2015

Facilitating an ant colony algorithm for multi-objective data-intensive service provisions

Lijuan Wang  
*University Of Wollongong, lw840@uowmail.edu.au*

Jun Shen  
*University of Wollongong, jshen@uow.edu.au*

Junzhou Luo  
*Southeast University*

Publication Details  
Facilitating an ant colony algorithm for multi-objective data-intensive service provisions

Abstract
The explosion of enormous sources of digital data has led to greater dependence on data-intensive services. Applications based on data-intensive services have become one of the most challenging applications in cloud computing. The service provision, and in particular service composition, will face new challenges as the services and data grow. In this paper, we will evaluate an ant colony system to resolve the multi-objective data-intensive service composition problem. The algorithm for a multi-objective context will get a set of Pareto-optimal solutions considering two objectives at the same time: the total cost and the total execution time of a composite service.

Disciplines
Engineering | Science and Technology Studies

Publication Details

This journal article is available at Research Online: http://ro.uow.edu.au/eispapers/3212
Facilitating an Ant Colony Algorithm for Multi-Objective Data-Intensive Service Provision

Lijuan Wang, Jun Shen, Junzhou Luo

*School of Information Systems and Technology, University of Wollongong, NSW, Australia
+School of Computer Science and Engineering, Southeast University, Nanjing, P. R. China

Abstract

The explosion of enormous sources of digital data has led to greater dependence on data-intensive services. Applications based on data-intensive services have become one of the most challenging applications in cloud computing. The service provision, and in particular service composition, will face new challenges as the services and data grow. In this paper, we will evaluate an ant colony system to resolve the multi-objective data-intensive service composition problem. The algorithm for a multi-objective context will get a set of Pareto-optimal solutions considering two objectives at the same time: the total cost and the total execution time of a composite service.

Keywords: Ant colony system algorithm, genetic algorithm, multi-objective optimization, data-intensive service composition, cloud computing, quality of service

1. Introduction

In recent years, the data generated by scientific activities, social networking, social media, as well as commercial applications have increased exponentially. This explosion of digital data has led to greater dependence on data-intensive services. As a result, applications based on data-intensive services have become one of the most challenging applications in service oriented computing and cloud computing. The scope, number, and complexity of data-intensive services are all set to soar in the future. However, service provision, and in particular service composition, will face new challenges such as autonomy, scalability, adaptability, and robustness. Indeed, new mechanisms are needed to overcome those issues.

One of the motivations of our work is the Alpha Magnetic Spectrometer (AMS) experiment, which uses cloud computing to process a huge amount of data. The AMS, also designated AMS-02, is a particle physics experiment module that is mounted on the International Space Station. The purpose of the AMS experiment is to advance knowledge of the universe and lead to the understanding of its origin by searching for antimatter and dark matter while performing precision measurements of cosmic rays composition and flux. The AMS-02 SOC (Science Operation Center) at Southeast University in China (labeled as AMS-02 SOC@SEU) is supported by the IBM-sponsored Cloud

*Corresponding author

Email addresses: lw840@uowmail.edu.au (Lijuan Wang), jshen@uow.edu.au (Jun Shen), jluo@seu.edu.cn (Junzhou Luo)

Preprint submitted to Journal of Computer and System Sciences
Computing Center with 3500 CPU core and 500TB storage. The AMS-02 SOC@SEU typically receives 200GB of data from AMS and generates 700GB of data after processing them, on each day.

Scientists and remote users deploy different processes, such as data mining, image processing, or data query on a large amount of data at AMS-02 SOC. The use of Web services technologies provides valuable solutions to speed up the scientific data analysis. A composition of a set of services as a composite service can be reused by other researchers. The cost and response time of each service in the data-intensive service composition are critical for the composite service. The data-intensive service composition has the following special challenges. First, the large number of data sets and increases of functionally equivalent services make the composition complex. Second, the size and the number of distributed data sets increase the communication and storage cost, which affect the performance of the whole application process. Third, the cost of transferring data to and from service platforms increases as the number of data sets increases. Fourth, the dynamic nature of cloud computing and data replication needs a dynamic and adaptive mechanism to regulate the interaction between data and service users and providers.

This paper proposes a multi-objective ant colony system (MOACS) for data-intensive service composition. We focus on the scalability and adaptability of service composition, and particular attention is paid to multi-objective optimization related to cost and quality of service (QoS) attributes. The other above mentioned challenges had been solved in our earlier work [22, 24]. To the best of our knowledge, our work is the first application of the ACS meta-heuristic to the data-intensive service composition problem with global QoS constraints, where both overall cost and execution time need to be minimized. Also, our work gives a comprehensive comparison between MOACS and multi-objective genetic algorithm (MOGA), which is lacking in the literature.

The main contributions of this paper are the following: first, we propose a multi-objective ant colony system for data-intensive service composition with global QoS constraints; second, to evaluate our proposed algorithm, we conduct experiments from many different scenarios with respect to five performance metrics; third, we present a comprehensive comparison of the proposed algorithm with a multi-objective genetic algorithm. The lessons learned from our experimental results are that, when we have a large number of concrete services available for each abstract service, a multi-objective genetic algorithm can achieve better solutions. On the other hand, whenever the number of concrete services available is small, such as in some simple and repetitive scientific computation, a multi-objective ant colony system is to be preferred to a multi-objective genetic algorithm.

