Data-Intensive Service Provision Based on Particle Swarm Optimization

Lijuan Wang [1] [*], **Jun Shen**[2]

[1] *School of Cyber Engineering, Xidian University, Xi'an, China*

[2] *School of Computing and Information Technology, University of Wollongong, Wollongong, Australia*

Received 26 June 2017

Accepted 24 November 2017

Abstract

The data-intensive service provision is characterized by the large of scale of services and data and also the high-dimensions of QoS. However, most of the existing works failed to take into account the characteristics of data-intensive services and the effect of the big data sets on the whole performance of service provision. There are many new challenges for service provision, especially in terms of autonomy, scalability, adaptability, and robustness. In this paper, we will propose a discrete particle swarm optimization algorithm to resolve the data-intensive service provision problem. To evaluate the proposed algorithm, we compared it with an ant colony optimization algorithm and a genetic algorithm with respect to three performance metrics.

Keywords: data-intensive service provision; ant colony optimization; genetic algorithm; particle swarm optimization.

1. Introduction

Data-intensive science is emerging as the fourth scientific paradigm, and new techniques and technologies for the new scientific paradigm are needed.[1] As a result, applications based on data-intensive services have become one of the most challenging applications in service oriented computing and cloud computing.[2,3] The authors of Ref. 4 presented a survey of the challenges, techniques, and technologies of data-intensive applications. For data-intensive applications, a variety of services for data mining, data storage, data placement, data replication, data transfer, and data movement have been deployed in distributed computing environments. To compose these data-intensive services will be more challenging, especially in terms of autonomy, scalability, adaptability, and robustness.

Ref. 4 proposed that bio-inspired computing was one of the potential techniques to solve the data-intensive problems. The authors stated that biological computing models were better appropriate for data-intensive problems because they had mechanisms with high-efficiency to organize, access, and process data. The authors of Ref. 5 already proved that it was useful for service management and discovery to add biological mechanisms to services. In our previous work,[6] we have presented a hierarchical taxonomy of Web service composition approaches and provided a detailed analysis of each approach. Then we found that the bio-inspired algorithms could overcome the challenging requirements of the data-intensive service provision. It is useful for the provision of data-intensive services to explore key features and mechanisms of biological systems.

In this paper, we propose a discrete particle swarm optimization(PSO) algorithm to deal with the

[*] Corresponding author, E-mail: ljwang@xidian.edu.cn

data-intensive service provision problem. Then the performance of the proposed algorithm is compared with an ant colony optimization(ACO) algorithm and a genetic algorithm(GA).

The remainder of the paper is organized as follows. In the next section, the related work is given. Section 3 introduces the data-intensive service provision problem and the particle swarm optimization algorithm. Section 4 presents the experimental settings and numerical results. Finally, section 5 concludes this paper.

2. Related Work

The process of developing a composite service is called service composition.[7] Service composition can be performed by composing either component Web services or composite services. The component Web services are developed independently by different service providers, so some services may have same functionality but differ in quality of service (QoS) attributes as well as other non-functional properties. 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. Given a request of composite service, which involves a set of abstract services and dependency relationships among them, there is a list of service candidate sets, which includes many concrete services for each abstract service. Web service selection refers to finding one service candidate to implement each abstract service according to users' requirements, which is an important part of Web service composition. For each abstract service of a composite service, the service composition process is to bind one of its corresponding concrete services and meet the constraints specified for some of the QoS attributes.[8] The final goal of the composite service construction is achieved by solving the well-known service composition problem.

In the past ten years, bio-inspired algorithms such as the ACO algorithm, the GA, and the PSO algorithm have been used to solve the Web service selection and composition problem. The ACO algorithm is a probabilistic technique proposed by Dr. Marco Dorigo in 1992 in his PhD thesis. It is widely used for solving combinatorial optimization problems which can be reduced to finding good paths through graphs. The authors of Ref. 9 used different pheromones to denote different QoS attributes. Ref. 10 modeled the Web service selection problem as a multi-objective optimization problem, and proposed a multi-objective chaos ACO algorithm to solve it. Ref. 11 integrated the max-min ant system into the framework of culture algorithm to solve the Web service selection problem.

The GA belongs to the larger class of evolutionary algorithms, which generate approximate solutions to optimization and search problems using techniques inspired by the principles of natural evolution: selection, crossover, and mutation. The original GA was invented by Dr. John Holland in 1975. The authors of Ref. 12 proposed a GA with static and dynamic penalty strategies in the fitness function. Ref. 13 designed a repair GA to address the Web service composition problem in the presence of domain constraints and inter service dependencies. Ref. 14 proposed a GA to conduct the service composition, in which considering quantitative and qualitative non-functional properties.

