On the Dynamic Ant Colony Algorithm Optimization Based on Multi-pheromones

Ya-mei XIA, Jun-liang CHEN, Xiang-wu MENG
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts & Telecommunications, Beijing 100876, P.R. China
xiayamei@gmail.com, chjl@bupt.edu.cn, mengxw@bupt.edu.cnl

Abstract

In this paper, an algorithm DACO (Dynamic Ant Colony Optimization Algorithm Based on Multi-pheromones) is put forward to apply to the dynamics of web services state and QoS in service composition optimization. In order to denote users' needs more accurately, this algorithm sets multiple pheromones. The DACO is also improved based on experiment in order to make it better and faster converge to optimization value. Simulation experiment in this paper shows that the DACO is more effective than Ant Colony Algorithm and a Genetic Algorithm applied to services composition.

1. Introduction

Optimization has been a popular research topic for decades. Particle Swarm Optimization (PSO) is now established as an efficient optimization algorithm for static functions in a variety of contexts [1]. With the dynamic optimization problems appearing, the PSO is also applied in dynamic optimization [2]. The dynamic optimization problems have been changing over time, which causes changes in the position of optima as well as the characteristics of the search space. This leads to the fact that existing optima may disappear, while new optima may appear. Optimization under the dynamic environments is a challenging task that attracts great attention.

The Ant Colony Optimization (ACO) algorithm, brought forward by Italian scholars in 1990s, is a kind of heuristic random optimization algorithm through the imitation of ant's food seeking behaviors, and the optimization value is obtained in it by the pheromones updating mechanism [3-5]. Years' researches show that, with such merits as distributing computing, easy to collaborate with other algorithms, robust, etc., ACO is especially effective in finding better value when used in composition optimization [6,7].

ACO is another global optimization algorithm succeeding the Genetic Algorithm. Compared with Genetic Algorithm, ACO is simpler, with less parameter and no such operations as intersecting and variation, but with better optimization result. After being advanced, ACO has witnessed rapid development and been applied to many composition optimization fields as an excellent choice, such as TSP, optimization PID control, parameter assessment of chaos coefficient, test set optimization, neural network, artificial intelligence, signal transmit, rule learning etc. But there exist inherent limitations in ACO, such as slow convergence, tendency to stagnancy, etc. To overcome the above limitations, many researchers have put forward many improvement methods. [8-15]

Ant Colony Algorithm is introduced in this paper to promote the effectiveness and quality of services composition, which proves its value and significance theoretically and practically.

The semantic web promises to bring automation to the areas of web service discovery, composition and invocation. In order to realize these benefits, rich semantic descriptions of web services must be created, and the technology of ontology is one of such description tools. In this paper the technology of ontology is applied to services composition. The approaches used in services composition include: semantics matching approach based on interface, services composition approach based on model driving, services composition approach based on services ontology, etc. Recently, many other new web services composition approaches are put forward. Lin Lin and Ping Lin have presented an overview of BPEL as an orchestration language for web services, and CCXML and SCXML as orchestration languages for call control and other similar contexts. BPEL, CCXML, and SCXML can be complementary tools in an

978-0-7695-3131-1/08 $25.00 © 2008 IEEE
DOI 10.1109/ICIS.2008.112

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 16,2010 at 11:43:25 UTC from IEEE Xplore. Restrictions apply.

orchestration toolkit and this can be used to solve real-time communications, including telecom applications and converged telecom-data applications [16]. John T. E. Timm and Gerald C. Gannod present a model-driven architecture based approach for specifying semantic web services composition through the use of a UML profile that extends class and activity diagrams [17]. Ruth & Tu and Shen & Su have also made profound analysis into web services [18, 19].

It is evident that services composition technology tends to be more dynamic. In dynamic services composition, it is necessary to choose Web services time after time to ensure that the composed services can meet users' needs. Currently when web services are becoming more abundant, users are tending to require individualized services, not only asking for more functions of composed services, but also for the effectiveness and preferences. At the same time, the QoS attribute provided by different services with the same function can differ to a large degree. As a result, selecting services is becoming especially important. In addition, the optimization efficiency of services composition is decreasing sharply because of the increasing of web services amount, which calls on effective methods to optimize the services composition.