The remainder of this paper is organized follows: Section 2 reviews related work. Section 3 specifies the multi-objective service composition problem with global QoS constraints. Section 4 further investigates how a multi-objective ant colony system could be used to solve the problem. Section 5 presents experimental results and analysis. Finally, Section 6 concludes this paper and proposes future work.
2. Related work

In the context of Web service composition, abstract services are the functional descriptions of services, and concrete services represent the existing services available for potential invocation of their functionalities and capabilities. When the functions of several concrete services are consistent with the functional description of an abstract service, these concrete services are the service candidates for the abstract service and QoS attributes are used to distinguish them. Web service composition composes the existing Web services into a new service to accomplish workflow tasks. Web service selection is an important part of Web service composition.

We have been applying bio-inspired algorithms to tackle this type of problem [12, 13, 17–24, 27]. In [12, 13, 27], we investigated how ant colony optimization (ACO) algorithms were used for peer selection in service composition. In [23], we presented a survey on bio-inspired algorithms for Web service composition. In [20], we designed an ant colony system to solve the single-objective data-intensive service composition problem. We also proposed genetic algorithms to solve the same problem [18, 22]. Comparisons with the mixed integer programming (MIP) method and the random selection approaches showed the scalability and effectiveness of our algorithms. In [17], we designed a data replica selection algorithm based on ACS, which was used to select the best replicas in order to reduce the response time and cost of services. In [21], we designed four strategies to modify the pheromone information in order to adapt ACS to handle the dynamic scenarios where new services were provided, some services were discontinued, or the QoS attributes of some services were changed. We also proposed an ant-inspired negotiation approach based on the above dynamic ACS, and designed a multi-phase, multi-party negotiation protocol [19, 24].

A variety of studies adopted ACO algorithms to solve service composition problems. The authors of [7] used one pheromone matrix in their MOACO algorithm, and the heuristic information was the value of a function of all objectives. Based on this study, the paper [16] adopted the chaos variable and presented a chaos MOACO algorithm. The chaos variable was used in order to overcome the problem of low efficiency and partial optimization that ant colony algorithm brought. On the other hand, the paper [25] investigated a dynamic ACO algorithm with multi-pheromone to optimize service composition. They set various pheromones to denote different QoS attributes. In [28], the QoS-based Web service composition problem was modeled as a multi-objective optimization problem, and a $k$-tuple pheromone was used to represent $k$ objectives. The authors of [29] modeled the service selection problem as a graph with AND/OR vertices and proposed the ant clone rule, which was applied when ants were at an AND vertex.

The comparison of our MOACS with other studies [2, 3, 10, 11, 15, 25, 26, 28, 29] is shown in Table 1. The studies listed in Table 1 used ACO algorithms or GA to solve a similar problem to the one described in this paper. The contributions of our work which differentiate it from these studies are the following.

1. Compared with the studies [3, 10, 11, 28], the sequential, parallel, and conditional structures are considered in the composition structures in our work.
2. Our algorithm is suitable to resolve the service selection problem with global QoS constraints, which is different from the studies [2, 3, 11, 26, 28].
### Table 1: The novel contributions of our work compared with other studies

<table>
<thead>
<tr>
<th>Articles</th>
<th>Composition structure</th>
<th>Global QoS constraints</th>
<th>Performance metric</th>
<th>Method of evaluation</th>
<th>Test scenarios</th>
<th>Outcomes of the experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2]</td>
<td>Complex structures</td>
<td>None</td>
<td>Applicability</td>
<td>None</td>
<td>Single scenario</td>
<td>The applicability of the multi-objective evolutionary algorithm for service composition</td>
</tr>
<tr>
<td>[3]</td>
<td>Sequential</td>
<td>None</td>
<td>The number of non-dominated solutions</td>
<td>None</td>
<td>Two test groups</td>
<td>The results obtained from the NSGA-II validated the feasibility of the proposed service selection approach</td>
</tr>
<tr>
<td>[10]</td>
<td>Sequential</td>
<td>Considered</td>
<td>The number of non-dominated solutions</td>
<td>None</td>
<td>Two scenarios</td>
<td>The results indicated that NSGA-II was fit for solving the service composition problem</td>
</tr>
<tr>
<td>[11]</td>
<td>Sequential</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>The proposed MOGA was not simulated with experiments</td>
</tr>
<tr>
<td>[15]</td>
<td>Complex structures</td>
<td>Considered</td>
<td>The quality of solutions</td>
<td>Comparison with NSGA-II</td>
<td>A workflow consisted of four abstract services, and each abstract services was associated with three concrete services</td>
<td>The performance of the proposed algorithms outperformed the linear programming method and the NSGA-II</td>
</tr>
<tr>
<td>[25]</td>
<td>Complex structures</td>
<td>Considered</td>
<td>Computation time</td>
<td>Comparison with GA and ACO</td>
<td>Varying number of concrete services</td>
<td>Dynamic ACO had better performance than ACO and GA</td>
</tr>
<tr>
<td>[26]</td>
<td>Complex structures</td>
<td>None</td>
<td>The fitness value</td>
<td>Comparison with GA</td>
<td>A workflow consisted of ten tasks and each task has ten services</td>
<td>NSGA-II had quicker convergence than GA</td>
</tr>
<tr>
<td>[28]</td>
<td>Sequential</td>
<td>None</td>
<td>Computation time</td>
<td>Comparison with MOGA</td>
<td>Ten scenarios</td>
<td>multi-objective ant colony optimization algorithm could find near-optimal solutions and was scalable</td>
</tr>
<tr>
<td>[29]</td>
<td>Complex structures</td>
<td>Considered</td>
<td>Computation time</td>
<td>Comparison with exhaustive searching method</td>
<td>Varying number of concrete services</td>
<td>ACS algorithm had higher convergence speed and needed fewer iterations</td>
</tr>
</tbody>
</table>

This paper

- Complex structures

- Considered

- Five performance metrics

- Comparison with MOGA

- 15 different scenarios

- When a large number of concrete services are available for each abstract service, MOGA can achieve better solutions in a reasonable time. On the other hand, whenever the number of concrete services available is small, MOACS should be preferred.

