Future Generation Computer Systems 86 (2018) 1008–1018

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

QoS-aware Big service composition using MapReduce based evolutionary algorithm with guided mutation

Chandrashekar Jatoth [a,b], G.R. Gangadharan [a,*], Ugo Fiore [c], Rajkumar Buyya [d]

[a] Institute for Development and Research in Banking Technology, Hyderabad, India
[b] SCIS, University of Hyderabad, Hyderabad, India
[c] Department of Molecular Medicine and Medical Biotechnologies, Federico II University, Italy
[d] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, The University of Melbourne, Australia

HIGHLIGHTS

- The problem of producing service composition with an optimal QoS attribute that satisfies the customer requirements is a complex and challenging issue, especially in a Big service environment.
- Proposal of a novel MapReduce-based Evolutionary Algorithm with Guided Mutation .
- An optimal service composition method in Big Service Environment provides the best performance concerning feasibility and scalability.
- By performing *T*-test and Wilcoxon signed rank test at 1% level of significance, we observed that our proposed method outperforms other methods.

ARTICLE INFO

Article history:
Received 1 December 2016
Received in revised form 30 March 2017
Accepted 16 July 2017
Available online 21 July 2017

Keywords:
Web service
Big data
Quality of Service (QoS)
MapReduce
Meta-heuristic algorithm

ABSTRACT

Big services are the collection of interrelated services across virtual and physical domains for analyzing and processing big data. Big service composition is a strategy of aggregating these big services from various domains that addresses the requirements of a customer. Generally, a composite service is created from a repository of services where individual services are selected based on their optimal values of Quality of Service (QoS) attributes distinct to each service composition. However, the problem of producing a service composition with an optimal QoS value that satisfies the requirements of a customer is a complex and challenging issue, especially in a Big service environment. In this paper, we propose a novel MapReduce-based Evolutionary Algorithm with Guided Mutation that leads to an efficient composition of Big services with better performance and execution time. Further, the method includes a MapReduce-skyline operator that improves the quality of results and the process of convergence. By performing *T*-test and Wilcoxon signed rank test at 1% level of significance, we observed that our proposed method outperforms other methods.

1. Introduction

Big services are collections of interrelated web services for handling and dealing with big data, having the properties of customer focus, massiveness, complexity, heterogeneity, convergence, credibility, and value across physical and virtual domains [1]. Big service composition consists in aggregating services from multiple domains to create a composite or aggregated, big service that addresses customer requirements [1]. A composite big service is complex in nature and should be able to scale with the growth of data volumes and to improve adaptability and maintainability of diversified data.

Several web services provide similar functionality with different non-functional properties, specified as Quality of Service (QoS) attributes. Individual services are selected based on their QoS attributes so that the constraints and preferences specific to each service composition can be satisfied by the overall QoS of the composite service. The QoS of a composite service is defined by attributes such as cost, response time, availability, reliability, and throughput. However, producing a service composition with an optimal QoS value that satisfies the customer requirements

* Corresponding author.
E-mail addresses: jcshekar@idrbt.ac.in (C. Jatoth), GRgangadharan@idrbt.ac.in (G.R. Gangadharan), ufiore@unina.it (U. Fiore), rbuyya@unimelb.edu.au (R. Buyya).

http://dx.doi.org/10.1016/j.future.2017.07.042
0167-739X/

is a complex task, especially in a Big service environment. QoS-aware service composition is as a NP-hard optimization problem. Numerous studies inclusive of exact, heuristic and meta-heuristic algorithms have been proposed to solve QoS-aware web service composition [2]. Genetic algorithms (GA) [3–7] have been widely used for solving QoS-aware service composition. The selection efficiency is, however, related with the number of candidate services [8].

As the computational intelligence algorithms require more iterations and more execution time to find the near-optimal solution, parallelization becomes attractive [9]. An efficient parallelized algorithm is faster than its sequential counterpart. The parallel algorithm should handle large amounts of data and scale well with a growing number of computing nodes. However, depending on the nature of the algorithm, some problems can be encountered such as inefficient communication, unfair load balancing, and node failure, which make the process of scaling of the algorithm to large numbers of processors very difficult.

In this paper, we propose a novel MapReduce-based Evolutionary Algorithm with Guided Mutation (MR-EA/G) for Big service composition. The salient contributions of this paper are as follows:

- A novel method for pre-selection of services offering the QoS value required for the user's satisfaction, which improves the quality of the result and speeds up the process of convergence using a novel MapReduce-Skyline operator.
- A novel MapReduce-based Evolutionary Algorithm with Guided Mutation (MR-EA/G) that reduces computational time and increase convergence rate.
- An empirical analysis illustrating the performance of MR-EA/G in terms of feasibility, scalability, and optimality with different QoS attributes for solving Big service composition.

The rest of the paper is organized as follows. Section 2 describes the modeling of QoS-aware Big service composition. The proposed method MR-EA/G is described in Section 3. Section 4 presents performance evaluation. Related work is discussed in Section 5 followed by concluding remarks in Section 6.

