2014

An improved genetic algorithm for cost-effective data-intensive service composition

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

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

Junzhou Luo
Southeast University

Fang Dong
Southeast University

Publication Details

Wang, L., Shen, J., Luo, J. and Dong, F. (2014). An improved genetic algorithm for cost-effective data-intensive service composition. 2013 Ninth International Conference on Semantics, Knowledge and Grids (SKG) (pp. 105-112). United States: The Institute of Electrical and Electronics Engineers, Inc. 2014

An improved genetic algorithm for cost-effective data-intensive service composition

Abstract
The explosion of digital data and the dependence on data-intensive services have been recognized as the most significant characteristics of IT trends in the current decade. Designing workflow of data-intensive services requires data analysis from multiple sources to get required composite services. Composing such services requires effective transfer of large data. Thus many new challenges are posed to control the cost and revenue of the whole composition. This paper addresses the data-intensive service composition and presents an innovative data-intensive service selection algorithm based on a modified genetic algorithm. The performance of this new algorithm is also tested by simulations and compared against other traditional approaches, such as mix integer programming. The contributions of this paper are three folds: 1) An economical model for data-intensive service provision is proposed, 2) An extensible QoS model is also proposed to calculate the QoS values of data-intensive services, 3) Finally, a modified genetic algorithm-based approach is introduced to compose data-intensive services. A local selection method with modifications of crossover and mutation operators is adopted for this algorithm. The results of experiments will demonstrate the scalability and effectiveness of our proposed algorithm.

Keywords
genetic, algorithm, service, cost, improved, effective, composition, data, intensive

Publication Details
Wang, L., Shen, J., Luo, J. and Dong, F. (2014). An improved genetic algorithm for cost-effective data-intensive service composition. 2013 Ninth International Conference on Semantics, Knowledge and Grids (SKG) (pp. 105-112). United States: The Institute of Electrical and Electronics Engineers, Inc. 2014

An Improved Genetic Algorithm for Cost-Effective Data-Intensive Service Composition

Lijuan Wang[1], Jun Shen[1], Junzhou Luo[2], Fang Dong[2]
[1]School of Information Systems and Technology
University of Wollongong, NSW, Australia
lw840@uowmail.edu.au
jshen@uow.edu.au
[2]School of Computer Science and Engineering
Southeast University, Nanjing, P. R. China
jluo@seu.edu.cn
fdong@seu.edu.cn

***Abstract*— The explosion of digital data and the dependence on data-intensive services have been recognized as the most significant characteristics of IT trends in the current decade. Designing workflow of data-intensive services requires data analysis from multiple sources to get required composite services. Composing such services requires effective transfer of large data. Thus many new challenges are posed to control the cost and revenue of the whole composition. This paper addresses the data-intensive service composition and presents an innovative data-intensive service selection algorithm based on a modified genetic algorithm. The performance of this new algorithm is also tested by simulations and compared against other traditional approaches, such as mix integer programming. The contributions of this paper are three folds: 1) An economical model for data-intensive service provision is proposed; 2) An extensible QoS model is also proposed to calculate the QoS values of data-intensive services; 3) Finally, a modified genetic algorithm-based approach is introduced to compose data-intensive services. A local selection method with modifications of crossover and mutation operators is adopted for this algorithm. The results of experiments will demonstrate the scalability and effectiveness of our proposed algorithm.**

I. INTRODUCTION

In recent years, the data generated by scientific activities, social networking, social media, as well as commercial applications has exponentially increased. This explosion of digital data and the dependence on data-intensive services are key characteristics of the IT tread in this decade. Effective developments of applications based on data-intensive services have become of the most important challenges in service-oriented computing. For example, computational scientists often perform complex computational analysis, which both consumes and produces large data sets. They build large-scale scientific applications which are composed from various data-intensive services into a composite service which can be re-used by other users. In this context, data-intensive service composition presents the following challenges.

- The data intensity and the communication cost of mass data transfer will have a significant impact on the performance of data-intensive applications.
- Large number of data sets and increasing functionality of related services make the composition complex. Furthermore, the development of cloud computing has made increasingly diverse services available, enabling Web-based publishing at different quality levels, for both the data itself and services based on it.
- Current businesses using cloud computing have begun to provide data management services, data transfer services, and data storage services. The delivery of Internet-scale data-intensive services has brought a number of challenges, such as the maintenance of quality of service (QoS) as well as excellent opportunities for businesses to gain market share, and for scientific programs to save on cost and energy as well as on space for data management.
- Dynamic and adaptive mechanisms are needed to be able to make optimized service composition decisions in a distributed manner.

In this paper, we address the challenges listed above for data-intensive service composition. We propose a modified genetic algorithm to compose data-intensive services. The differences between our algorithm and others are as follows.

