A Multi-objective Ant Colony Optimization algorithm for Web Service Instance Selection

Fang Qiqing¹, a, Hu Yamin², b, Lv Shujun³, c, Zhou Fen⁴, d and Hu Yahui⁵,e*

¹Air Force Early Warning Academy, China
²fangqiqing@yeah.net, b hym2100@163.com, c sayonlvshujun@126.com, d 296813225@qq.com, e hyh5800@163.com

* hyh5800@163.com

Keywords: Multi-Objective, Web Services, Global Optimization.

Abstract. In this paper, we present a multi-objective Web Services selection algorithm, which aims to achieve the dynamic web service composition. Firstly, we analyze multi-objective optimization problems with user constraints. Furthermore, we optimize the different QoS parameters of the workflow. And experimental results prove our algorithm is effective for solving the Web Services selection problem.

Introduction

Dynamic composition of Web Services[1] is the key technology of SOA implementation, and Web Services selection is an important issue in dynamic Web Service composition. However, the existing approaches[2,3,4,5,6] on service selection of dynamic Web Service composition are almost QoS local optimization or mono-objective based, and can not resolve the problem of Web Services selection with QoS global optimization and multi-objective.

With the meta-heuristic proach[7] and multi-objective ant colony optimization[8], we formulate the global service selection problem, describe a method based on multi-objective ant colony optimization, and compare the global optimized approach with the MOGA.

Problem Formulation

We assume that an Abstract Service Plan with sequential services process \( AP =< WS_{T1}, WS_{T2}, ..., WS_{Tn} > \) is generated automatically by an AI planner or manually. So we are not concerned about the compatibility issue among services but only focus on the QoS service selection problem.

Assuming that there are total \( n \) steps in the Abstract Service Plan, each service type has \( M \) instances and each Web Service Instance has three QoS parameters, \( Qos =< Cost, Time, Reliability > \), where Cost is the price of the service instance. Time is the time taken to deliver services between service requestors and providers. Reliability represents the ability of a Web Service to perform its required functions under stated conditions for a specified time interval:

Cost: \( \text{Cost}(P) = \sum_{n \in P} \text{Cost}(n) \) (1)

Time: \( \text{Time}(P) = \sum_{n \in P} \text{Time}(n) \) (2)

Reliability: \( \text{Reliability}(P) = \prod_{n \in P} \text{Reliability}(n) \) (3)

Multi-objective Ant Colony Optimization algorithm for Web Service Instance Selection

Our Web Service Instance Selection Algorithm is base on the generalization of the Multi-objective Ant Colony Optimization algorithm (MOACO) [8]. This approach uses a colony of ants for the
construction of \( m \) solutions \( P \) at every generation. Then, the known Pareto Front \( P_{\text{known}} \) is updated, including all non-dominate solutions. Finally, the pheromone matrix \( \tau_{ij} \) is updated. We modified it to solve the Web Service Instances selection problem, as shown in algorithm 1.

**Precedure MOACO4WS**

- initialize \( S, D, N_r, \varphi \)
- initialize \( \tau_{ij} \)
- while stop criterion is not verified
  - repeat-for \( k = 1 \) to \( m \)
    - **Construct Solution** \( P \) (See Figure 4)
    - if \( \{ P \mid P \in P_{\text{known}} \} \) then
      \[ P_{\text{known}} = P_{\text{known}} \cup \{ P \mid P \rightarrow P, \forall P, P \in P_{\text{known}} \} \]
    - end-if
  - end-repeat-for
  - Update of \( \tau_{ij} \)
- end-while

**Algorithm 1. General Procedure of MOACO**

**Experiment**

The experiments results is shown in Tab. 1. The average number of concrete workflow solutions of each algorithm that are in \( P_{\text{apr}} \), denote as \( \mid P_{\text{apr}} \mid \). The set of solutions that are dominated by \( P_{\text{apr}} \) is denoted as \( \{ P_{\text{apr}} \} \). The number of found solutions is \( \mid P_{\text{alg}} \mid \) and the percentage of solutions present in \( P_{\text{apr}} \) is \( \% ( \in P_{\text{apr}} ) \).