The second part of this paper makes a model of services composition optimization, the third part puts forward the DACO, the fourth part makes an simulation experiment to verify the feasibility and validity of the algorithm, and the result of the experiment is compared with that of a genetic algorithm applied in service composition optimization. The fifth part of the paper gives a conclusion and a tentative plan for further research.

2. Model Building

We will firstly generate demonstrations according to different services composition approaches. The definitions of the services composition demonstration are as follow:

Definition1: The services composition graph is one direction and simple connected graph, each path in the graph denotes a complete services composition, each node denotes a services orchestration, each arch represents a service, the beginning point denotes the input of the services composition, and the ending point denotes the output of the service composition.

Definition2: The services composition graph is with no isolated point and suspension point, and except beginning point and ending point, all the in-degree and out-degree of each node is ≥ 1. There is no loop in the graph, and at least one path from the beginning point to the ending point. Refer to Figure 1.

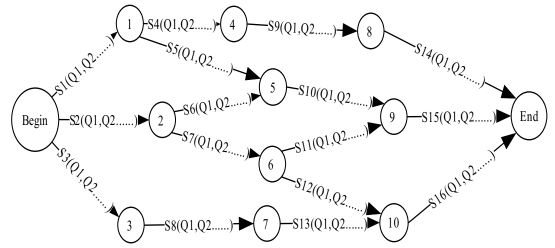

Figure 1. Services Composition Graph

In addition, a service has multiple QoS, and each person has a different preference weight on each QoS. In order to express user's preference more accurately and really, the possible preference of a service is decomposed into many sub preferences of QoS, and the preference values of each QoS will be obtained respectively, after which the preferences of multiple QoS will be further compounded to express users' preference to this service. Different QoS have different attribute, as exemplified by the following attribute:

'>' attribute, which means that the bigger the QoS value, the greater the preference tendency. For example, the bigger the service bandwidth, the greater the users' preference.

'<' attribute, which means that the smaller the QoS value, the greater the preference tendency. For example, other things being equal, the lower the consumption price, the greater the users' preference tendency.

'=' attribute, which means that the closer the QoS value to the preference constraint value, the greater the preference level. For example, the user will prefer to choose those consumption services that he can afford.

Section attribute means that when the chosen value is in a certain scope, the user will show the preference. For example, when traveling, if someone doesn't like night train, he will choose the train in the section of [8:00, 22:00], which means that time section is from 8:00 in the morning to 22:00 at night.

We will define the operation of different attribute type, p_i denotes the preference of q_i .which is as follows:

'>' attribute operation: $p_i = \frac{q_i - l}{range}$

'<' attribute operation: $p_i = \frac{l - q_i}{range}$

l is the preference critical value of the user, and *range* is the range of corresponding QoS.

'=' attribute operation: if the value is equal to the preference critical value of user, $p_i = 1$, else $p_i = 0$.

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 16,2010 at 11:43:25 UTC from IEEE Xplore. Restrictions apply.

Section attribute operation: if the value is between the two critical values, $p_i = 1$, else $p_i = 0$.

Definition 3: The obtaining of a service preference is divided into must-be-satisfied QoS attribute and may-be-satisfied QoS attributes. Set the user's preference to a service being decomposed into QoS attribute as $\{ Q_1, Q_2, \cdots, Q_n \}$, and among them $\{ Q_1, Q_2, \cdots, Q_k \}$ is must-be-satisfied preference, and $\{ Q_{k+1}, Q_{k+2}, \cdots, Q_n \}$ is may-be-satisfied preference. This information is obtained from user's preference. Set service set is $\{ s_1, s_2, \cdots, s_m \}$, and for s_i $(i = 1, \cdots, m)$ the corresponding QoS value is $\{ q_{i1}, q_{i2}, \cdots, q_{in} \}$, $\{ q_{i1}, q_{i2}, \cdots, q_{ik} \}$ is must-be-satisfied preference, and $\{ q_{ik+1}, q_{ik+2}, \cdots, q_{in} \}$ is may-be-satisfied preference. Set new services set is A and A is null. The process of service selecting is:

```
For services { $s_1, s_2, \cdots, s_m$ }
  For set { $q_{i1}, q_{i2}, \cdots, q_{ik}$ }
    If $\forall q_{ij}$ satisfy( $q_{ij}$ , $Q_j$ )
      then put $s_i$ to A;
      Else abandon $s_i$ ;
```

