac{[\tau_{ij}(t)]^{\alpha}\bullet[\eta_{ij}(t)]^{\beta}}{\sum_{s\subset(nodes-tabu_k)}[\tau_{is}(t)]^{\alpha}\bullet[\eta_{is}(t)]^{\beta}}, if \quad node_j \in (nodes-tabu_k)\end{cases} \tag{12}$$

In each of ant colonies, a list will be set up to store solutions that found in each loop. The solutions would be descendent sorted according to the value of single QoS index. At the end of each loop, algorithm will select the best solution and update its pheromone matrix of corresponding path. The pheromone update method is the same as which depicted in formula (7). Hence the algorithm exploring solutions space will consider particular QoS index as well as the overall QoS indexes, which will lead ants to try more potential solutions and avoid the algorithm falling into premature convergence.

The $(m+1)th$ ant colony is different with the other ant colonies in that it will optimize all QoS indexes at the same time. This ant colony does not equip its own pheromone matrix but randomly choose one of the pheromone matrixes of the m ant colonies to compute the state transition probability. At the end of each loop, the $(m+1)th$ ant colony will evaluate the quality of solutions by method described in section 3 and choose the best one to update pheromone matrix on corresponding path. When the algorithm reach the end condition, the top solutions obtained by $(m+1)th$ ant colony will be returned as the output of algorithm.

The main steps of DVA-MOACO algorithm is depicted as follows:

- Step 1, Initializing input parameters for algorithm. Assume that the input parameters are $\{N, E, AS, Q\}$, where N is the nodes set, E is the edges set, AS is the abstract service chain, which defines the structure, and Q is the QoS indexes set. The number of optimization objects is $m=|Q|$ and the number of ant colonies is $AntC_{num}=m+1$. Let the maximum loop number of algorithm is $Iter_{\max}$ and the current loop number is $Iter=0$. Each ant colony is allocated K ants and initiated pheromone density of all edges to be a const, where K is a number greater than $|N|$. Finally, the number of solutions wished to be return by algorithm is set as TOP.
- Step 2, ant colonies start the loop to send out ants to travel on service chain. The upper bound of loop number is $Iter_{\max}$. Each of the m ant colonies is allocated a particular QoS index respectively.
- Step 3, for each of ants in an ant colony, they begin search path independently.

- Step 4, In order to make sure every ant will visit every edge of E, the algorithm sets up a stack for each ant to record the forking node that have been visited by ant. For kth ant, the stack is marked as $stack_k$. Assume that current node that the kth ant staying is $node_i$. If $node_i$ is a forking node, then algorithm will push $node_i$ into $stack_k$ and the ant will randomly select a branch to travel; if $node_i$ is a join node and there are other branches haven't visited, then the ant will move to the node which is the top element of $stack_k$; if all branches have been visited, the top element will be removed from $stack_k$.
- Step 5, Assume the next node that the kth ant shall move to is $node_j$, this ant will select candidate service for $node_j$ by formula (12).
- Step 6, the kth ant move to $node_j$ and add $node_j$ to $tabu_k$.
- Step 7, if there are nodes have not been visited, then the algorithm goes to Step 4, otherwise it go to Step 8.
- Step 8, if all edges have been travelled by all ants then the pheromone will be updated according to the single QoS index that belongs to the colony; otherwise, algorithm goes to step 3.
- Step 9, the $(m+1)th$ ant colony randomly choose one of the m ant colonies' pheromone matrix and repeat the process from step 3 to step 8. Meanwhile, the formula (12) in step 5 shall be change to formula (8). The best solution obtain in each loop of $(m+1)th$ ant colony will be stored in a list named as $solution_{m+1}$.
- Step 10, if the number of loop has reached $Iter_{\max}$ then algorithm will terminate and return the TOP solutions in $solution_{m+1}$ as result; otherwise, algorithm goes to step 2.

5. Algorithm Evaluation

Here, we use a service chain with five nodes to be the test instance as shown in Figure 3. Four QoS indexes are considered which are availability, accessibility, reliability and response time. We randomly generate a number that greater than 0 and less than 1 for each service instance's QoS indexes. Meanwhile, a matrix to record the DTC between any two adjacent service instances is randomly produced, in which the DTC is a number greater than 0 and less than 1.