The PSO is one of the evolutionary computational techniques developed by Dr. Eberhart and Dr. Kennedy in 1995, which was inspired by the social behavior of bird flocking and fish schooling. The PSO algorithm finds the optimal solution through iterations after initializing a group of random particles. Zhang[15] proposed a QoS-aware Web service selection approach based on particle swarm optimization. Kang and so on[16] transformed the Web service selection problem into a multi-objective optimization problem with global QoS constraints, then introduced a particle swarm optimization algorithm to solve it. Ref. 17 presented a hybrid algorithm that combined the Munkers algorithm with particle swarm optimization to solve the Web service composition problem.

In our previous work,[18] we have presented a survey on bio-inspired algorithms for Web service composition. We also conducted a systematic review[19] on the current research of Web service concretization based on ACO algorithms, GAs, and PSO algorithms. The data-intensive service provision is characterized by the large of scale of services and data and also the high-dimensions of QoS. However, most of the existing works failed to take into account the characteristics of data-intensive services and the effect of the big data sets on the whole performance of service provision. In this paper, we will investigate the applications of the particle swarm optimization algorithm to solve the data-intensive service provision problem.

3. Data-Intensive Service Provision Based on PSO

3.1. Problem Statement

3.1.1. Data-Intensive Service Provision Problem

The data-intensive service composition problem is an extension of the traditional service composition problem, in which data sets as inputs and outputs of services, are incorporated. The problem is modeled as a graph, denoted as $G = (V,E,D)$, as shown in Fig. 1, where $V = \{AS_1, AS_2, \ldots, AS_n\}$ and E represent the vertices and edges of the graph respectively, $D = \{d_1, d_2, \ldots, d_z\}$ represents a set of z data servers. Each edge (AS_i, AS_j) represent a relationship between AS_i and AS_j. Each abstract service AS_i has its own service candidate set $cs_i = \{cs_{i,1}, cs_{i,2}, \ldots, cs_{i,m}\}, i \in \{1,\ldots,n\}$, which includes all concrete services to execute AS_i. Each abstract service AS_i requires a set of data sets, denoted by DT^i, that are distributed on a subset of D. A binary decision variable $x_{i,j}$ is the constraint used to represent only one concrete service is selected to replace each abstract service during the process of service composition, where $x_{i,j}$ is set to 1 if $cs_{i,j}$ is selected to replace abstract service AS_i and 0 otherwise.

For simplicity, it is assumed that all data sets needed by each service have already been distributed in data centers prior to service composition following a uniform distribution. In addition, we will only consider the cost and response time of data-intensive services.

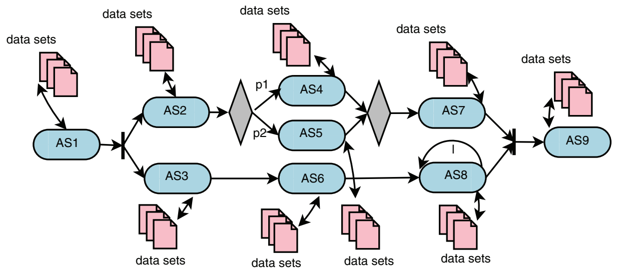

Fig. 1. An example of graph for data-intensive service provision.

3.1.2. QoS Model

Consider a data-intensive service $\widetilde{s}$ on platform y has been chosen to implement abstract service AS_i, which is connected by links of different bandwidths with all the data servers. The price of data set dt is denoted by p_{dt}, which is the fee that a data user has to pay to the data provider for the data usage. The size of data set dt is denoted by $size(dt)$. $C_{ac}(\widetilde{s})$ and $C_{tr}(\widetilde{s})$ are used to denote the access cost and the transfer cost of all data sets required by $\widetilde{s}$, respectively. $C_{sr}(\widetilde{s})$ is used to denote the service related cost, which mainly includes the cost to provide the service and the cost to process the data sets. The data transfer time, $T_t(dt, d_{dt}, y)$, is the time to transfer the data set $dt \in DT^i$ from the remote platform d_{dt} that houses the data to the local platform y, which has the service requesting the data. The cost for service $\widetilde{s}$, denoted by $Cost(\widetilde{s})$, can be described by (1).

$$\begin{aligned}
&Cost(\widetilde{s}) = C_{ac}(\widetilde{s}) + C_{tr}(\widetilde{s}) + C_{sr}(\widetilde{s}) \\
&C_{ac}(\widetilde{s}) = \sum_{dt \in DT^i} p_{dt} \\
&C_{tr}(\widetilde{s}) = \sum_{dt \in DT^i} T_t(dt, d_{dt}, y) * tcost \\
&T_t(dt, d_{dt}, y) = size(dt)/bw(d_{dt}, y)
\end{aligned} \tag{1}$$

The variable $tcost$ is the cost of data transfer per time unit, $bw(d_{dt}, y)$ is the network bandwidth between data server d_{dt} and service platform y, and $size(dt)/bw(d_{dt}, y)$ denotes the practical transfer time.