3. In our work, all objectives share the same pheromone trails and all the objectives have the same importance, in this case, a set of Pareto-optimal solutions are found. Only single solutions were given by other studies [2, 25, 29].

4. We adopted five performance metrics to measure the Pareto-optimal solutions, which is different from all the other studies. One performance metric cannot adequately measure the performance of multi-objective optimization algorithms, as explained in Section 5.

5. In order to verify the solutions obtained using the proposed algorithm, we implemented a MOGA. The MOGA is based on the improved version of the non-dominated sorting genetic algorithm (NSGA-II) [5], but we need to customize and modify the NSGA-II in order to handle our problem.

### 3. Multi-Objective Data-Intensive Service Composition Problem with Global QoS Constraints

The goal of the majority of existing multi-objective optimization algorithms is to find Pareto-optimal solutions. The concept of dominance is used to relate the solutions found in these algorithms. In a minimization problem for all objectives, a solution $x_1$ dominates another solution $x_2$ (denotes as $x_1 < x_2$) if and only if the two following...
conditions are true: 1) $x_1$ is no worse than $x_2$ in all objectives, namely, $F_i(x_1) \leq F_i(x_2)$ ($i \in \{1, 2, \ldots, N\}$, $N$ is the number of objective functions), and 2) $x_1$ is strictly better than $x_2$ in at least one objective, namely, $F_j(x_1) < F_j(x_2)$ ($\exists j \in \{1, 2, \ldots, N\}$). In a minimization problem for all objectives, a solution $x_1$ is said to cover another solution $x_2$ (denotes as $x_1 \preceq x_2$) if one of the two following conditions is true: 1) $x_1$ dominates $x_2$, or 2) $x_1$ is equal to $x_2$ in all objectives. Among a set of solutions, the non-dominated set of solutions are those that are not dominated by any member of the set. A solution is said to be Pareto-optimal if it is not dominated by any other possible solution. Thus, the Pareto-optimal solutions to a multi-objective optimization problem form the Pareto front or Pareto-optimal set [14]. Pareto-optimal sets are the solutions that cannot be improved in one objective function without deteriorating their performance in at least one of the remaining objective functions.

In the service composition, a graph is used to represent the dependencies between services. Fig. 1 presents an example of a graph in which data sets, as the inputs and outputs of services, are incorporated. The data-intensive service composition problem with global QoS constraints (DISCP_GQoSC) is an extension of the service composition problem, denoted as $G = \{V, E, D, start, end\}$ and is mathematically stated as:

Minimize an objective function $F$, given:

1. $V = \{AS_1, AS_2, \ldots, AS_n\}$ represents the set of $n$ abstract services, and $start$ and $end$ represent two virtual tasks;
2. $cs_i = \{cs_{i,1}, cs_{i,2}, \ldots, cs_{i,m}\}$ is the service candidate set of $AS_i$, which includes all concrete services to implement $AS_i$;
3. $q_{cs_{i,j}} = [q_{cs_{i,j}}^1, q_{cs_{i,j}}^2, \ldots, q_{cs_{i,j}}^r]$ with $r$ QoS parameters is the QoS vector of concrete service $cs_{i,j}$;
4. $E$ represents the edges of the graph, which includes all links between concrete services of any two connected service candidate sets;
5. $D = \{d_1, d_2, \ldots, d_z\}$ represents a set of $z$ data servers;
6. $DT^i = \{dt^1, dt^2, \ldots, dt^k\}$ represents a set of $k$ data sets which are required by abstract service $AS_i$, and these data sets are distributed on a subset of $D$;
7. $Q_c = [Q_1^c, Q_2^c, \ldots, Q_r^c]$ ($1 \leq u \leq r$) represents a set of global QoS constraints, which define requirements regarding the aggregated QoS values of the requested composite service.

In a traditional service composition problem, a single objective function $F$ may be chosen from any of the following
ones:

1. Inverse of overall utility

\[ F_1 = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} (U(cs_{i,j})x_{i,j}) \]

where \( U(cs_{i,j}) \) is the utility of concrete service \( cs_{i,j} \). The variable \( x_{i,j} \) (\( \sum_{j=1}^{m} x_{i,j} = 1 \)) represents only one concrete service is selected to implement each abstract service during the process of service composition, where \( x_{i,j} \) is set to 1 if \( cs_{i,j} \) is selected to implement abstract service \( AS_i \) and 0 otherwise.

2. Overall Cost

\[ F_2 = \sum_{i=1}^{n} \sum_{j=1}^{m} (Cost(cs_{i,j})x_{i,j}) \]

where \( Cost(cs_{i,j}) \) is the cost of concrete service \( cs_{i,j} \).