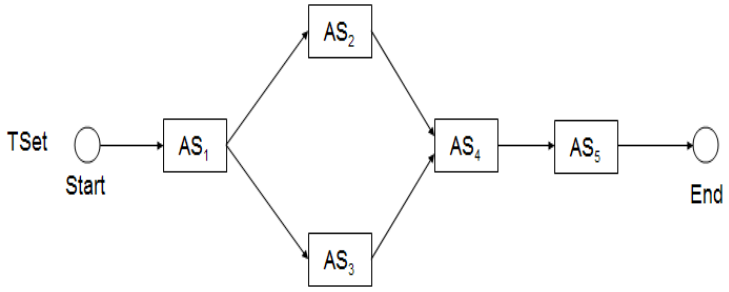

Figure 2. The Service Chain as Test Instance

The test environment is a PC with following configuration: CPU Intel i5 2.53GHz; Memory 2048M; and OS Windows XP.

We use different size of candidate services pool to compare the performance and effect of ESA (Exhaustive Search Algorithm) and DVA-MOACO algorithm. The parameters of DVA-MOACO algorithm are described in Table 2.

Table 2. Parameter Setting for DVA-MOACO Algorithm

Parameter	Symbol	Value
Number of ant colonies	M	4
Number of ants in ant colony	K	6
Weight of history experience	α	1
Weight of heuristic information	β	2
Pheromone evaporation rate	ρ	0.5

We conducted five tests on the conditions that the size of candidate services pool for each atom service is 10, 20, 30, 40 and 50 respectively. The result is depicted in table Figure 4-6.

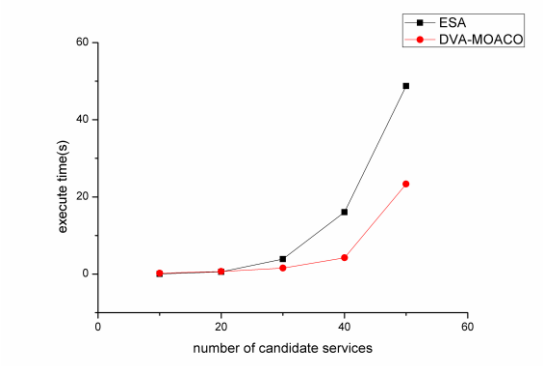

Figure 4. Comparison of Average Running Time

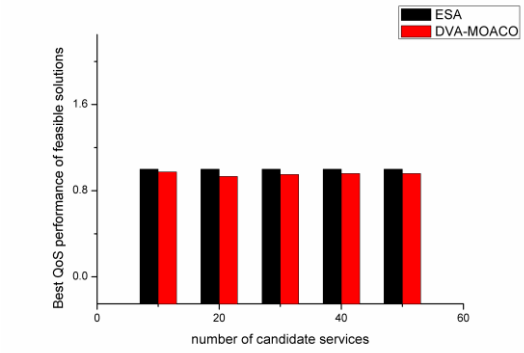

Figure 5. Comparison of QoS

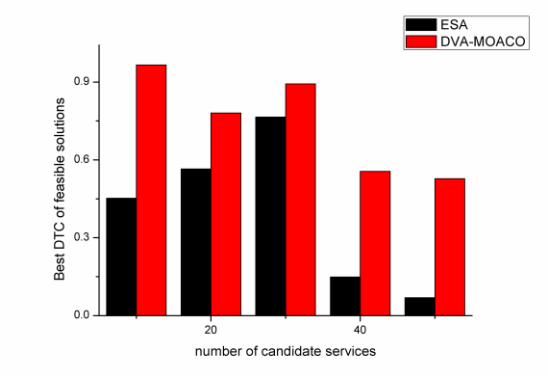

Figure 6. Comparison of DTC

Average Execution Time- As we can see in Figure 4, ESA algorithm uses less time to find solutions when the size of candidate service pool is relative smaller, which are 10 and 20. It is because that the convergence speed is slower than searching directly. As the expansion of candidate service pool, DVA-MOACO algorithm shows the advantage of speed while the searching time of ESA algorithm increased significantly.