The estimated execution time for service $\widetilde{s}$, denoted by $T_{et}(\widetilde{s})$, includes the time for processing data sets, which is denoted by $T_p(\widetilde{s})$, and the time for accessing data sets, which is denoted by $T_{ad}(\widetilde{s})$. $T_{ad}(\widetilde{s})$

is the maximum value of response time for accessing all data sets required by service $\widetilde{s}$. The access response time of data set dt, which is denoted by $T_{rt}(dt)$, includes the data transfer time $T_t(dt, d_{dt}, y)$, the storage access latency $T_{sal}(d_{dt})$, and the request waiting time $T_{wt}(d_{dt})$. Thus, $T_{et}(\widetilde{s})$ can be described by (2).

$$\begin{aligned}
T_{et}(\widetilde{s}) &= T_p(\widetilde{s}) + T_{ad}(\widetilde{s}) \\
T_{ad}(\widetilde{s}) &= \max_{dt \in DT^i} \left(T_{rt}(dt)\right) \\
T_{rt}(dt) &= T_t(dt, d_{dt}, y) + T_{sal}(d_{dt}) + T_{wt}(d_{dt}) \\
T_{sal}(d_{dt}) &= size(dt)/sp(d_{dt}) \\
T_{wt}(d_{dt}) &= \sum_{i=1}^{nr} \left(size(dt^i)/sp(d_{dt})\right)
\end{aligned} \tag{2}$$

The variable $sp(d_{dt})$ is the storage media speed of d_{dt}, and nr is the number of data requests waiting in the queue prior to the underlying request for dt. The data transfer time depends on the network bandwidth and the size of the data. The storage access latency is the delayed time for the storage media to serve the requests, and it depends on the size of the data and storage type.[20] Each storage media 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 that are prior to it have finished.

3.1.3. *Utility Function*

Suppose a composite service CS is composed of n abstract services, and there are m concrete services to implement each abstract service. Each concrete service $cs_{i,j}$ is associated with a QoS vector $q_{ij} = [q_{ij}^1, q_{ij}^2, \ldots, q_{ij}^r]$ with r QoS parameters ($i \in \{1,2,\ldots,n\}$, $j \in \{1,2,\ldots,m\}$). The set of QoS attributes can be classified into two groups: positive and negative QoS attributes. The values of negative QoS attributes like response time need to be minimized. The higher their values, the lower the QoS attributes. The values of positive QoS attributes such as availability need to be maximized. The higher their values, the higher the QoS attributes. In order to evaluate the multidimensional quality of concrete service $cs_{i,j}$, an evaluation function is used. The function maps the quality vector q_{ij} into a single real value to enable selecting of service candidates. In this paper, a multiple attribute decision-making approach for the evaluation function is used, that is, the simple additive weighting (SAW) technique.[21]

There are two phases in applying SAW: 1) the scaling phase is used to normalize all QoS attributes to the same scale, independent of their units and ranges; 2) the weighting phase is used to compute the utility of each service candidate by using weights depending on users' priorities and preferences. For negative QoS attributes, values are scaled according to (3). For positive QoS attributes, values are scaled according to (4).

$$V_{i,j}^k = \begin{cases} \dfrac{Q_{k,i}^{max} - q_{ij}^k}{Q_k^{MAX} - Q_k^{MIN}} & \text{if } Q_k^{MAX} - Q_k^{MIN} \neq 0 \\ 1 & \text{if } Q_k^{MAX} - Q_k^{MIN} = 0 \end{cases} \tag{3}$$

$$V_{i,j}^k = \begin{cases} \dfrac{q_{ij}^k - Q_{k,i}^{min}}{Q_k^{MAX} - Q_k^{MIN}} & \text{if } Q_k^{MAX} - Q_k^{MIN} \neq 0 \\ 1 & \text{if } Q_k^{MAX} - Q_k^{MIN} = 0 \end{cases} \tag{4}$$

In (3) and (4), $Q_{k,i}^{max}$ and $Q_{k,i}^{min}$ ($k \in \{1,2,\ldots,r\}$) are the maximum and minimum values of the *k-th* QoS attributes for candidate set cs_i, which are given by (5).

$$\begin{aligned}
Q_{k,i}^{min} &= \min_{\forall cs_{i,j} \in cs_i} q_{ij}^k \\
Q_{k,i}^{max} &= \max_{\forall cs_{i,j} \in cs_i} q_{ij}^k
\end{aligned} \tag{5}$$