- Most other studies have adopted roulette wheel selection which has problems when the fitness varies widely [5], [6], [8], [17], [18]. The selection operator in this paper is the combination of elitism selection and tournament selection, by first copying the fittest individual into the next generation and then producing other individuals of the population. Elitism selection can increase the performance of the GA very rapidly. Tournament selection is among the most widely used selection strategies in evolutionary algorithms. In addition, the combination of the two selection methods can guarantee the algorithm not to get stuck in local optima.
- In order to escape from local optima, a local selection rule and some modifications of the crossover and the mutation operators are adopted. The local selection method can improve the convergence speed to the global optimum. The

modifications of the crossover and the mutation operators can prevent the creation of the unfeasible solutions.
- Very little research has considered the effect of data intensity and the communication cost of mass data transfer on service composition. This paper is the first effort to address data-intensive service composition using a modified genetic algorithm.

The remainder of this paper is organized as follows: Section II reviews related work. Section III further specifies the problem. Section IV investigates how a modified genetic algorithm with a local selection method could be used to solve the data-intensive service composition problem. Section V shows the performance evaluation. Finally, Section VI concludes this paper and proposes future work.

II. RELATED WORK

In a QoS-aware service composition process, the service composer needs to select services to satisfy the quality constraints and achieve the optimization goals. The optimization goals can vary and can include such aims, as minimizing the overall cost of service usage or the overall response time, or providing the best value for user-defined utility function. The research area of QoS-aware service composition has attracted much attention in the past years. It should be noted that the optimization processes in most QoS-aware service composition studies aim to satisfy quality constraints and provide the "best" solutions with highest QoS. However, from a cost perspective, consumers will not always pay more, even if the qualities of the requested composite services exceed their expectations. This is because most service-composition processes involve static price-setting models.

A variety of studies adopted genetic algorithm (GA) to solve QoS-based service composition problems. The authors of [5] proposed a GA-based approach for QoS-aware service composition. Their experimental results showed that the differences between static and dynamic fitness functions were not significant. By comparing the GA approach with the linear integer programming method, the authors pointed out that integer programming was preferable when the number of concrete services was small. The paper [7] presented a cost-driven web service selection based on genetic algorithm. In the process of population initialization, an individual was checked to see whether the constraints were satisfied after it was generated. The checking process was also applied after each crossover operation and mutation operation.

In [9], [17], [28], a genetic algorithm based on a relation matrix coding scheme for QoS-aware service selection problem was proposed. The authors argued that the one dimension coding scheme cannot express all paths of composite service at the same time but the relation matrix coding schemes can. [28] adopted an initial population policy and the mutation policy to promote the fitness of genetic algorithm. [9] presented an enhanced initial population policy and an evolution policy for GA. [17] focused on how fitness function and a mutation policy affected the performance of the genetic algorithm.

The paper [18] proposed a hybrid genetic algorithm utilizing a local optimizer to improve the fitness value of the individuals in the population in order to improve the overall QoS value. The local optimizer was used to improve the individuals in the initial population after it was randomly generated. The study also used the local optimizer in the population at end of each generation, but this was not represented in the algorithm. The strategy for unfeasible individuals was that a penalty was given to their fitness values. Similarly, the paper [11] presented a hybrid genetic algorithm using a local improver for QoS-aware service composition. The local improvement procedure was based on an iterative neighborhood search so that a given solution was replaced with best neighbor found.

The existing GA-based approaches for service composition focus on different aspects such as the chromosome encoding scheme, initial population policy, crossover policy, mutation policy, the fitness function with a static or a dynamic penalty factor, the operators, and population diversity handling. Refer to the encoding method in GA-based approaches, some adopted a one dimensional chromosome encoding method [5], [29], but when the number of candidate services becomes very large, the readability of the chromosomes is very weak and it cannot represent the semantic information. So some GA-based approaches adopted a relation matrix coding scheme method [17]. However, this method frequently generated illegal individuals and became less efficient. The authors of [6] proposed a tree-encoding model which could express various composition relationships and carried static-models of service workflow and supported re-planning at run-time.

Mixed integer programming (MIP) approaches have been proposed to solve QoS-aware service composition with global constraints [2], [3], [5], [8], [12], [30]. We also formulate the data-intensive service composition with constrained time problem as a MIP model and use the open source integer programming system *lpsolve* version 5.5 [4] to solve it. Random selection approach is also a GA-based approach [5], [6], [8], [18]. This approach would randomly select a service candidate to replace during the mutation operation.

Genetic algorithms belong to the category of bio-inspired algorithms, which offer many advantages for dealing with data-intensive service provision problems. Bio-inspired optimization algorithms have been proposed to solve the service provision problem, because of the simplicity of the algorithms and the rapid convergence to optimal or near-optimal solutions [20], [22]. Biological entities can learn from their environment. They can sense the surrounding conditions and adaptively invoke behaviors suited to the conditions. Biological inspired systems are typically made of a population of simple agents, which try to build the feasible solution to apply the stochastic decision policy repeatedly. They are decentralized and self-organized systems. We have already done pilot studies in applying bio-inspired algorithms to tackle service composition problems [14], [19], [20], [21], [27]. [21] was the first effort to address the lower cost data-intensive service composition problem. Also, there is ample evidence regarding the applicability of genetic algorithms for large-scale optimization

problems [13], [23] and service composition in cloud computing [25]. In order to deal with the dynamic changes of services and network conditions in cloud computing, as well as the constraints of different users and the flexibility of the selection criteria, a modified genetic algorithm-based data-intensive service composition approach is proposed in this paper.