3. Overall execution time

\[ F_3 = \sum_{i=1}^{n} \sum_{j=1}^{m} (T_{et}(cs_{i,j})x_{i,j}) \]

where \( T_{et}(cs_{i,j}) \) is the execution time of concrete service \( cs_{i,j} \).

It should be noted that other composition structures and their aggregation functions can also be used in the objective functions. Here, we only list the sequential structure. For the multi-objective context of the present work, the objective function \( F \) is considered as a two-dimensional vector, considering the overall cost and execution time, with no objective considered as more important than the other. In this case, a set of Pareto-optimal solutions may be found.

4. Data-Intensive Service Selection Based on MOACS

The proposed MOACO algorithm is based on ACS, which uses a unique ant colony to simultaneously minimize all functions. All objectives share the same pheromone trails. In every iteration, an ant \( k \) (\( \forall k \in \{1,2,\ldots,Nants\} \), \( Nants \) is the number of ants) constructs one feasible solution, starting at the start vertex and successively choosing a next vertex from the set of feasible vertices \( N^k_i \) (where subindex \( i \) represents that ant \( k \) is at vertex \( s_i \), \( s_i \) is the concrete service which is chosen to implement abstract service \( AS_i \)). \( N^k_i \) is the neighborhood of vertex \( s_i \), which includes all the vertices directly connected to vertex \( s_i \) in the graph, except for the predecessor of vertex \( s_i \). For each ant \( k \), a feasible solution is found until it arrives at the end vertex. The key to MOACS for DISCP.GQoSC is how to determine the state transition rule, the local updating rule, and the global updating rule.

4.1. State Transition Rule

When ant \( k \) arrives at vertex \( s_i \), it will choose successor \( s_j \) to move to by applying the rule given by (1).

\[
\begin{cases} 
j = \underset{j \in N^k_i}{\arg \max} \{[\tau_{ij}]^{[\eta_{ij}^{\text{C}}]^{[\eta_{ij}^{T}]^{1-\lambda}]}\}, & \text{if } q \leq q_0; \\
\text{randomly selected from } N^k_i, & \text{otherwise.}
\end{cases}
\]
The variable $q$ is a random variable uniformly distributed in $[0, 1]$, $q_0$ ($0 \leq q_0 \leq 1$) is a parameter, $\lambda = k/N_{\text{ants}}$, $\beta$ weights the relative importance of the objectives with respect to the heuristic information, and $\tau_{ij}$ is the pheromone density on edge $(s_i, s_j)$. If $q > q_0$, then $j$ is randomly selected from $N_i^k$ according to the probability distribution given by (2).

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^{(\eta_{ij}^s)^{1-\beta}}}{\sum_{j' \in s^k} [\tau_{ij'}]^{(\eta_{ij'}^s)^{1-\beta}}}, & \text{if } j \in N_i^k, \\ 0, & \text{otherwise}. \end{cases} \quad (2)$$

The heuristic information considering service cost is calculated according to (3).

$$\eta_{ij}^s = 1/\text{Cost}(s_j)$$

$$\text{Cost}(s_j) = C_{\text{ac}}(s_j) + C_{\text{tr}}(s_j) + C_{\text{tr}}(s_j)$$

$$C_{\text{ac}}(s_j) = \sum_{dt \in \text{DT}^i} p_{dt}$$

$$C_{\text{tr}}(s_j) = \sum_{dt \in \text{DT}^i} T_i(dt, d_{dt}, y) \times t\text{cost}$$

$$T_i(dt, d_{dt}, y) = \text{size}(dt)/bw(d_{dt}, y)$$

The variables $C_{\text{ac}}(s_j)$ and $C_{\text{tr}}(s_j)$ represent the access cost and the transfer cost of all data sets required by $s_j$. $C_{\text{tr}}(S_j)$ is the service related cost which mainly includes the cost to provide the service and the cost to process the data sets. $p_{dt}$ represents the price of data set $dt$. For each data set $dt \in \text{DT}^i$, the time to transfer it from data server $d_{dt}$ to service platform $y$ is denoted by $T_i(dt, d_{dt}, y)$. $t\text{cost}$ is the cost of data transfer for per time unit, $bw(d_{dt}, y)$ is the network bandwidth between $d_{dt}$ and $y$, and $\text{size}(dt)/bw(d_{dt}, y)$ denotes the practical transfer time.

The heuristic information considering service execution time is calculated according to (4).

$$\eta_{ij}^s = 1/T_{\text{exec}}(s_j)$$

$$T_{\text{exec}}(s_j) = T_{\text{pf}}(s_j) + T_{\text{ac}}(s_j)$$

$$T_{\text{ac}}(s_j) = \max_{dt \in \text{DT}^i} (T_i(dt, d_{dt}, y) + T_{\text{def}}(d_{dt}) + T_{\text{ac}}(d_{dt}))$$

$$T_{\text{def}}(d_{dt}) = \text{size}(dt)/sp(d_{dt})$$

$$T_{\text{ac}}(d_{dt}) = \sum_{j=1}^{nr} (\text{size}(dt_j)/sp(d_{dt}))$$

The variables $T_{\text{pf}}(s_j)$ and $T_{\text{ac}}(s_j)$ are the time for processing data sets and the time for accessing data sets respectively. $T_{\text{ac}}(s_j)$ is the maximum value of time for accessing all data sets required by $s_j$. The storage access latency $T_{\text{def}}(d_{dt})$ is the delayed time for the storage media to serve the requests. It depends on the size of the data and the storage type. $sp(d_{dt})$ is the storage media speed. Each storage medium has many requests at the same time and it serves only one request at a time. The current request needs to wait until all requests prior to it in the queue finish, and the request waiting time is represented as $T_{\text{ac}}(d_{dt})$. The variable $nr$ is the number of data requests waiting in the queue prior to the underlying request for $dt$. 