Furthermore, Q_k^{MAX} and Q_k^{MIN} ($k \in \{1,2,\ldots,r\}$) are the maximum and minimum possible aggregated values of the *k-th* QoS attributes for the composite service CS, which can be estimated by (6).

$$\begin{aligned}
Q_k^{MIN} &= F(k)_{i=1}^n Q_{k,i}^{min} \\
Q_k^{MAX} &= F(k)_{i=1}^n Q_{k,i}^{max}
\end{aligned} \tag{6}$$

The function F denotes the aggregation function of the *k-th* QoS attribute, which could refer our previous work.[18]

In addition, we use $Q_k^{MAX} - Q_k^{MIN}$ instead of $Q_{k,i}^{max} - Q_{k,i}^{min}$ as the denominator of (3) and (4) because this scaling approach[22,23] ensures that the evaluation of each concrete service is globally valid.

And this scaling method is important for guiding local selection.

The utility function for a concrete service $cs_{i,j}$ is computed according to (7).

$$U_{cs_{i,j}} = \sum_{k=1}^{r} (V_{i,j}^{k} * W_k) \tag{7}$$

Here, $W_k \in [0,1]$ and $\sum_{k=1}^{r} W_k = 1$. W_k represents the weight of *k-th* quality criteria with value normally provided by the users based on their own preferences.

The utility function for a composite service CS is also computed according to the SAW method, which is similar to (3)-(7). When scaling the aggregated values of QoS attributes for a composite service, the numerators of (3) and (4) should be replaced by (8) and (9).

$$Q_k^{MAX} - q_{CS}^{k} \tag{8}$$

$$q_{CS}^{k} - Q_k^{MIN} \tag{9}$$

Here, q_{CS}^{k} is the *k-th* QoS attribute of composite service CS.

3.2. Data-Intensive Service Provision Based on PSO algorithm

3.2.1. The standard PSO algorithm

PSO has been widely used to solve optimization problems because it is very robust, and it is simple and easy to implement. It uses a population of particles that fly through the search space of the problem with given velocities. The velocity of a particle determines the direction and distance of its evolution. Each particle i corresponds to a possible solution of the problem. The position of each particle is determined by a d-dimensional vector $X_i^t = \{x_{i1}, x_{i2}, \ldots, x_{id}\}$, and the velocity of the particle is determined by vector $V_i^t = \{v_{i1}, v_{i2}, \ldots, v_{id}\}$, where d is equal to the dimension of the problem hyperspace, t means the *t-th* iteration. At each iteration, the velocity of an individual particle is adjusted according to the historical best position for the particle itself and the neighborhood best position. Both the particle best position and the neighborhood best position are derived according to a user defined fitness function.[24]

The particles update their positions and velocities according to (10).

$$\begin{aligned} X_i^{t+1} &= X_i^t + V_i^{t+1} \\ V_i^{t+1} &= \omega V_i^t + c_1\gamma_1(P_i^t - X_i^t) + c_2\gamma_2(P_g^t - X_i^t) \end{aligned} \tag{10}$$

The variable ω is the inertia weight that controls the exploration and exploitation of the search space. γ_1 and γ_2 are two mutually independent random numbers. The variables c_1 (cognition coefficient) and c_2 (social coefficient) are the acceleration constants,[24] which change the velocity of particle i towards P_i^t and P_g^t. The variable P_i^t is the best position ever found by the particle i, whose corresponding fitness value is called the particle's best. The variable P_g^t is the best position found by the whole swarm, whose corresponding fitness value is called global best.

The following procedure can be used for implementing the PSO algorithm.

1 *Initialization. The swarm population is formed.*
2 *Evaluation. The fitness of each particle is evaluated.*
3 *Modification. The best position of each particle, the velocity of each particle, and the best position of the whole swarm are modified.*
4 *Update. Each particle is moved to a new position.*
5 *Termination. Steps 2-4 are repeated until a stopping criterion is met.*

The standard PSO approach is commonly used on real-valued vector spaces, and can also be applied to discrete-valued problems where either binary or integer variables have to be arranged into particles.[24] When integer solutions (not necessarily 0 or 1) are needed, one way is to round off the real optimum values to the nearest integer,[25] and the other way is to define new operations and entities of the PSO approach. For example, in a discrete PSO approach, the velocity's update is randomly selected from the three parts in the right-hand side of Eq. (10), using probabilities depending on the value of the fitness function.[24] Inspired by Ref. 26, in the following section we will redefine operations and entities of Eq. (10) to adapted the PSO algorithm to solve the service composition problem.