III. PROBLEM STATEMENT

A. An Economical Model of Data-Intensive Service Provision

In general, data-intensive service composition will be supported cooperatively by service composers, service providers, and data providers. Different providers need a method to regulate and price their resources, and they all want to have a position in the market whilst maximizing their profits. An economical model of data-intensive service provision is assumed to be an accurate representation of the reality and to offer a suitable way to regulate the interactions among the three providers. As shown in Fig. 1, in the downstream market, the service composer seeks optimal strategies to select elementary services provided by multiple service providers, who compete on the basis of price and quality of services. From the service composer's point of view, it is important to be able to assess the value of the needed services and how much it wants to pay for them to satisfy its users' requirements as well as to maximize its profit. From the service provider's perspective, it is important to be able to analyze its competitive position and improve its offers if it is to win contracts with the service composer. In the upstream market, the service provider requests the data from the data provider. The price of the data may affect the total cost and the price of services. Therefore the prices of service and data have a crucial impact on the service composer's and the service provider's profits.

This paper makes a distinction between cost and other QoS attributes because cost is usually related to other quality attributes and it becomes more important in data-intensive service provision. In traditional service composition, executable services and its input/output data are usually in the same site.

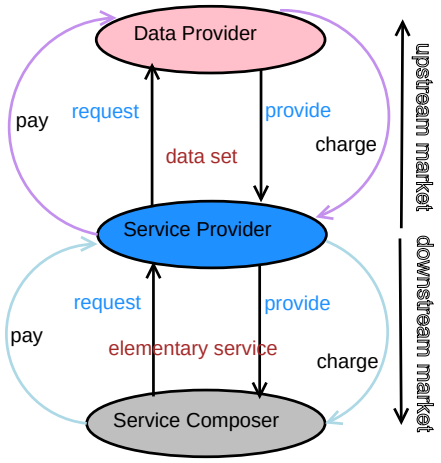

Fig. 1. Service and data set usage and charging relationship.

Thus the cost for data staging can be neglected or the cost is a constant determined before execution, and service selection algorithms need not consider it. However, in data-intensive service composition, providers charge users depending on the user's location and the amount of data transferred. Each service requests data sets from the storage resources (or data servers). Each of these data sets may be replicated at several locations that are connected to each other and to the service endpoints through networks of varying capability [10], [15], [24]. When composing data-intensive services, optimizing the cost of data is a priority, as data play the dominant role in the data-intensive service composition.

B. Data-Intensive Service Composition

In a Web service environment, abstract services are the functional descriptions of services, and concrete services represent the existing services which are available for potential invocation of their functionality and capabilities. When the function of several concrete services is consistent with the functional description of an abstract service, these concrete services are the service candidates of the abstract service and QoS attributes are used to distinguish them.

The data-intensive service composition problem is modelled as a directed graph, denoted as $G = (V, E, D)$, 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, which means that AS_i has to finish before invoking 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 k 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. Fig. 2 gives an example of a directed graph for data-intensive service composition, in which data sets, as the inputs and outputs of services, are incorporated. 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

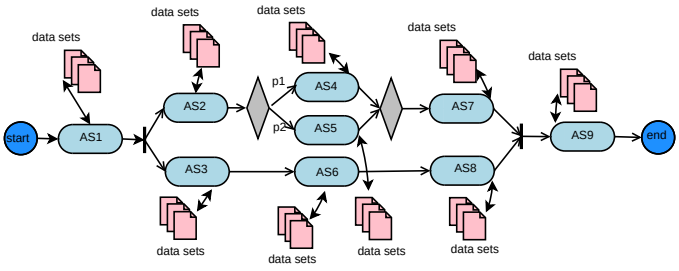

Fig. 2. An example of directed graph for data-intensive service composition

data-intensive services.

C. QoS Model

Consider a data-intensive service $cs_{i,j}$ on site y has been chosen to replace 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 $size(dt)$. $C_{ac}(AS_i)$ and $C_{tr}(AS_i)$ are used to denote the access cost and the transfer cost of all data sets required by AS_i respectively. $C_{sr}(AS_i)$ is used to denote the service related cost which mainly includes the cost to provision the service and the cost to process the data sets. For each data set $dt \in DT^i$, the time to transfer it from d_{dt} to y is denoted by $T_t(dt, d_{dt}, y)$. The cost for the task, $Cost(AS_i)$, can be described by (1).