7
4.2. Local Updating Rule

When building a solution (namely, the ants find an executed path) of the service composition problem, the ants use a local pheromone updating rule that they apply immediately after moving from vertex \(s_i\) to vertex \(s_j\) during the path construction, which is shown by (5).

\[
\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0
\]

The variable \(\xi (0 < \xi < 1)\) is a parameter and \(\tau_0\) is initially calculated by \(\tau_0 = 1/(NN \times C(S_0) \times T(S_0))\), with \(NN\) is the number of nodes in the graph, \(S_0\) is the solution generated by the nearest neighbor heuristic [6]. This is due to the fact that it is a good practice to set the initial pheromone concentration to a value that is slightly higher than the expected amount of pheromone deposited by the ants. \(C(S_0)\) represents the overall cost of the solution \(S_0\) and \(T(S_0)\) represents the overall execution time of \(S_0\).

4.3. Global Updating Rule

The global non-dominated set of solutions, which the ants found from the beginning of the trial, is stored in the Pareto-optimal set \(GP\). In each iteration, the solution found by each ant is recorded to a set \(P\). After all ants arrive at the end vertex, the non-dominated set of solutions \(LP\), are found from \(P\). Each solution in \(LP\) is compared with the solutions in \(GP\) in order to check if it is non-dominated. If it is a new Pareto-optimal solution, it is added to \(GP\) and all solutions dominated by the added one are erased from \(GP\). Since all non-dominated solutions are considered as best solutions for a multi-objective optimization problem, we suppose that all non-dominated solutions have the same quality. Therefore, for each solution \(\psi^{GP} \in GP\), the pheromone information is globally updated according to (6).

\[
\tau_{ij} = (1 - \rho)\tau_{ij} + \rho/(C(\psi^{GP}) \times T(\psi^{GP})), \quad \forall (i, j) \in \psi^{GP}
\]

The variable \(\rho (0 < \rho < 1)\) is the pheromone evaporation rate, \(C(\psi^{GP})\) is the cost of a given solution \(\psi^{GP}\) while \(T(\psi^{GP})\) is the execution time of \(\psi^{GP}\).

The implementation of our MOACS is given in Algorithm 1.

5. Experiments and analysis

In our experiments, a trial testing method is adopted to determine most suitable values for all parameters of MOACS and MOGA. Because each parameter in the two algorithms affects the performance of the algorithms, we give a range for each parameter, considering other researchers’ earlier experiments. In each algorithm, we vary one parameter and fix other parameters, then test the performance of the algorithm by 50 runs. Then we compare the average values and get the final value for the fixed parameter. Finally, the parameters of MOACS considered in this paper are: \(\beta = 2, q_0 = 0.9, \rho = 0.1, \xi = 0.1, Nants = 100\). The parameters of MOGA used in this paper are: the population is 100, the crossover probability is 0.9, and the mutation probability is \(1/n\) (where \(n\) is the number of abstract services in a composite service).
Algorithm 1 Multi-objective data-intensive service selection algorithm based on MOACS

1: step = 0; //iteration counter
2: Initialization; //MaxIt is the maximum number of iteration
3: while I do
4: step = step + 1;
5: P = ∅; // The solutions found by ants for each iteration
6: set all ants at start vertex;
7: for each ant k in Nants do
8: while ant k is not at the end vertex do
9: construct a solution using (1), (2), (3), and (4);
10: record the solution to P;
11: apply the local updating rule (5);
12: end while
13: end for
14: when all ants arrive at end vertex, find the non-dominated set LP from P;
15: update the global non-dominated set GP;
16: apply the global updating rule (6) to GP;
17: if step > MaxIt then
18: break;
19: end if
20: end while
21: output all solutions in GP.

5.1. Performance Metrics

There are two goals in a multi-objective optimization: the convergence to an approximation set (the Pareto-optimal set found in a single run) and the maintenance of diversity in solutions of the Pareto-optimal set [5]. These two goals cannot be measured adequately with one single performance metric. Meanwhile, the outcome of a multi-objective optimization run will generally consist of a varying number of non-dominated solutions. Various performance metrics to measure these two goals have been suggested [1, 4, 9, 30]. Here, the following five performance metrics were chosen: 1) the computation time, 2) the overall non-dominated vector generation (ONVG), 3) the comparison metric (C metric), 4) the size of the dominated space, and 5) the summary attainment surface. The first four metrics measure the convergence of the Pareto-optimal solutions, while the fifth metric measures the distribution of the Pareto-optimal set obtained by a multi-objective optimization algorithm.

1. The computation time, also called running time, is the length of time required to perform the algorithm.
2. The ONVG metric measures the number of distinct non-dominated solutions in the calculated Pareto-optimal set \( GP \). It is defined as \( \text{ONVG} = |GP|_c \), where \(|c|\) denotes cardinality. The larger the value of the ONVG, the more we know about the details of the Pareto-optimal set.