3.2.2. *The proposed discrete PSO algorithm*

The first key issue when designing the PSO algorithm lies in its particle representation that related to the problem hyperspace. In order to construct a direct relationship between the domain of the service composition problem and the PSO particles, d numbers of dimensions are presented n abstract services in the composition process. Hence, each particle i corresponds to a candidate solution of the service composition problem. The location of a particle in the d-th dimension represents the concrete service which the d-th abstract service selects. Fig. 2 shows a particle representation for a service composition problem with nine abstract services and each has one hundred concrete services.

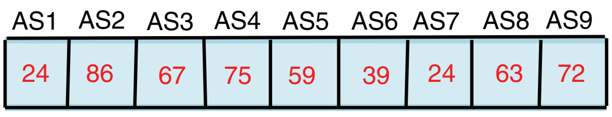

Fig. 2. Illustration of a particle's solution.

The fitness function of the PSO algorithm is the utility function which presented in section 3.1.3. Formally, the optimization problem in this paper is described as follows. Find a solution CS in graph G by replacing each abstract service AS_i in V with a concrete service $cs_{i,j} \in cs_i$ such that the overall fitness U_{CS} is maximized with the constrained condition.

The process of generating a new position for a selected particle in the swarm is depicted in Eq. (11).

$$\begin{aligned} V_i^{t+1} &= V_i^t \oplus \left((P_i^t \ominus X_i^t) \oplus (P_g^t \ominus X_i^t) \right) \\ X_i^{t+1} &= X_i^t \oplus V_i^{t+1} \end{aligned} \tag{11}$$

The definitions of the operators, used in the body of Eq. (11) are as follow.

The subtract operator ($\ominus$). Differences between a desired position P_i^t (or P_g^t) and the current position X_i^t of the i-th particle can be viewed as a velocity, which can be presented by a d-dimensional vector in which, each element shows that whether the content of the corresponding element in X_i^t is different from the desired one or not. If yes, that element gets its value from P_i^t (or P_g^t). For those elements that have the same content in X_i^t and P_i^t (or P_g^t), their corresponding values are assigned to zero.

The add operator ($\oplus$). This operator is a crossover operator that typically is used in GAs. We select two cut points from the current position X_i^t and the updated velocity V_i^{t+1} of the i-th particle, or from the two new created velocities by the subtract operator ($\ominus$), namely, $P_i^t \ominus X_i^t$ and $P_g^t \ominus X_i^t$, or from the old velocity V_i^t and the new created velocity by the add operator, and exchange dimensions between them. This type of crossover is traditionally known as two-point crossover, and it produces two new chains. Then this two new chains are checked if they have zero elements. If yes, the zero element will be replaced by the assignment of another concrete service in the service candidate set, according to the local selection approach. The local selection approach is based on the utilities of the concrete services. Prior to the crossover operation, the utility of each concrete service in each service candidate set is computed. Then we select the chain that has the bigger fitness function value. Fig. 3 illustrates the manner in which $\oplus$ operator performs.

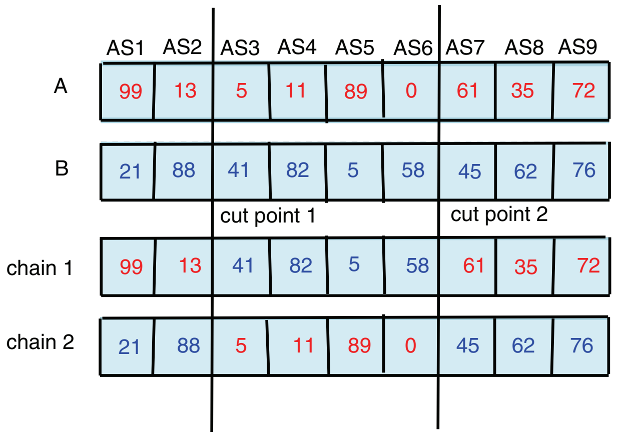

Fig. 3. Illustration of the manner in which $\oplus$ operator performs.

As it seems, the proposed equations have all major characteristics of the standard PSO equations.

The coefficient is a vector of ones. The following pseudo-code describes in detail the steps of the proposed PSO algorithm.