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

Subject to:

$$\begin{aligned} &AS_i = \sum_{j=1}^{m} cs_{i,j} x_{i,j} \\ &\sum_{j=1}^{m} x_{ij} = 1, x_{ij} \in \{0,1\}, i \in \{1, \ldots, n\} \end{aligned} \tag{2}$$

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

The estimated execution time for the task, $T_{et}(AS_i)$, includes the time for processing data sets $T_p(AS_i)$ and the time for accessing data sets $T_{ad}(AS_i)$. $T_{ad}(AS_i)$ is the maximum value of time for accessing all data sets required by the task. The access response time of data set dt, $T_{rt}(dt)$, 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, $T_{et}(AS_i)$ can be given by (3).

$$\begin{aligned} &T_{et}(AS_i) = T_p(AS_i) + T_{ad}(AS_i) \\ &T_{ad}(AS_i) = \max_{dt \in DT^i} \left(T_{rt}(dt)\right) \\ &T_{rt}(dt) = T_t(dt, d_{dt}, y) + T_{sal}(d_{dt}) + T_{wt}(d_{dt}) \\ &T_{sal}(d_{dt}) = size(dt)/sp(d_{dt}) \\ &T_{wt}(d_{dt}) = \sum_{i=1}^{nr} \left(size(dt^i)/sp(d_{dt})\right) \end{aligned} \tag{3}$$

where $sp(d_{dt})$ is the storage media speed, nr is the number of data requests waiting in the queue prior to the underlying request for dt. The data transfer time $T_t(dt, d_{dt}, y)$ is the time to transfer the data set from the remote site that houses the data replica to the local site which has the service that requested the replica. It depends on the network bandwidth and the size of the data replica. 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 [1]. 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 prior it in the queue finish.

Using a genetic algorithm, the problem of finding a data-intensive service composition solution is considered as an optimization problem, in which the overall fitness value has to be maximized. 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 $F(CS)$ is maximized with the constrained execution time. The fitness function is described in the following section.

IV. DATA-INTENSIVE SERVICE SELECTION BASED ON A MODIFIED GENETIC ALGORITHM

Genetic algorithms (GAs) belong to the larger class of evolutionary algorithms (EAs), which generate approximate solutions to optimization and search problems by using techniques inspired by the principles of natural evolution: selection, crossover, and mutation. GA is a powerful tool to solve combinatorial optimization problems [16]. To use a GA to search for a solution to the data-intensive service composition problem, the first step is to encode the problem with a suitable genome. The encoding scheme of chromosomes in this paper is the integer array coding scheme. The initial population is created randomly according to a directed graph. Only one branch of the conditional operations is selected according to a certain probability. The selection operator in this paper is the combination of elitism selection and tournament selection.

The crossover operator in this paper is the single point crossover. The crossover point is created randomly but must be checked as to whether it will create unfeasible solutions. The checking rule is that there is one and only one branch of each conditional operation to be selected.

The mutation operator for each chromosome replaces the value of the gene with the assignment of another concrete service in the service candidate set according to the local selection rule which will be described as follows. The probability of mutation is for the locus of gene. The locus for each gene represents its own position in the chromosome. Every locus in each chromosome which was created by the crossover operation is checked for possible mutation. Before the mutation operation, it is necessary to check whether the value of the gene equals zero. If the value of the gene equals zero, it means the related abstract service is in a conditional branch and it is not selected, so the mutation operator will not be applied to this locus.

The local selection rule in the mutation operator is based on the utility of the concrete service. Prior to the mutation operation, the utility of each concrete service in each service candidate set is computed. Then all the concrete services in

each service candidate set are sorted in descending order according to their utility. When the mutation operation is applied, the replacement process will search another service candidate from the beginning of the service candidate set until the assignment is different from the old assignment, and then replace it.

Suppose a composite service CS is composed of n tasks, and there are m concrete services to execute each task. Each concrete service $cs_{i,j}$ is associated with a QoS vector $q_{ij} = [q^1_{ij}, q^2_{ij}, \ldots, q^r_{ij}]$ with r parameters. 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 [26].

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 (4). For positive QoS attributes, values are scaled according to (5).

$$V^k_{i,j} = \begin{cases} \dfrac{Q^{max}_{k,i} - q^k_{ij}}{Q^{max}_{k,i} - Q^{min}_{k,i}} & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} \neq 0 \\ 1 & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} = 0 \end{cases} \quad (4)$$

$$V^k_{i,j} = \begin{cases} \dfrac{q^k_{ij} - Q^{min}_{k,i}}{Q^{max}_{k,i} - Q^{min}_{k,i}} & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} \neq 0 \\ 1 & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} = 0 \end{cases} \quad (5)$$

In (4) and (5), $Q^{max}_{k,i}$ and $Q^{min}_{k,i}$ ($k \in \{1, 2, \ldots, r\}$) are used to represent the maximal value and the minimal value of the k-th QoS attributes of all concrete services in candidate set cs_i, which are given by (6).

$$\begin{aligned} Q^{min}_{k,i} &= \min_{\forall cs_{i,j} \in cs_i} q^k_{ij} \\ Q^{max}_{k,i} &= \max_{\forall cs_{i,j} \in cs_i} q^k_{ij} \end{aligned} \quad (6)$$

The overall score of $cs_{i,j}$ is computed according to (7).

$$score_{cs_{i,j}} = \sum_{k=1}^{r} (V^k_{i,j} * W_k) \quad (7)$$

where $W_k \in [0, 1]$ 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.