3. The C metric is based on comparing a pair of non-dominated sets by computing the fraction of each set that is covered by the other. \( C \) maps the ordered pair \((A, B)\) into the interval \([0, 1]\):

\[
C(A, B) = \frac{\{b \in B \mid \exists a \in A : a \preceq b\}}{|B|} \tag{7}
\]

where \(|B|\) means the number of solutions in the non-dominated set \( B \), and \( a \preceq b \) means \( a \) covers \( b \). \( C(A, B) = 1 \) means that all solutions in \( B \) are dominated by \( A \). The opposite, \( C(A, B) = 0 \) means that none of the solutions in \( B \) is dominated by \( A \). It is important to note that both \( C(A, B) \) and \( C(B, A) \) have to be considered, since \( C(A, B) \) is not necessarily equal to \( 1 - C(B, A) \). \( C(A, B) > C(B, A) \) means that the non-dominated set \( A \) has better solutions than \( B \).

4. The size of the dominated space \( S(A) \) indicates how well the Pareto-optimal set is approximated by the non-dominated set \( A \) of the algorithm [30]. For each non-dominated solution in \( A \), we can compute the values of all objective functions. These values comprise a point in the solution space. Fig. 2 illustrates an example of a dominated space, which is separated by four points \((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\). Since the optimization involves the minimization of two objectives, a reasonable maximum value for each objective (\( \text{maxI} \) and \( \text{maxII} \)) has to be chosen to determine the size of the dominated space. The greater the size of the space dominated by the non-dominated set, the closer the solutions are to the Pareto-optimal set. Here, \( PS(A) \) is used to indicate this metric.

\[
PS(A) = \frac{S(A)}{\text{maxI} \ast \text{maxII}} \ast 100\% \tag{8}
\]

where

\[
S(A) = (\text{maxI} - x_1) \ast (\text{maxII} - y_1) + (\text{maxI} - x_2) \ast (y_1 - y_2) + (\text{maxI} - x_3) \ast (y_2 - y_3) + (\text{maxI} - x_4) \ast (y_3 - y_4)
\]

Fig. 2: A space dominated by a non-dominated set for a minimization of two objectives
5. A summary attainment surface is a visual approach to summarizing a number of runs of a multi-objective optimizer. An attainment surface is a boundary in the objective space that separates those points that are dominated by or equal to at least one of the points, from those that no point dominates or equals \[8\]. The attainment surface emphasizes the achieved distribution and indicates the quality of the individual solutions. Knowles \[9\] proposed an algorithm to plot approximate summary attainment surfaces with any number of objectives, and he suggested that it was more useful to plot the median summary attainment surface to compare the performance of the optimizers. The median summary attainment surface, also called 50%-attainment surface of the optimizer, is the attainment surface on which all points are attained in exactly 50% of the runs \[8, 9\]. For a two-objective problem, the more the points of the median attainment surface of an algorithm close to the origin of the rectangular coordinate system, the better the solutions of the algorithm are.

5.2. Test Case Generation

For the purpose of our evaluation, different scenarios were considered where a composite application comprises services from \(n\) abstract services, and \(n\) varies in our experiments between 10 and 50, in increments of 10. There are \(m\) concrete services in each service candidate set, and \(m\) varies in our experiments between 10 and 100, in increments of 10. Each abstract service requires a set of \(k\) data sets, and \(k\) is fixed at 10 in our experiments. A scenario generation system is designed to generate the scenario for experiments. The system first determines a basic scenario, which includes sequence, conditional and parallel structures. With this basic scenario, other scenarios are generated by either placing an abstract service into it or adding another composition structure as substructure. This procedure continues until the scenario has the predefined number of abstract services.

For each scenario, the price of a data set, the network bandwidth (Mbps) between each data server and service endpoint, the storage media speed (Mbps), the size (MB) of a data set and the number of data requests in the waiting queue were randomly generated from the following intervals: \([1,100]\), \([1,100]\), \([1,100]\), \([1000,10000]\) and \([1,10]\). Then every scenario was performed with 21 runs (with 11 being the median line of all 21 attainment surfaces), and every run was stopped after 300 generations. All runs of the same scenario use the same data, and the average results over 21 independent runs are reported.

5.3. Result Analysis

Table 2 shows the means of the computation time of each scenario. In the upper half of Table 2, the second column indicates that the MOACS needs more computation time when the number of concrete services increases, while the third column shows the computation time of MOGA remains almost steady as the number of concrete services increases. This is because, by using the integer array coding scheme, the change in the number of concrete services will not influence the length of the genome. The computation time of both MOACS and MOGA increases when the number of abstract services increases, which is indicated by the lower half of Table 2. The upper half of Table 2 indicates that when the number of abstract services and concrete services is small, MOACS is better than
Table 2: Means of Computation Time