<table>
<thead>
<tr>
<th>Test Group</th>
<th>Iterations</th>
<th>Algorithm</th>
<th>( \in P_{\text{apr}} )</th>
<th>( P_{\text{apr}} )</th>
<th>( P_{\text{apr}} )</th>
<th>( P_{\text{alg}} )</th>
<th>( % ( \in P_{\text{apr}} ) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group1</td>
<td>100</td>
<td>( P_{\text{ACO}} )</td>
<td>14.5</td>
<td>0</td>
<td>14.5</td>
<td>97%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>( P_{\text{ACO}} )</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( P_{\text{GA}} )</td>
<td>13.8</td>
<td>0</td>
<td>13.8</td>
<td>92%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( P_{\text{GA}} )</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Group2</td>
<td>100</td>
<td>( P_{\text{ACO}} )</td>
<td>22.1</td>
<td>4.2</td>
<td>26.3</td>
<td>91%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>( P_{\text{ACO}} )</td>
<td>23.3</td>
<td>5.1</td>
<td>28.4</td>
<td>97%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( P_{\text{GA}} )</td>
<td>17.3</td>
<td>1</td>
<td>18.3</td>
<td>72%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( P_{\text{GA}} )</td>
<td>21</td>
<td>7.3</td>
<td>28.3</td>
<td>88%</td>
<td></td>
</tr>
<tr>
<td>Group3</td>
<td>100</td>
<td>( P_{\text{ACO}} )</td>
<td>30.1</td>
<td>3.3</td>
<td>33.4</td>
<td>84%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>( P_{\text{ACO}} )</td>
<td>26.8</td>
<td>6.4</td>
<td>33.2</td>
<td>74%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( P_{\text{GA}} )</td>
<td>33</td>
<td>2.1</td>
<td>35.1</td>
<td>92%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( P_{\text{GA}} )</td>
<td>29.3</td>
<td>7.3</td>
<td>36.6</td>
<td>81%</td>
<td></td>
</tr>
<tr>
<td>Group4</td>
<td>100</td>
<td>( P_{\text{ACO}} )</td>
<td>41</td>
<td>6.3</td>
<td>47.3</td>
<td>80%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>( P_{\text{ACO}} )</td>
<td>45.2</td>
<td>7.1</td>
<td>52.3</td>
<td>89%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( P_{\text{GA}} )</td>
<td>33</td>
<td>9.4</td>
<td>42.4</td>
<td>64%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( P_{\text{GA}} )</td>
<td>38.7</td>
<td>12.3</td>
<td>51</td>
<td>76%</td>
<td></td>
</tr>
</tbody>
</table>

From the above table, we can see:
(1) In Group1, when the iteration number is 200, both the founded solutions of MOACO and 
MOGA are almost belong to $P_{apr}$. But when the iteration number is 100, MOACO overcoming
MOGA.

(2) For a larger number of Web Service Instance and Web Service type (Group2-Group4), the 
MOACO also demonstrated to be better than the MOGA. In fact, MOACO obtained a larger number 
of solutions belonging to $P_{apr}$, for all run times.

Also, the running time between our algorithm and that the genetic algorithm is compared in this 
experiment. The time for finding out the optimal concrete workflows is showed in Fig.1. It can be seen 
that the time astringency of algorithm proposed in this paper is better than the MOGA.

![Figure 1: The comparison of execution time](image)

Conclusion

A global optimization and multi-objective Web Services selection algorithm based on MOACO is 
proposed to resolve the question of multi-objective services composition optimization with QoS 
constraints for dynamic Web Service Composition. Considering the presented experimental results, 
MOACO is able to find more best solutions than the recently published QoS Global Optimization 
Based on Multi-objective Genetic Algorithm (MOGA). Experimental results also indicate the 
feasibility and efficiency of the algorithm.

References

provisioning of dynamic Web services, Proc. of the 18th Int’l Conf. on Data Engineering, IEEE 