When using a GA to solve service composition problems, the fitness function always corresponds to QoS attributes. In this paper, the fitness function is used to evaluate the utility of a potential solution. As we discussed before, this paper will consider only the cost and response time of data-intensive services. The fitness value of a solution CS is computed as $F(CS) = \sum_{k=1}^{2} \frac{Q^{MAX}_k - q^k_{CS}}{Q^{MAX}_k - Q^{MIN}_k} * W_k$, where q^k_{CS} is the k-th aggregated QoS value, computed by applying the aggregation functions described in Table 1 of [5]. $Q^{MIN}_k = \sum_{i=1}^{n} Q^{min}_{k,i}$ and $Q^{MAX}_k = \sum_{i=1}^{n} Q^{max}_{k,i}$ represent the maximal value and the minimal value of the k-th QoS attributes of all the possible solutions, respectively.

The service selection algorithm based on a modified GA for data-intensive service composition is given in Algorithm 1 on the next page.

V. PERFORMANCE EVALUATION

The aim of this evaluation is to analyze the performance of the proposed algorithm: 1) observing the evolution of fitness value over the GA generations; 2) comparing our GA with mixed integer programming approach [2], [12]; 3) comparing our GA with GA-based random selection approach [5], [6], [8], [18]. All the experiments are conducted on computers with Inter Core i5 2500 CPU (3.3GHz and 8 GB RAM).

A. Test Case Generation

The values of parameters considered in this paper are: $Npop = 20$, $MaxIt = 1000$, $PC = 0.7$, $PM = 0.1$, $EIT = 100$. The weights for QoS attribute cost and response time are 0.8 and 0.2 respectively. QoS values of different concrete services, the number of data sets required by each abstract service, size and price of each data set, storage media speed and the number of waiting request of each data server, network bandwidth between data server and service endpoint were generated randomly with uniform distribution. There are two branches in all the conditional patterns with the probability of 0.5. 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 request in the waiting queue were randomly generated with uniform distribution from the following intervals: [1,100], [1,100], [1,100], [1000,10000] and [1,10].

The performance of the modified genetic algorithm is affiliated to the size of the data-intensive service composition problem. The size of the problem depends on the number of abstract services in the workflow and the number of concrete services for each abstract service. For the purpose of our evaluation, we considered some scenarios, where a composite application comprises services from n abstract services (n varies in our experiments between 10 and 50, in increments of 10). There are m concrete services in each service candidate set (m varies in our experiments between 100 and 1000, in increments of 100). Each abstract service requires a set of k data sets (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. Within this basic scenario, other scenarios are generated by

Algorithm 1 Data-intensive service selection algorithm based on GA

Input:
$MaxIt$: the maximum number of iterations;
EIT: the best fitness value remains unchanged for EIT generations;
$nstop$: the number of changes of the best fitness value;
G: the directed graph;
Output:
S: a service execute path to create a composite service ;

1: $iga = 1$;
2: $nstop = 0$;
3: randomly create an initial $Population$;
4: calculates the utility of each concrete service according to (7);
5: calculates the fitness value of each individual;
6: sorts and reorganizes $Population$ according to the fitness value of each individual in descending order;
7: records the best fitness, changes $nstop$;
8: **while** $iga < MaxIt$ **do**
9: $iga = iga + 1$;
10: applies elitism and tournament selection strategy;
11: **for** each pair of the selected parents **do**
12: applies crossover operator to produce two offspring;
13: adds the two offspring into the next generation;
14: **end for**
15: **for** each new individual x created by crossover operation **do**
16: **for** each locus l of x **do**
17: **if** the value of l is not zero and l needs to mutate **then**
18: stores x and calculate the fitness $F(x)$;
19: uses the local selection rule to replace the value of l with assignment of another service candidate to produce new offspring $x1$;
20: calculates the fitness $F(x1)$;
21: **if** $F(x) > F(x1)$ **then**
22: $x1 = x$;// the old individual with bigger fitness value survives;
23: **end if**
24: **end if**
25: **end for**
26: **end for**
27: all individuals in the next generation create the new $Population$;
28: **for** each individual in $Population$ **do**
29: calculates the fitness values;
30: **end for**
31: sorts and reorganizes $Population$ according to the fitness value of each individual in descending order;
32: records the best fitness, changes $nstop$;
33: **if** $(iga > MaxIt) \bigcup (nstop > EIT)$ **then**
34: break;
35: **end if**
36: **end while**
37: sets the first individual in $Population$ to S;
38: **return** S.

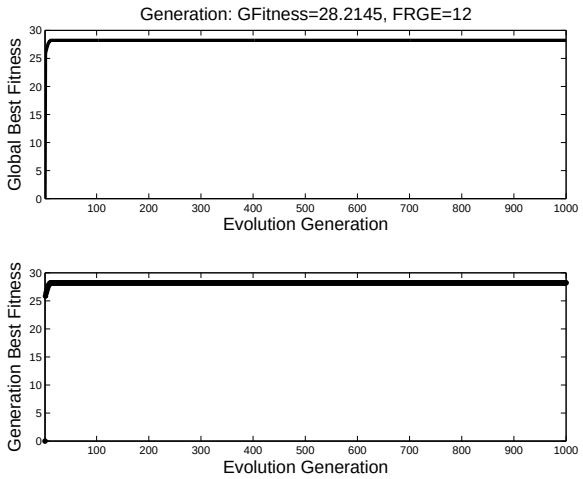

Fig. 3. The evolution of fitness value

either placing an abstract service into it or adding another composition structure as substructure. This procedure ends until the scenario has the predefined number of abstract services. All scenarios were executed 50 times and the average values are reported from these experiments.