satisfy(q_{ij} , Q_j) is defined as satisfaction comparing rule according to the operation definition of attribute of q_{ij} . If *satisfy*(q_{ij} , Q_j) is true, then the q_{ij} quality of service s_i satisfy the users' preference of Q_j , else the q_{ij} quality of service s_i doesn't satisfy the users' preference of Q_j , service s_i can't satisfy users' preference, and in this condition we abandon service s_i .

After the above operation, if A is not null, there exists such service that satisfy must-be-satisfied preference. Set the number of services in A is t , and set A as $\{ s_1, s_2, \cdots, s_t \}$.

```
For services { $s_1, s_2, \cdots, s_t$ }
  For set { $q_{ik+1}, q_{ik+2}, \cdots, q_{in}$ }
    $p_{ij} = p(q_{ij})$
   $ss_i = c(p_{ij}, w_j)$
s = MaxSatisfying( $ss_1, ss_2, \cdots, ss_t$ );
```

$p(q_{ij})$ denotes the rule by which the preference of q_{ij} is got according to the operation definition of attribute of q_{ij} . $c(p_{ij}, w_j)$ is to compound the preference of service s_i . *MaxSatisfying*($ss_1, ss_2, \cdots, ss_t$) denotes the rule by which we got the max satisfaction service of s whose value is the maximum in $\{ ss_1, ss_2, \cdots, ss_t \}$.

3. The Dynamic Ant Colony Algorithm Optimization Based on Multi-pheromone (DACO)

There is only one kind of pheromone in ACO, which cannot deal with the question of multiple attributes in web services composition; in ACO, the path and path weight is stable, and cannot fit for dynamic web services composition. Additionally, ACO chiefly makes use of positive feedback to enhance relative optimization. In this method, when evolving to a certain iterative, the increasing enhancement of the pheromone on local optimization paths makes the ants gather on fewer paths, and prematurity and stagnancy will appear. In this case, the optimization value is only local. Aiming at addressing the above disadvantages, the DACO is put forward to fit for the dynamic services composition optimization and to promote the algorithm's effectiveness.

Web services state is random and instable and the QoS attributes of the service will change sometimes, so in the process of services composition optimization some cases should be considered, such as uselessness of the services, increase of new services, and QoS change. These questions can be solved by deleting an arch, adding an arch or changing the weight of the arch in the services composition graph. Adding an arch must meet such requirements as not forming loop. When services are useless, the corresponding arch need to be deleted, which is a more complicated case, for the deleting of one arch may lead to the emerging of suspension points in the graph.

3.1 Pheromone Strategy

As a service has many QoS attributes, we set various pheromones to denote different QoS attributes. Set a service has n QoS attributes $Q_1, Q_2, \ldots, Q_n$, and correspondingly set n pheromones to denote n QoS attributes, so the updating rule of the k th pheromone in (i, j) arch on the time of $t + n$ is as follows:

$$Q_{kij}(t+n) = (1-\rho)Q_{kij}(t) + \rho \Delta Q_{kij}(t+n)$$

$Q_{kij}(t)$ denotes the value of the k th pheromone Q_k on time t in the arch of (i, j) ; $\Delta Q_{kij}(t+n)$ denotes the k th pheromone Q_k on time of $t + n$ of the service (i, j) being felt; ρ denotes updating actor of the pheromone; 1- ρ denotes remained actor of the pheromone. In the above case, ρ denotes the degree that the service uses the newly-experienced Q_k value, and 1- ρ denotes the Q_k value experienced before.

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 16,2010 at 11:43:25 UTC from IEEE Xplore. Restrictions apply.

From the above pheromone updating rules, we can find that pheromones can make dynamic adjustment according to the changes of QoS attributes.