Quality- As depicted in Figure 5 and Figure 6, ESA algorithm uses the brute force strategy to explore every potential solution and compare their quality so that it can find the best solution in case the solution space is relative small. In the five test cases, ESA algorithm always can get the best solution with the best QoS performance. But due to only considering QoS performance, the solutions produced by ESA algorithm cannot guarantee the better DTC. On the contrary, although DVA-MOACO algorithm may not reach the best QoS performance but they can cut down DTC remarkably. Taken together, the quality of solutions produced by DVA-MOACO algorithm is better than which produced by ESA algorithm.

6. Conclusions

In this paper, we proposed a new approach called DVA-MOACO for geographical knowledge cloud service composition that inspired by ant colony optimization method. DVA-MOACO aims to produce feasible service chain with lower data transfer cost under multi-QoS constraint. DVA-MOACO algorithm newly introduces a QoS evaluation model that can assess the overall QoS performance of service chain with complex structure. More important, DVA-MOACO algorithm has the promising capability of self-adaptive path searching that the paths with heavy data transfer cost will be avoided. The simulation result illustrates that DVA-MOACO algorithm can find feasible solutions with better QoS performance and lower data transfer cost.

Acknowledgement

This work is supported partly by the National Science Foundation of China under Grants 30972299, 41001203 and 41071267, and by Fujian Sci. & Tech. program under Grant 2010I0008 and 2010HZ0004-1, and by EC FP7-2009-People-IRSES under Grant 247608.

References

[1] W. Xiaozhu, C. Chongcheng, L. Jianfeng, *et al*. “GeoKSCloud：Motivation, Design and Application”, The Journal of Geo-Information Science, vol. 16, no. 2, **(2014)**, pp. 273-28.
[2] K. Ren, J. Song, M. Zhu and N. Xiao. “A bargaining-driven global QoS adjustment approach for optimizing service composition execution path”. The Journal of Supercomputing, **(2011)**, pp. 1-24.
[3] R T Marler, J S Arora, “Survey of multi-objective optimization methods for engineering[J]”, Structural and multidisciplinary optimization. vol. 26, no. 6, **(2004)**, pp. 369-395