B. Result Analysis

We give one example of all experiments to show the evolution of fitness value over the GA generations. Fig. 3 shows a scenario where a composite application comprises services from 30 different service classes, the number of service candidates per class is 500, and the number of data set for each service candidate is 10. In Fig. 3, the short line on the above half denotes the fitness value of the best individual from the beginning of the trial, and the point on the below half denotes the fitness value of the best individual of each generation. The value of 'GFitness' is the fitness value of the best individual from the beginning of the trial and it depends on the QoS attributes of services. Different values of QoS attributes give different values of 'GFitness'. That is to say, the change of the value of 'GFitness' has no significance for the simulation results. The value of 'FRGE' is the number of generations when the best fitness value appeared and from this generation the value of the best fitness will not change.

When comparing GA with MIP, the GA is required to reach the same solution as of MIP. In Fig. 4 on the next page, we compare the performance of our modified genetic algorithm with the performance of MIP approach with respect to the number of abstract services. The number of abstract services varies from 10 to 50, while the number of concrete services per service candidate set is fixed at 500 and the number of data sets required by each abstract service is fixed at 10. The results of this experiment indicate that the performance of both approach degrade as the number of abstract service increases, however, the GA approach still outperforms the MIP approach.

In Fig. 5 on the next page, we study the performance of both approaches with respect to the number of concrete services in