Data-Intensive Service Provision based on Discrete PSO

```
$t = 0$;
for $i = 1$ to swarm size
    Initialize $X_i^t$ and $V_i^t$;
    $P_i^t \leftarrow X_i^t$;
end for
calculates the fitness value of each particle $U(X_i^t)$;
$P_g^t \leftarrow \{X_l^t | l = \arg\max_{\forall i}\{U(X_i^t)\}\}$;
while 1
    for $i = 1$ to swarm size
        update velocity and position using (11);
        calculates the fitness value $U(X_i^{t+1})$;
        if $U(X_i^{t+1}) > U(P_i^t)$
            $P_i^{t+1} \leftarrow X_i^{t+1}$;
        else
            $P_i^{t+1} \leftarrow P_i^t$;
        end if
    end for
    if $\arg\max_{\forall i}\{U(P_i^{t+1})\} > U(P_g^t)$
        $P_g^{t+1} \leftarrow \{P_l^{t+1} | l = \arg\max_{\forall i}\{U(P_i^{t+1})\}\}$;
    end if
    $t \leftarrow t + 1$;
    if stopping criteria are true
        break;
    end if
end while
return $P_g^t$.
```

4. Computational experiments

4.1. Evaluation method and the dataset

In this section, we investigate a comparison study on the effectiveness of the proposed PSO algorithm, the ACO algorithm[27] and the GA[28] by solving the data-intensive service composition problem. The experimental frameworks involves two influencing variables: the number of abstract services in the composite service, and the number of concrete services for each abstract service. Here, the influence of the number of data sets is not considered.

For the purpose of the evaluation, different scenarios are considered where a composite service consists of n abstract services, and n varies in the experiments between 10 and 50, in increments of 10. There are m concrete services in each service candidate set, and m varies in the experiments between 100 and 1000, in increments of 100. Each abstract service requires a set of k data sets, and k is fixed at 10 in the experiments. A scenario generation system is designed to generate all scenarios for the 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.

Three performance factors were evaluated: 1) the required computation time; 2) the quality of the solution; and 3) the value of $OptIT$, which is the number of the iterations when the best utility appeared, and from this iteration the best utility will not change until the termination condition has been reached. As PSO, ACS and GA are sub-optimal, the solutions obtained by these three bio-inspired algorithms have been evaluated through comparing them with the optimal solutions obtained by the mixed integer programming (MIP) approach.[22,29] The open source integer programming system *lpsolve* version 5.5.2.5[30] was used for the MIP approach.

The number of ants in the ACO[27], the population in the GA[28], and the number of particles in PSO is all set to 100. And the maximum number of iterations is set to 1000. The synthetic datasets were experimented. For each scenario, the price of a data set, the network bandwidth (Mbps) between each data server and the service platform, 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 50 runs.

All runs of the same scenario use the same data, and the average results over 50 independent runs are reported. All the experiments are conducted on MATLAB 7.13.0.564 and executed on a computer with Inter(R) Core(TM) i7-6700HQ CPU 2.6GHz with 16GB of RAM.

4.2. *Results Analysis*

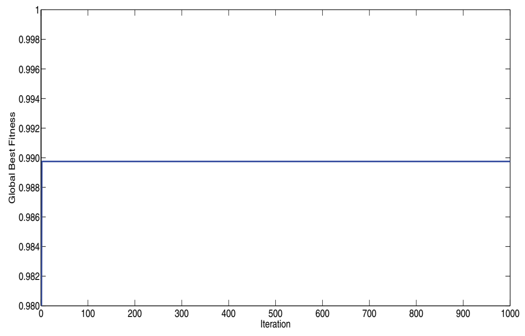

Fig. 4. The evolution of $U(P_g^t)$ over the PSO iteration.

One example of all experiments is given to show the evolution of fitness value over the PSO iteration, which is shown in Fig. 4. The figure shows a scenario where a composite service consists of 10 different abstract services, and each abstract service has 200 service candidates. In the figure, the short blue line represents the global best fitness value in the current iteration. The results indicate that the PSO could find the best fitness in the second iteration.

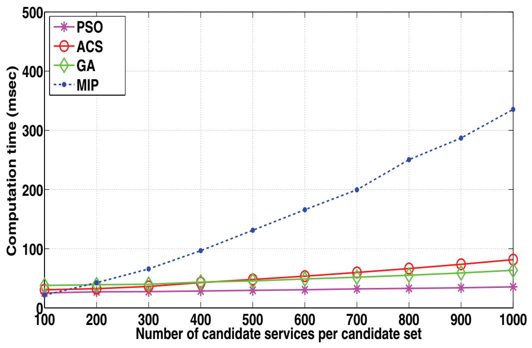

Fig. 5. The computation time of PSO, ACS, GA, and MIP vs. number of candidate services

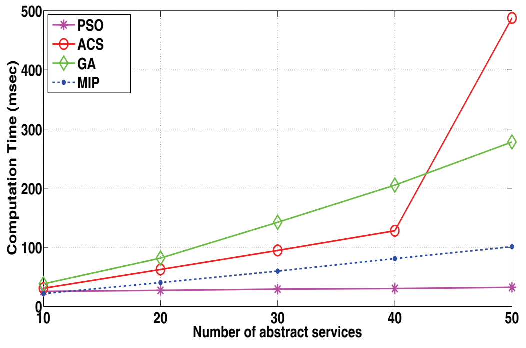

Fig. 6. The computation time of PSO, ACS, GA, and MIP vs. number of abstract services