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>MOACS</th>
<th>MOGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$ is fixed at 10, $m$ varies between 10 and 100, in increments of 10</td>
<td>22.3333</td>
<td>29.6667</td>
</tr>
<tr>
<td></td>
<td>23.6190</td>
<td>30.0952</td>
</tr>
<tr>
<td></td>
<td>24.4762</td>
<td>29.8571</td>
</tr>
<tr>
<td></td>
<td>25.0952</td>
<td>31.3333</td>
</tr>
<tr>
<td></td>
<td>26.5238</td>
<td>29.9524</td>
</tr>
<tr>
<td></td>
<td>27.1905</td>
<td>31.2857</td>
</tr>
<tr>
<td></td>
<td>27.6667</td>
<td>31.1429</td>
</tr>
<tr>
<td></td>
<td>29.5238</td>
<td>30.9048</td>
</tr>
<tr>
<td></td>
<td>30.6667</td>
<td>30.7143</td>
</tr>
<tr>
<td></td>
<td>30.4286</td>
<td>30.2381</td>
</tr>
<tr>
<td>$m$ is fixed at 50, $n$ varies between 10 and 50, in increments of 10</td>
<td>26.5238</td>
<td>29.9524</td>
</tr>
<tr>
<td></td>
<td>61.2381</td>
<td>30.7143</td>
</tr>
<tr>
<td></td>
<td>98.9048</td>
<td>30.2381</td>
</tr>
<tr>
<td></td>
<td>141.9524</td>
<td>31.1905</td>
</tr>
<tr>
<td></td>
<td>285.5714</td>
<td>31.7143</td>
</tr>
</tbody>
</table>

MOGA since the means of the computation time of MOACS are lower than those of MOGA except in the scenario where $n = 10$ and $m = 100$. Meanwhile, the lower half of Table 2 indicates that MOGA is more scalable than MOACS when there is a large number of concrete services and abstract services.

Table 3: Means of ONVG

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>MOACS</th>
<th>MOGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$ is fixed at 10, $m$ varies between 10 and 100, in increments of 10</td>
<td>14.9048</td>
<td>14.8571</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>36.0476</td>
</tr>
<tr>
<td></td>
<td>27.4762</td>
<td>34.4286</td>
</tr>
<tr>
<td></td>
<td>22.0952</td>
<td>27.5714</td>
</tr>
<tr>
<td></td>
<td>35.8571</td>
<td>49.8045</td>
</tr>
<tr>
<td></td>
<td>25.3333</td>
<td>28.0952</td>
</tr>
<tr>
<td></td>
<td>20.1905</td>
<td>27.4286</td>
</tr>
<tr>
<td></td>
<td>38.5238</td>
<td>46.0476</td>
</tr>
<tr>
<td></td>
<td>34.8095</td>
<td>38.3333</td>
</tr>
<tr>
<td></td>
<td>15.3333</td>
<td>16.5714</td>
</tr>
<tr>
<td>$m$ is fixed at 50, $n$ varies between 10 and 50, in increments of 10</td>
<td>35.8571</td>
<td>49.8045</td>
</tr>
<tr>
<td></td>
<td>75.4762</td>
<td>69.6196</td>
</tr>
<tr>
<td></td>
<td>90.7619</td>
<td>83.1905</td>
</tr>
<tr>
<td></td>
<td>114.9524</td>
<td>82.3810</td>
</tr>
<tr>
<td></td>
<td>131.5714</td>
<td>84.1429</td>
</tr>
</tbody>
</table>

Table 3 gives the means of ONVG. By comparing the second and third column of the upper half of Table 3, we conclude that MOGA can get more non-dominated solutions than MOACS except in the scenario where $n = 10$ and $m = 10$. On the other hand, the lower half of Table 3 indicates that MOACS can find more non-dominated solutions
Table 4: Means of $PS(A)$

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>MOACS</th>
<th>MOGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$ is fixed at 10, $m$ varies between 10 and 100, in increments of 10</td>
<td>82.65%</td>
<td>82.64%</td>
</tr>
<tr>
<td>84.86%</td>
<td>84.77%</td>
<td></td>
</tr>
<tr>
<td>85.29%</td>
<td>85.25%</td>
<td></td>
</tr>
<tr>
<td>84.65%</td>
<td>84.61%</td>
<td></td>
</tr>
<tr>
<td>85.88%</td>
<td>85.87%</td>
<td></td>
</tr>
<tr>
<td>86.30%</td>
<td>86.12%</td>
<td></td>
</tr>
<tr>
<td>86.17%</td>
<td>86.02%</td>
<td></td>
</tr>
<tr>
<td>86.12%</td>
<td>85.92%</td>
<td></td>
</tr>
<tr>
<td>86.72%</td>
<td>86.38%</td>
<td></td>
</tr>
<tr>
<td>86.33%</td>
<td>86.28%</td>
<td></td>
</tr>
</tbody>
</table>

$\text{m}$ is fixed at 50, $n$ varies between 10 and 50, in increments of 10:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>85.88%</td>
<td>85.87%</td>
<td>0.9524</td>
</tr>
<tr>
<td>0.6298</td>
<td>0.6298</td>
<td>0.6037</td>
</tr>
<tr>
<td>0.8429</td>
<td>0.8429</td>
<td>0.8311</td>
</tr>
<tr>
<td>0.8311</td>
<td>0.8311</td>
<td>0.5982</td>
</tr>
<tr>
<td>0.6219</td>
<td>0.6219</td>
<td>0.5127</td>
</tr>
<tr>
<td>0.5127</td>
<td>0.5127</td>
<td>0.4523</td>
</tr>
<tr>
<td>0.4523</td>
<td>0.4523</td>
<td>0.5189</td>
</tr>
<tr>
<td>0.5189</td>
<td>0.5189</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Means of C Metric