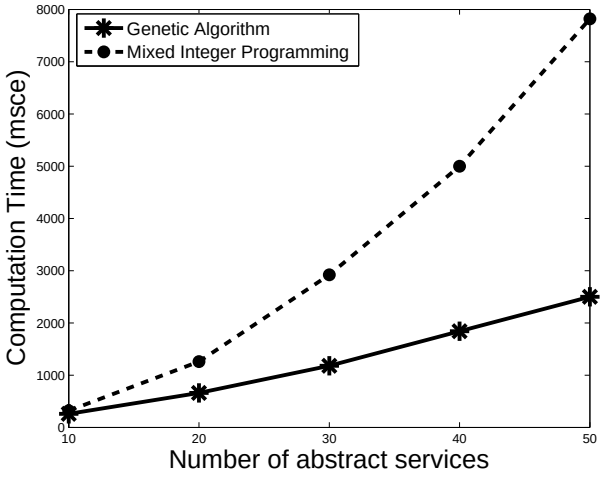

Fig. 4. Performance vs. number of abstract services

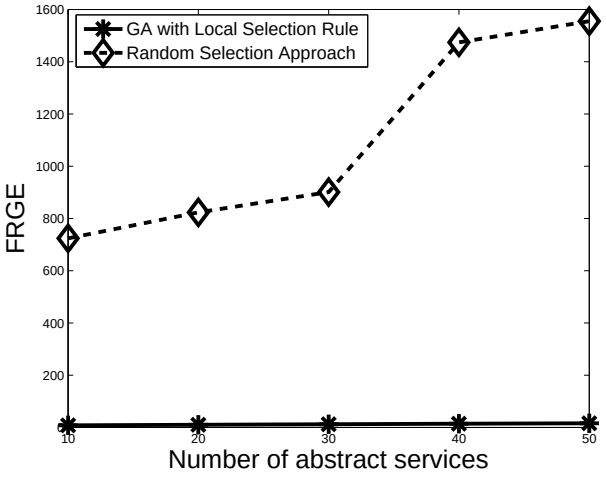

Fig. 6. Performance vs. number of abstract services

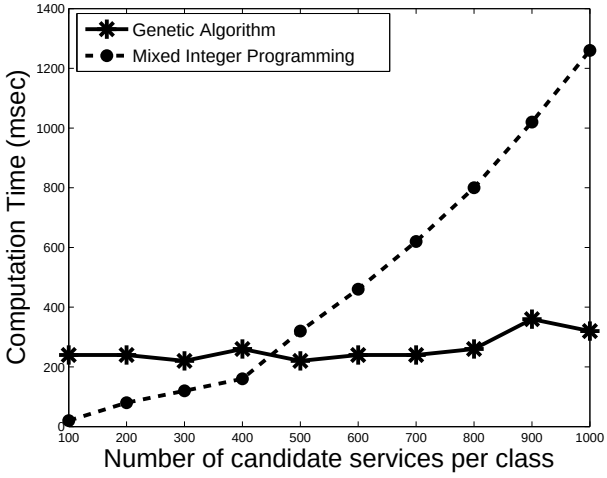

Fig. 5. Performance vs. number of concrete services

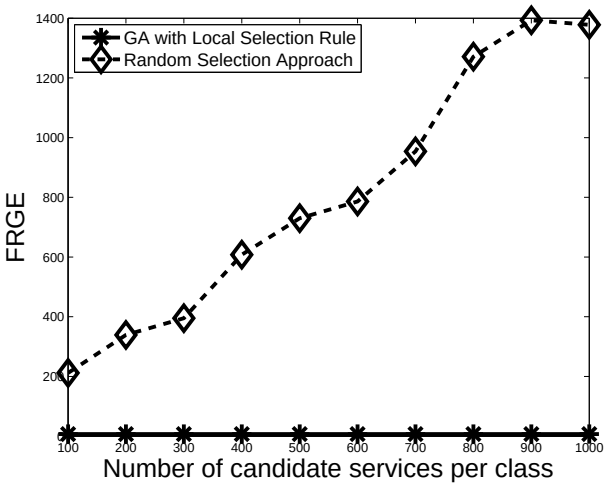

Fig. 7. Performance vs. number of concrete services

the composition. The number of concrete services per service class varies from 100 to 1000, while the number of abstract services is set to 10, and the number of data sets is fixed at 10. By increasing the number of service candidates, the required computation time of the proposed GA increases very slowly, this makes our algorithm more scalable.

When comparing GA with random selection approach, the random selection approach is required to reach the same solution as of GA. In Fig. 6 and Fig. 7 on the next page, we compare the number of generations to reach the best fitness value, namely, the value of 'FRGE' of our modified genetic algorithm using local selection rule with 'FRGE' of random selection approach. Both of GA and random selection approach are required to reach the same solution as of MIP.

In Fig. 6, the value of 'FRGE' for GA is 8, 10, 12, 14 and 16 respectively, and the value of 'FRGE' for random selection approach is 724, 823, 901, 1474 and 1555 respectively when the number of abstract services between 10 and 50. Fig. 6 indicates that both approaches need more generations to reach the best fitness when increasing the number of abstract services, however, the GA with local selection rule outperforms the random selection approach.

In Fig. 7, the value of 'FRGE' for GA is all 7, and the value of 'FRGE' for random selection approach is 212, 339, 395, 608, 730, 786, 954, 1271, 1393 and 1378 respectively when the number of concrete services from 100 to 1000. Fig. 7 indicates that by increasing the number of concrete services, the value of 'FRGE' in the GA with local selection rule will not change but that in the random selection approach increases very quickly. That is to say, to reach the same solution as of MIP, the random selection approach needs more generations.

VI. CONCLUSION

In this study, we presented and evaluated a new modified genetic algorithm to support data-intensive service compo-

sition. The composition problem is modeled as a directed graph. The algorithm adopts the combination of elitism selection and tournament selection, a local selection method, a modified crossover and new mutation operations to find the optimal solution. We investigate how bio-inspired algorithms facilitate data-intensive service composition. Our goal is to reduce the cost of service composition that involves large amounts of data transfer, data placement, and data storage. The service composers, the service providers, and the data providers all need a wise approach to regulate and price their resources. Towards this, we present an economical model of data-intensive service provision and an extensible QoS model. Comparisons with mixed integer programming and random selection approaches show the scalability and effectiveness of our proposed algorithm. Future extensions of the work will develop algorithms for negotiation processes among service composers, service providers and data providers, and to perform a thorough comparison of GA with other approaches such as ant colony optimization algorithms.

REFERENCES

[1] H.H.E. Al-Mistarihi, and C.H. Yong, "Response Time Optimization for Replica Selection Service in Data Grids," *Journal of Computer Science*, vol. 4, no. 6, pp. 487-493, 2008.

[2] M. Alrifai, T. Risse, and W. Nejdl, "A Hybrid Approach for Efficient Web Service Composition With End-To-End QoS Constraints," *ACM Transactions on the Web*, vol. 6, no. 2, pp. 7:1-7:31, 2012.

[3] D. Ardagna, and B. Pernici, "Adaptive Service Composition in Flexible Processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369-384, 2007.

[4] M. Berkelaar, K. Eikland, and P. Notebaert, lpsolve: Open Source (mixed-integer) Linear Programming System. [Online]. Available: http://lpsolve.sourceforge.net/

[5] G. Canfora, M. Penta, R. Espositio, and M.L. Villani, "An Approach for QoS-Aware Service Composition Based on Genetic Algorithms," *Proc. 2005 Conf. Genetic and Evolutionary Computation*, pp. 1069-1075, 2005.

[6] C. Gao, M. Cai, and H. Chen, "QoS-Aware Service Composition Based On Tree-Coded Genetic Algorithm," *Proc. 31st Annual Int'l Computer Software and Applications Conference*, pp. 361-367, 2007.

[7] L. Cao, M. Li, and J. Cao, "Cost-Driven Web Service Selection Using Genetic Algorithm," *Internet and Network Economics*, Lecture Notes in Computer Science, vol. 3828, pp. 906-915, 2005.

[8] M.C. Jaeger, and M. Gero, "QoS-based Selection of Services: The Implementation of a Genetic Algorithm," *Proc. KiVS 2007 Workshop: Service-Oriented Architectures and Service Oriented Computing*, pp. 359-370, 2007.