In Fig. 5 and Fig. 6, the computation time of PSO, ACS, GA, and MIP are compared with respect to the number of candidate services and the number of abstract services. In Fig. 5, the number of candidate services for each abstract service varies from 100 to 1000, while the number of abstract service is set to 10. The results indicate that the proposed PSO is faster than other methods. By increasing the number of candidate services, the required computation time of PSO increases very slowly, this makes PSO be more scalable. In Fig. 6, the number of abstract services varies from 10 to 50, while the number of candidate services for each abstract service is fixed at 100. The results of this experiment indicate that the performance of all methods degrades as the number of abstract services increases. Again, we observe that PSO outperforms others. The reason is that we adopt a local selection approach in the add operator of our discrete PSO algorithm.

Furthermore, after collecting all the results of the scenarios, we find that the fitness value of the final best solution obtained by PSO, ACS, GA in all scenarios are the same with that of the MIP approach. In other words, the results show that all the three bio-inspired algorithms are able to achieve the optimal solutions. That's because we use $Q_k^{MAX} - Q_k^{MIN}$ as the denominator of Eq. (3) and Eq. (4), and this strategy could guide the local selection and ensure the evaluation of each concrete service is globally valid.

We also compared the mean of $OptIT$ values obtained by PSO, ACS and GA with respect to a varying number of candidate services and a varying number of abstract services. When the number of abstract services is fixed at 10, as the number of concrete services increases from 100 to 1000, the mean of $OptIT$ values obtained by PSO, ACS and GA are 2, 1 and 5 in all scenarios, respectively. That is to say, the three bio-inspired algorithms found the best fitness value in the fixed iteration, and it does not change as the number of concrete services increases. When the number of concrete services is fixed at 100, as the number of abstract services increases from 10 to 50, the mean of $OptIT$ values obtained by PSO is 2 for all scenarios, the mean of $OptIT$ values obtained by ACS is 1, 1, 1, 1, 3 for each scenario, and the mean of $OptIT$ values obtained by GA is 5, 8, 10, 11, 12 for each scenario. The results show that ACS and GA need more iterations to get the best fitness value.

5. Conclusion

In this paper, we present and evaluate a discrete particle swarm optimization algorithm to support the data-intensive service composition. The composition problem is modeled as a directed graph. Replacing the classical operators in the standard particle swarm optimization algorithm, we proposed equations analogous to them. In our discrete version of particle swarm optimization, the representation of the position and velocity of the particle is integer vector, by which we accomplished the mapping between the concrete service selection and the particle. The algorithm adopts a local selection method and a two-point crossover operation to find the optimal solution. Comparisons with the ant colony optimization algorithm, the genetic algorithm, and the mixed integer programming approach show the scalability and effectiveness of our proposed algorithm. For future work, extension of the proposed approach will apply for solving multi-objective and dynamic data-intensive service composition problem.

Acknowledgments

This work has been supported by National Natural Science Foundation of China under Grant No. 61702398 and No. 61502370, Natural Science Basic Research Plan in Shaanxi Province of China (Program No.2016JQ6003), China 111 Project under Grant No.B16037, and the Fundamental Research Funds for the Central Universities under Grant No.XJS15045.

References

1. G. Bell, T. Hey, A. Szalay, Beyond the data deluge, Science 323 (5919) (2009) 1297–1298.
2. W. Huang, Z. Chen, W. Dong, H. Li, B. Cao, J. Cao, Mobile internet big data platform in china unicom, Tsinghua Science and Technology 19 (1) (2014) 95–101.
3. Y. Tian, C. Liu, Z. Chen, J. Wan, X. Peng, Performance evaluation and dynamic optimization of speed scaling on web servers in cloud computing, Tsinghua Science and Technology 18 (3) (2014) 298–307.
4. C. Philip Chen, C. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on big data, Information Sciences 275 (2014) 314–347.
5. S. Balasubramaniam, D. Botvich, R. Carroll, J. Mineraud, T. Nakano, T. Suda, W. Donnelly, Biologically inspired future service environment, Computer Networks 55 (15) (2011) 3423–3440.
6. L. Wang, J. Shen, J. Luo, Bio-inspired cost-aware optimization for data-intensive service provision, Concurrency and Computation: Practice and Experience 27 (18) (2015) 5662–5685.
7. S. Dustdar, M. P. Papazoglou, Services and service composition - an introduction, Information Technology 50 (2) (2008) 86–92.
8. F. Dong, J. Luo, J. Jin, J. Shi, Y. Yang, J. Shen, Accelerating skycube computation with partial and parallel processing for service selection, IEEE Transactions on Services Computing, Online Available (99), `http://dx.doi.org/10.1109/TSC.2017.2762681`.
9. Y. Xia, J. Chen, X. Meng, On the dynamic ant colony algorithm optimization based on multi-pheromones, in: Proceedings of 7th IEEE/ACIS International Conference on Computer and Information Science (ICIS '08), IEEE Computer Society, Washington, DC, USA, 2008, pp. 630–635.
10. L. Wang, Y. He, A web service composition algorithm based on global qos optimizing with mocaco, in: L. Jiang (Ed.), Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE 2011), Vol. 111 of Advances in Intelligent and Soft Computing, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 79–86.
11. Z. J. Wang, Z. Z. Liu, X. F. Zhou, Y. S. Lou, An approach for composite web service selection based on dgqos, The International Journal of Advanced Manu-