3.2 Heuristic Actor

For the services composition optimization in the case when QoS value is not clear, the weight of the optimization attribute is obtained subsequently by running DACO, and then the weight is evaluated to corresponding service. After choosing a service, each ant will use the experience of new service QoS to update the weight of the service to act as the heuristic actor of this service in the next round of algorithm. In the beginning of the algorithm's running, we set the heuristic actor as 1 in order that it won't affect the result of the formulas of the pheromone updating rule.

3.3 Algorithm Improvement Strategy

Owing to the inherent limitations in ACO, such as convergent to local optimization, slow convergence etc., we use the following improvement rules to advance the performance of the ACO.

Rule 1: After a round of optimization, we compare the weights of all paths, only updating the pheromone of the path with maximum pheromone. The experiment proves that the convergence rate and the probability of finding optimization of the DACO have increased greatly.

Rule 2: In order to avoid prematurity, stagnancy and converging to local optimization of the algorithm, we set variable L as the optimization path list and l as the length of the list. The value of l is defined by optimization dimensions. In optimization proceeding, the l maximum pheromone paths generated by comparing with the maximum pheromone path in each round are saved. When the algorithm reaches maximum round count, we compare the maximum pheromone path with the paths in L, compute the QoS of the corresponding services again according to the method defined in the second part and get the QoS services composition with higher preference as optimized findings. We need to add two points, one is that all above QoS is compared in real time, another is that setting L as a list owes to the services states of uselessness and QoS variety.

3.4 DACO Algorithm Flow

Input: Services composition list waiting to be selected.

Output: Max satisfaction services composition.

1) We set variable $N_{c\max}$ as max round amount, m as ants' amount, t as initialized time slice and t =0, N_c as cycle control variable and N_c =0, k as ant cycle control variable and k =0, the taboo table as $tabu_k(k=1,2,\cdots,m)$, the path table as $path_k$ $(k=1,2,\cdots,m)$ and max optimization paths list is L, etc. In the beginning we place ants on the beginning point and set the pheromone variable $\tau_{ij}(t)=const$. The value of $const$ is set according to the principle that has no effect on the succeeding algorithm running.

2) The cycle control variable $N_c = N_{c+1}$, if $N_c \succ N_{c\max}$, then the running quit cycle, jump to step 11) and end.

3) The ant count variable $k=k+1$, if $k \succ m$ then the running quit k cycle and jump to step 8).

4) Find the paths that connect with the node that the ant k is located, compute transferring probabilities according to the formula of state transferring probability defined in ACO, and choose the next path starting from this node.

5) Add the new node selected into $tabu_k$. Get the value of variable $q_1,q_2,\cdots,q_n$ of the service and compare it with the limitation value of $Q_1,Q_2,\ldots,Q_n$ from users' preference ontology according to definition 3 in section 2.

Add the new node selected into $path_k$, place ant on new node and wait for next selecting.

If there exists some QoS value that doesn't satisfy the limitation, give up this services, select another path and add 1 to the dissatisfying degree of this service. If the dissatisfying degree is greater than the value set, delete this service, move ant to the direction of the beginning point, place ant on first node that is not deleted, and at the same time, delete corresponding path from $path_k$. If other path from the first node that is not deleted to other nodes does not exist, continue to move the ant to the direction of the beginning point until arriving at a node for which another path exists, and then place the ant on that node, beginning service selecting again.

6) If the new node selected is the ending point in the service composition graph, this ant's services selection is over. The running will jump to step 7), else jump to step 4).

7) Summarize and record the preference value of every service selected by this ant in this round according to definition 3.

8) Compare the sum of the preference of all the ants' composed services in this round, get the most satisfying service composition, and at the same time update the pheromone on this path.

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 16,2010 at 11:43:25 UTC from IEEE Xplore. Restrictions apply.

9) The most satisfying service composition path obtained in step 8) will be compared with the original paths in L , and then new max satisfying service composition paths will be obtained.

10) Empty $tabu_k$ and $path_k$, and jump to step 2).

11）Compare the service composition path that has the max pheromone with paths in L, and get the most satisfying service according to definition 3 as the optimized findings.