[9] Y. Ma, and C. Zhang, "Quick Convergence of Genetic Algorithm for QoS-Driven Web Service Selection," *Computer Networks*, vol. 52, no. 5, pp. 1093-1104, April 2008.

[10] M. Milenkovic, E. Castro-Leon, and J.R. Blakley, "Power-Aware Management in Cloud Data Centers," *Proc. 1st Int'l Conf. Cloud Computing (CloudCom '09)*, pp. 668-673, 2009.

[11] J.A. Parejo, P. Fernandez, and A.R. Cortes, "QoS-Aware Services Composition Using Tabu Search and Hybrid Genetic Algorithms," *Actas de Talleres de Ingeniera Del Software y Bases de Datos*, vol. 2, no. 1, pp. 55-66, 2008.

[12] R. Ramacher, and L. Monch, "Cost-Minimizing Service Selection in the Presence of End-to-End QoS Constraints and Complex Charging Models," *Proc. IEEE Ninth Int'l Conf. Services Computing*, pp. 154-161, 2012.

[13] F. Rosenberg, M. Muller, P. Leitner, A. Michlmayr, A. Bouguettaya, and S. Dustdar, "Metaheuristic Optimization of Large-Scale QoS-Aware Service Compositions," *2010 IEEE Int'l Conf. Services Computing*, pp. 97-104, 2010.

[14] J. Shen, G. Beydoun, S. Yuan, and G. Low, "Comparison of Bio-Inspired Algorithms for Peer Selection in Services Composition," *Proc. IEEE Int'l Conf. Services Computing*, pp. 250-257, 2011.

[15] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, "Multi-Tiered on Demand Resources Scheduling for VM-Based Data Center," *Proc. 9th IEEE/ACM Int'l Symp. Cluster Computing and the Grid*, pp. 148-155, 2009.

[16] M. Srinivas, and L.M. Patnaik, "Genetic Algorithms: A Survey," *Computer*, vol. 27, no. 6, pp. 17-26, 1994.

[17] S. Su, C. Zhang, and J. Chen, "An Improved Genetic Algorithm for Web Services Selection," *Distributed Applications and Interoperable Systems*, Lecture Notes in Computer Science, vol. 4531, pp. 284-295, 2007.

[18] M. Tang, and L. Ai, "A Hybrid Genetic Algorithm for the Optimal Constrained Web Service Selection Problem In Web Service Composition," *Proc. the 2010 World Congress on Computational Intelligence*, pp. 268-275, 2010.

[19] L. Wang, J. Luo, J. Shen, and F. Dong, "Cost and Time Aware Ant Colony Algorithm for Data Replica in Alpha Magnetic Spectrometer Experiment," *IEEE International Congress on Big Data*, pp. 254-261, 2013.

[20] L. Wang, J. Shen, and J. Yong, "A Survey on Bio-Inspired Algorithms for Web Service Composition," *IEEE 16th Int'l Conf. Computer Supported Cooperative Work in Design*, pp. 569-574, 2012.

[21] L. Wang, J. Shen, "Towards Bio-Inspired Cost Minimisation for Data-intensive Service Provision," *IEEE Int'l Conf. Services Economics*, pp. 16-23, 2012.

[22] L. Wang, J. Shen, C. Di, Y. Li, and Q. Zhou, "Towards Minimizing Cost For Composite Data Intensive Services," *IEEE 17th Int'l Conf. Computer Supported Cooperative Work in Design*, pp. 293-299, 2013.

[23] L. Wang, H. Siegel, V. Roychowdhury, and A. Maciejewski, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm Based Approach," *Journal of Parallel and Distributed Computing*, vol. 47, no. 1, pp. 8-22, 1997.

[24] M. Winter, "Data Center Consolidation: A Step Towards Infrastructure Clouds," *Cloud Computing*, Lecture Notes in Computer Science, vol. 5931, pp. 190-199, 2009.

[25] Z. Ye, X. Zhou, and A. Bouguettaya, "Genetic Algorithm Based QoS-Aware Service Compositions in Cloud Computing," *Database Systems for Advanced Applications*, Lecture Notes in Computer Science, vol. 6588, pp. 321-334, 2011.

[26] K.P. Yoon, and H.C. Land, "Multiple Attribute Decision Making: An Introduction," *Sage university papers series (quantitative applications in the social sciences)*, Thousand Oaks, CA, Sage Publications, 1995.

[27] S. Yuan, J. Shen, and A. Krishna, "Ant Inspired Scalable Peer Selection in Ontology-Based Service Composition," *Proc. Fifth World Congress on Services*, pp. 95-102, 2009.

[28] C. Zhang, S. Su, and J. Chen, "Genetic Algorithm on Web Services Selection Supporting QoS," *Chinese Journal of Computers*, vol. 29, no. 7, pp. 1029-1037, 2006.

[29] L. Zhang, and B. Li, "Requirements Driven Dynamic Services Composition For Web Services And Grid Solutions," *Grid Computing*, vol. 2, no. 2, pp. 121-140, June 2004.

[30] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311-327, 2004.