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>$C(MOACS,MOGA)$</th>
<th>$C(MOGA,MOACS)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$ is fixed at 10, $m$ varies between 10 and 100, in increments of 10</td>
<td>0.9524</td>
<td>0.9524</td>
</tr>
<tr>
<td>0.6298</td>
<td>0.6298</td>
<td></td>
</tr>
<tr>
<td>0.6037</td>
<td>0.6037</td>
<td></td>
</tr>
<tr>
<td>0.8429</td>
<td>0.8429</td>
<td></td>
</tr>
<tr>
<td>0.8311</td>
<td>0.8311</td>
<td></td>
</tr>
<tr>
<td>0.5982</td>
<td>0.5982</td>
<td></td>
</tr>
<tr>
<td>0.6219</td>
<td>0.6219</td>
<td></td>
</tr>
<tr>
<td>0.5127</td>
<td>0.5127</td>
<td></td>
</tr>
<tr>
<td>0.4523</td>
<td>0.4523</td>
<td></td>
</tr>
<tr>
<td>0.5189</td>
<td>0.5189</td>
<td></td>
</tr>
</tbody>
</table>

$\text{m}$ is fixed at 50, $n$ varies between 10 and 50, in increments of 10:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8311</td>
<td>0.8311</td>
</tr>
<tr>
<td>0.4530</td>
<td>0.4530</td>
</tr>
<tr>
<td>0.4452</td>
<td>0.4452</td>
</tr>
<tr>
<td>0.2338</td>
<td>0.2338</td>
</tr>
<tr>
<td>0.2094</td>
<td>0.2094</td>
</tr>
</tbody>
</table>

than MOGA when the number of abstract services increases except in the scenario where $n = 10$ and $m = 50$.

Table 4 provides the means of $PS(A)$. By comparing the second and third column of Table 4, we conclude that MOACS is better than MOGA since MOACS always leads to a higher value of $PS(A)$.

Table 5 presents the means of the C metric. The value in the second column is equal to the value in the third column of Table 5. The results indicate that the convergence of the Pareto-optimal solutions of MOACS and MOGA has never been different, so we cannot say one is better than the other with respect to the C metric.
Fig. 3: Median summary attainment surface when the number of concrete services changes
Fig. 4: Median summary attainment surface when the number of abstract services changes

Fig. 3 shows the median summary attainment surface of MOACS and MOGA when the number of abstract services is fixed at 10, and the number of concrete services varies from 10 to 100, in increments of 10. Fig. 4 shows the median summary attainment surface of MOACS and MOGA when the number of concrete services is fixed at 50, and the number of abstract services varies from 10 to 50, in increments of 10.

In Fig. 3 and Fig. 4, the regions where there is no difference between the points of the median attainment surfaces of the two algorithms are indicated in gray dots, whereas those regions where the points of the two surfaces are found to be different from each other are plotted in stars and squares, respectively. In the regions where the points of the two surfaces are found to be different from each other, there are three situations: 1) if the points of the median attainment
surfaces of MOACS dominate those of MOGA, then the label MOACS is put near the points, 2) if the points of the median attainment surfaces of MOGA dominate those of MOACS, then the label MOGA is put near the points, 3) if there is no domination relationship between the points of the median attainment surfaces of MOACS and MOGA, then no label is put.

According to Fig. 3, MOACS is better than MOGA except for a small number of points of all the median attainment surfaces, since the points of the median attainment surface of MOACS are closer to the origin of the rectangular coordinate system. According to Fig. 4, both MOACS and MOGA have some points where there is no difference between them in the scenario where \( n = 10 \) and \( m = 50 \). In the scenario where \( n = 20 \) and \( m = 50 \), there are some points where MOGA is better than MOACS. In the remaining scenarios of Fig. 4, there is no domination relationship between the points of the median attainment surfaces of the two algorithms. But it is clear that MOGA generates more useful solutions than MOACS, which is indicated by the middle parts of the median attainment surfaces of both algorithms.

6. Conclusions

Data-intensive service provision faces new challenges with the rapid proliferation of services and the development of cloud computing. The range, number, and complexity of data-intensive services are all set to soar with an even more dynamic environment of services and data envisioned in the future. By a detailed analysis of the existing Web service concretization approaches, we find that it is useful for the provision of data-intensive services to explore key features and mechanisms of biological systems. The outcomes of our earlier studies also confirmed the applicability and efficiency of bio-inspired algorithms for solving data-intensive service provision issues.

This paper proposed a new multi-objective ant colony system for data-intensive service composition with global QoS constraints. The goal is to efficiently obtain a set of non-dominated solutions that simultaneously minimizes the total cost and the total execution time. In order to verify the performance of our algorithm in detail, we also implemented a multi-objective genetic algorithm. Both algorithms are simulated on many different scenarios and their performance is compared using five performance metrics. The comparison shows that the multi-objective ant colony system algorithm is preferred when the number of abstract services and the number of concrete services is small, whereas when we have a large number of concrete services available for each abstract service, the multi-objective genetic algorithm can achieve a better solution in a reasonable time. In our evaluations, we experimented with synthetic datasets without loss of generality.

Acknowledgment

This work is supported by National Key Basic Research program of China under Grants No. 2010CB328104, National Natural Science Foundation of China under Grants No. 61320106007 and No. 61202449, China National High Technology Research and Development Program under Grants No. 2013AA013503, China Specialized Research
References