4. Simulation and Performance Analysis

Gerardo Canfora etc. have put forward an approach for QoS aware service composition based on Genetic Algorithms. In their empirical study, the optimization case is constrained on cost and response time and the workflow contains 18 invocations of 8 distinct abstract services and have a cyclomatic complexity 5. For each abstract service there is a variable (average) number of available concrete services, i.e., 5, 10, 15, 20, 25. [20]

In the "Tourism Service Recommendation System of Beijing Pinggu District", we make the application simulation for the algorithm put forward in this paper. The simulation application system is built to evaluate the performance of DACO. In the simulation system, web services and the value of QoS attributes of services in the services ontology are generated by the random number generating algorithm. In addition, the simulation system can generate the services composition graph synchronously according to the process of services composition, and can get all candidate services parameters of each concrete service.

Because of the randomness of services composition process, the number of abstract services and the number of concrete services are also random. But in order to make the comparing easy and convenient, we limit the number of abstract services as 8. The number of concrete services corresponding to each abstract service is defined at random from 0 to 25.

The hardware environment of the simulation is: Intel Pentium IV CPU with 2.8 GHz, 512M memory; the operating system is Windows XP, and the chief realizing tool of the simulation system is Visual C++ 6.0.

Experiment data: The services composition graph contains 18 invocations of 8 distinct abstract services. Each abstract service with the same function in the graph includes 0-25 concrete candidate services waiting to be selected. In order to decrease errors, the test is executed repeatedly for 10 times, and the results are averaged to estimate the performance.

Firstly, in every test, by using linear programming we get the service with the max preference from concrete services for every abstract service dynamically then apply DACO to make global optimization. The performance test figures are as follows.

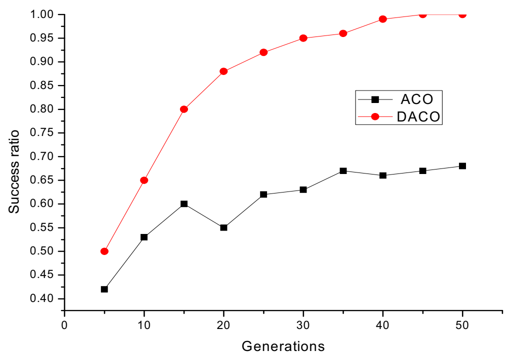

Figure 2. Performance test of ACO and DACO (containing multiple QoS)

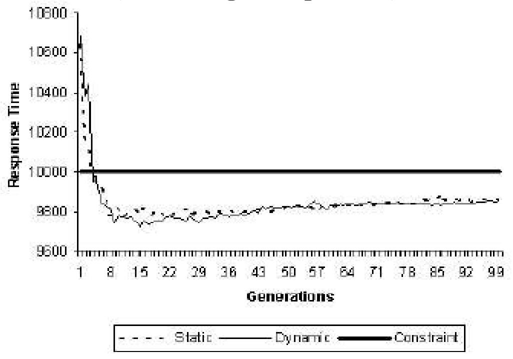

Figure 3. Performance test of GA (response time)

From Figure 2 we can find that in the case that α =2, β =2, and the ant number is 1, when the iterative times is 50, the algorithm converges to the optimized value. When getting the service with the max preference from the concrete service set, the operation time is equal to service number. Here we do the computing as the max service number 25. As the ant number is 1, 1 generation in DACO is equal to 1 generation in GA. We can see that the fitness of GA go to stabilization when generation times are 100 (Here we only give the performance test figure of response time). 25+50=75<100, so the performance of DACO is better than that of GA in [20].

5. Conclusion

This paper makes a research of ACO applied in the services composition optimization system, and put forward an improved DACO which can address such problems in the optimization process such as dynamics,

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 16,2010 at 11:43:25 UTC from IEEE Xplore. Restrictions apply.

instability, multiple QoS attribute limitations of web services. The effectiveness of the algorithm is verified in a "tourism service recommendation system", with the result that the DACO has better performance than typical Ant Colony Algorithm and the Genetic Algorithms applied to services composition. Hopefully, we will make a research that applies ACO to semantic web to make the improved algorithm individualized.

*Supported by the National Natural Science Foundation of China under Grant No. 60432010; the National Grand Fundamental Research 973 Program of China under Grant No. 2007CB307100

6. References

[1] K. Parsopoulos and M. Vrahatis. "Recent approaches to global optimization problems through particle swarm optimization," *Natural Comput.*, vol. 1, no. 2–3, 2002, pp. 235–306.