facturing Technology 56 (9-12) (2011) 1167–1179.
12. G. Canfora, M. Penta, R. Espositio, M. L. Villani, An approach for qos-aware service composition based on genetic algorithms, in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05), ACM, New York, NY, USA, 2005, pp. 1069–1075.
13. L. Ai, M. Tang, Qos-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm, in: Proceedings of IEEE 4th International Conference on eScience (eScience '08), IEEE Computer Society, Washington, DC, USA, 2008, pp. 119–126.
14. H. Wang, P. Ma, X. Zhou, A quantitative and qualitative approach for nfp-aware web service composition, in: Proceedings of IEEE 9th International Conference on Services Computing (SCC), IEEE Computer Society, Washington, DC, USA, 2012, pp. 202–209.
15. T. Zhang, Qos-aware web service selection based on particle swarm optimization, Journal of Networks 9 (3) (2014) 565–570.
16. G. Kang, J. Liu, M. Tang, Y. Xu, An effective dynamic web service selection strategy with global optimal qos based on particle swarm optimization algorithm, in: IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2012, pp. 2274–2279.
17. S. A. Ludwig, Applying particle swarm optimization to quality of service driven web service composition, in: IEEE 26th International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, 2012, pp. 613–620.
18. L. Wang, J. Shen, J. Yong, A survey on bio-inspired algorithms for web service composition, in: IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE Computer Society, Washington, DC, USA, 2012, pp. 569–574.
19. L. Wang, J. Shen, A systematic review of bio-inspired service concretization, IEEE Transactions on Services Computing 10 (4) (2017) 493–505.
20. H. H. E. Al-Mistarihi, C. H. Yong, Response time optimization for replica selection service in data grids, Journal of Computer Science 4 (6) (2008) 487–493.
21. K. P. Yoon, H. C. Land, Multiple Attribute Decision Making: An Introduction, Vol. 104 of Quantitative Applications in the Social Sciences, SAGE Publications, Inc., CA, Thousand Oaks, 1995.
22. M. Alrifai, T. Risse, W. Nejdl, A hybrid approach for efficient web service composition with end-to-end qos constraints, ACM Transactions on the Web 6 (2) (2012) 7:1–7:31.
23. S. X. Sun, J. Zhao, A decomposition-based approach for service composition with global qos guarantees, Information Sciences 199 (0) (2012) 138–153.
24. Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, R. G. Harley, Particle swarm optimization: Basic concepts, variants and applications in power systems, IEEE Transactions on Evolutionary Computation 12 (2) (2008) 171–195.
25. K. Parsopoulos, M. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Compuing 1 (2002) 235–306.
26. A. H. Kashan, B. Karimi, A discrete particle swarm optimization algorithm for scheduling parallel machines, Computers & Industrial Engineering 56 (1) (2009) 216–223.
27. L. Wang, J. Shen, G. Beydoun, Enhanced ant colony algorithm for cost-aware data-intensive service provision, in: IEEE 9th World Congress on Services, IEEE Computer Society, Washington, DC, USA, 2013, pp. 227–234.
28. L. Wang, J. Shen, J. Luo, F. Dong, An improved genetic algorithm for cost-effective data-intensive service composition, in: Proceedings of The 9th International Conference on Semantics, Knowledge & Grids (SKG), IEEE Computer Society, Washington, DC, USA, 2013, pp. 105–112.
29. R. Ramacher, L. Monch, Cost-minimizing service selection in the presence of end-to-end qos constraints and complex charging models, in: Proceedings of IEEE 9th International Conference on Services Computing (SCC), IEEE Computer Society, Washington, DC, USA, 2012, pp. 154–161.
30. M. Berkelaar, K. Eikland, P. Notebaert, lpsolve: Mixed-integer linear programming solver, `http://lpsolve.sourceforge.net/` (2004).