[4] K. Ren, J. Song, M.Zhu and N. Xiao. "A bargaining-driven global QoS adjustment approach for optimizing service composition execution path". The Journal of Supercomputing. **(2011)**, pp. 1-24
[5] T. Stützle, "Parallelization strategies for ant colony optimization[C]", Parallel Problem Solving from Nature—PPSN V. Springer Berlin Heidelberg, **(1998)**, pp. 722-731.
[6] R. Jovanovic, M. Tuba, D. Simian, "Comparison of different topologies for island-based multi-colony ant algorithms for the minimum weight vertex cover problem[J]", WSEAS Transactions on Computers, vol. 9, no. 1, **(2010)**, pp. 83-92.
[7] Y. S. Luo, Y. Qi, D.Hou, L. F Shen, Y. Chen, & X. Zhong, "A novel heuristic algorithm for QoS-aware end-to-end service composition", Computer Communications, vol. 34, no. 9, **(2011)**, pp. 1137-1144
[8] K. Adrian, I. Fuyuki, H. Shinichi. "Efficient heuristic approach with improved time complexity for QoS-aware service composition". 2011 IEEE 9th International Conference on Web Services, ICWS 2011,**, (2011)**, pp. 436-443
[9] H. Lican, Z. Xiaocen, H. Yinxiu, W. Gaoxuan, "A Qos optimization for intelligent and dynamic web service composition based on improved PSO algorithm. 2nd International Conference on Networking and Distributed Computing", ICNDC 2011, **(2011)**, pp. 214-217.
[10] M. Tang, L. Ai. "A Hybrid Genetic Algorithm for the Optimal Constrained Web Service Selection Problem in Web Service Composition". 2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010**(2010)**.
[11] C-C Huang, W-Y Liang, W-C Lee, Y-H Lai, "Constrained evolutionary computing approach to Web service compositions", International Journal of Systems Science, vol. 42, no. 10, **(2011)**, pp. 1625-1638
[12] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, T. David, & V. Acretoaie, "Ant-Inspired Technique for Automatic Web Service Composition and Selection[C,// Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010 12th International Symposium on. IEEE, **(2010)**, pp. 449 – 455.
[13] A. Forestiero, C. Mastroianni, G. Papuzzo, G. Spezzano. "A Proximity-Based Self-Organizing Framework for Service Composition and Discovery". Proceeding of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, **(2010)**.
[14] ZJ Wang, ZZ Liu, XF Zhou, YS Lou, "An approach for composite Web service selection based on DGQoS[J]", The International Journal of Advanced Manu-facturing Technology, vol. 56, **(2011)** pp. 1-13.
[15] X. Pennec, J. Montagnat, T. Glatard, "Efficient services composition for grid-enabled data-intensive applications[C]", // High-Performance Distributed Computing, International Symposium on. IEEE. **(2006)**, pp.333-334.
[16] M. Brambilla, S Comai, P Fraternali, I Manolescu, S Ceri. Model-driven specification of web services composition and integration with data-intensive web applications[J]. IEEE Data Eng Bull, vol. 25, **(2002)**, pp. 53-59.
[17] O. Ezenwoye, S. Busi, S M Sadjadi, "DYNAMICALLY RECONFIGURABLE DATA-INTENSIVE SERVICE COMPOSITION[J]", Webist, **(2010)**, pp. 125-130.
[18] L Wang, J Shen, J Luo, F Dong. An Improved Genetic Algorithm for Cost-Effective Data-Intensive Service Composition[C]// 2013 Ninth International Conference on Semantics, Knowledge and Grids (SKG). IEEE Computer Society, **(2013)**, pp. 105-112.
[19] A. Bucchiarone, L. Presti, "QoS Composition of Services for Data-Intensive Application[C]", // Internet and Web Applications and Services, 2007. ICIW '07. Second International Conference on. IEEE, **(2007)**, pp. 46-54.
[20] Y. Zhang, Y. Zhou , Y. Gao, "Optimizing the data intensive mediator-based Web services composition[J]", Frontiers of Www Research & Development Apweb Proceedings, **(2006)**, 3841:pp. 708-713.
[21] Y. Yu, H Ma, M. Zhang, "A Hybrid GP-Tabu Approach to QoS-Aware Data Intensive Web Service Composition[J]", Lecture Notes in Computer Science, **(2014)**, pp. 106-118.
[22] Y. Li, C. Lin, "QoS-Aware Service Composition for Workflow-Based Data-Intensive Applications[C]",// 2011 IEEE International Conference on Web Services. IEEE Computer Society, **(2011)**, pp. 452-459.
[23] T. Yu, Y. Zhang, K J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints[J]", ACM Transactions on the Web (TWEB), vol. 1, no. 1, **(2007)**: 6.
[24] W Zhang, CK Chang, T Feng, HY Jiang. QoS-based dynamic web service composition with ant colony optimization[C]. Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual. IEEE, **(2010)**, pp. 493-502.
[25] J. Dean, S. Ghemawat, "MapReduce: simplified data processing on large clusters [J]", Communications of the ACM, vol. 51, no. 1, **(2008)**, pp. 107-113.

Authors

Xiaozhu Wu, He received his Ph.D. degree in communication and information system from Fuzhou University in 2014. He is currently a lecture in college of mathematics and computer science at Fuzhou University, China. His fields of research are focused on service computing, knowledge engineering. (wxz@fzu.edu.cn).

Chen Chongcheng, He got his PhD degree in Cartography & GIS from The institute of Geography & Resource, Chinese Academy of Science, China in 2000. He has more than 15 years of experience of research and education in field of geoinformatics and applications. He is a full-time professor at Fuzhou University.His research interests include spatial data mining & geographical knowledge grid/cloud, spatial decision system, and geo-visualization & virtual geographical environment. He has coauthored more than 150 referred conferences and journal papers. He is currently leading his research group to develop an applicable geographical knowledge grid/cloud platform – GeoKSGrid.He is organizing an International Knowledge Cloud Consortium (IKCC) with the mission of leading Medusa Knowledge Cloud design and development, an innovative initiative for developing intelligent and advanced cloud applications.