[2] Tim Blackwell and Jürgen Branke. "Multiswarms, Exclusion, and Anti-Convergence in Dynamic Environments", *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 4, 2006.

[3] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies. In: Varela F, Bourgine P, eds. *Proc. of the ECAL'91 European Conf. of Artificial Life*. Paris: Elsevier, 1991. 134~144.

[4] Dortgo M, Maniezzo V, Colorni A. "Ant system: Optimization by a colony cooperating Agents", *IEEE Trans. on Systems, Man, and Cybernetics*, Part B: Cybernetics, 1996, 26(1): 29~41.

[5] Dortgo M, Gambardella L. M. "Ant colony system: A cooperative learning approach to the traveling salesman problem", *IEEE Trans. on Evolutionary Computation*, 1997, 1(l): 53~66.

[6] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Gener. Comput. Syst*., vol. 16, no. 8, 2000, pp. 851-871.

[7] X. Wang, X. Z. Gao, and S. J. Ovaska. "A Hybrid Optimization Algorithm Based on Ant Colony and Immune Principles", *International Journal of Computer Science & Applications*, Vol. 4 Issue 3, 2007, pp 30-44.

[8] PENG Pei-fu, LIN Ya-ping, HU Bin, ZHANG Gui-fang. "Optimal PID Control of Self-Adapted Ant Colony Algorithm Based on Genetic Gene". *Acta Electronica Sinica,* Vol.34, No.6, June 2006.

[9] Junzhong Ji, Ning Zhang, Chunnian Liu. An Ant Colony Optimization Algorithm for Learning Classification Rules. *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 2006.

[10] SUN Yan, MA Hua-dong, LIU Liang. An Ant-Colony Optimization Based Service Aware Routing Algorithm for Multimedia Sensor Networks. *Acta Electronica Sinica,* Vol.35 No.4 Apr. 2007.

[11] LI Ling-Zhi, ZHENG Hong-Yuan, DING Qiu-lin. "An Anycast Routing Based on Improved Ant Colony Algorithm", *Journal of Electronics & Information Technology,* Feb. 2007.

[12] Stutzle T, Hoos HH. "MAX-MIN ant system and local search for the traveling salesman problem", In: *IEEE Int'l Conf. on Evolutionary Computation*. Indianapolis: IEEE Press, 1997, 309~314.

[13] SUN Li-Juan1, WANG Ru-Chuan. Solving QoS Multicast Routing Problem Based on the Combination of Ant Colony Algorithm and Genetic Algorithm. *Acta Electronica Sinica,* Vol.34, No.8, Aug. 2006.

[14] HUANG Han, HAO Zhi-Feng, WU Chun-Guo, QIN Yong. "The Convergence Speed of Ant Colony Optimization", *Chinese Journal of Computers*, Vol.30 No.8 Aug. 2007.

[15] ZHAO Bao-Jiang, LI Shi-Yong. "Convergence Analysis of a Class of Adaptive Ant Colony Algorithm". *Proceedings of the 6th World Congress on Intelligent Control and Automation*, IEEE 2006, pp.3524-3527.

[16] Lin Lin and Ping Lin. "Orchestration in Web Services and Real-Time Communications". Web Services in Telecommunications, *IEEE Communications Magazine*, 2007.

[17] John T. E. Timm and Gerald C. Gannod. "Specifying Semantic Web Service Compositions Using UML and OCL". *WWW 2007.*

[18] Michael Ruth & Shengru Tu. "Towards Automating Regression Test Selection for Web Services". *WWW 2007.*

[19] Zhongnan Shen & Jianwen Su. "On Automated Composition for Web Services". *WWW 2007.*

[20] Gerardo Canfora, Massimiliano Di Penta, etc. "An Approach for QoSaware Service Composition Based on Genetic Algorithms". *GECCO'05*, 2005.

Authorized licensed use limited to: University of Johannesburg. Downloaded on June 16,2010 at 11:43:25 UTC from IEEE Xplore. Restrictions apply.