2. Modeling of QoS-aware big service composition

A composite service T is a fusion of m multiple tasks (or abstract services) $T = \{t_1, \ldots, t_m\}$. Each task t_i can be realized by one service in a specific set of k_i candidate services $C_i = \{s_1^i, \ldots, s_{k_i}^i\}$. The sets C_i of candidate services are subsets of the set of available concrete services $S = \{s_1, \ldots, s_n\}$. The services in the set C_i are similar in functionality but may differ in their QoS attributes.

Let $Q = \{q_1, \ldots, q_p\}$ be the QoS attributes. For a QoS attribute $q \in Q$, denote with $q(s_i)$ the value of that QoS attribute for a particular service s_i.

Let $w : Q \mapsto [0, 1]$ be a function giving the preference for each attribute: $w(q)$ specifies the user preference for attribute $q \in Q$.

Let $C = \{c_1, \ldots, c_p\}$ be a set of QoS constraints given by the user, where c_k is a constraint on QoS attribute k in a composite service.

QoS attributes can be considered as positively and negatively monotonic [10,11]. The process of service composition should map to higher values for positively monotonic QoS attributes and lower values for negatively monotonic QoS attributes. For example, throughput, availability, and reliability are positively monotonic attributes, while price and response time are negatively monotonic attributes. The list of QoS attributes and their description is reported in Table 1. Before evaluating the utility function (or fitness function) of a composite service, we calculate the normalized value of each QoS attribute as follows [10,12,13]:

$$v_q(x) = \begin{cases} 1 & \text{if } q^{\max} = q^{\min} \\ \dfrac{q^{\max} - x}{q^{\max} - q^{\min}} & \text{if } q \text{ is negative} \\ \dfrac{x - q^{\min}}{q^{\max} - q^{\min}} & \text{if } q \text{ is positive} \end{cases} \tag{1}$$

where $q^{\max}$ and $q^{\min}$ are maximum and minimum values of the QoS attribute q for all candidate services and x is the attribute value of a service respectively. With a slight abuse of notation, $v_q(i)$ will be used to mean $v_q(q(i))$.

The values of global QoS attributes are the aggregate values of QoS attributes of selected services for each task t_i, $1 \le i \le m$. The best composition is obtained by maximizing the utility function values of the global QoS attributes according to the user's preference. The user's preference is expressed as weights w_i where i is the QoS attribute. These user preferences are obtained by Analytical Hierarchy Process (AHP) method. The scale that we use for pairwise comparison is the one adopted by Saaty [14], ranging from 1 to 9 with the meaning reported in Table 2. The pairwise comparison of attributes was made with the help of domain experts. The weights of each criterion are described in Table 3. To check the consistency of the calculated weights, we obtain the consistency ratio (CR) ≤ 0.1. This ratio tells that our matrix is consistent and weights are valid.

Let A_q be the aggregation functions of QoS attribute q which computes the global value of q for the composition. Generally, a utility function evaluates the multi-dimensional QoS attributes of a composite service [12,10,13]. In this paper, we use the Simple Additive Weight model as a utility function. The utility functions of the normalized attributes are assumed to be linear. Thus the goal of the optimization is to find X that maximizes the global utility function computed as follows (similar to the scheme proposed in [10]):

$$\begin{aligned} & d(X) = \sum_{i=1}^{m} \sum_{q \in Q} w(q) A_q \left(v_q(i)\right) \\ & \text{subject to constraints:} \qquad A_q \left(v_q(i)\right) \le A_q \left(v_q^c(i)\right); \\ & \text{having} \sum_{q \in Q} w(q) = 1 \end{aligned} \tag{2}$$

where $A_q \left(v_q^c(i)\right)$ denotes the global constraints given by the users and $v_q(i)$ computed by the aggregation functions. The global QoS attribute is computed by recursively applying the aggregation functions to the building blocks that form the structure of the composition [3]. For example, the global response time would be the sum of response times of individual candidate services when the services are executed in sequence and would be the maximum of response times of individual services when the services are executed in parallel. In practice, diverse composition structure, for example, sequential, parallel, conditional, and loop can be engaged in a composition plan. In our paper, we have considered the tasks to be sequential in the composition plan and other plans are transformed into sequential model [11,15]. The aggregation functions for each of the QoS attributes considered are as follows:

$$A_{\mathrm{Pr}}(i) = \sum_j \mathrm{Pr}\left(s_j^i\right), \qquad A_{\mathrm{Th}}(i) = \min_j \mathrm{Th}\left(s_j^i\right),$$

$$A_{\mathrm{Av}}(i) = \prod_j \mathrm{Av}\left(s_j^i\right),$$

$$A_{\mathrm{Re}}(i) = \prod_j \mathrm{Re}\left(s_j^i\right), \qquad A_{\mathrm{Rt}}(i) = \sum_j \mathrm{Rt}\left(s_j^i\right)$$

Table 1
List of QoS attributes and their description.

S.No	QoS attributes name	Description
1	Price	The price that a service requester has to pay for invoking a service.
2	Throughput	The total number of web service invocations possible in a given amount of time (measured in invokes/sec).
3	Availability	The probability that a service is available during the request (measured in percentage).
4	Reliability	The probability that a request is correctly responded within the maximum expected time (measured in percentage).
5	Response time	The time interval between the moments when a user requests the service and when the user receives the response (measured in milliseconds).

Table 2
Scales for comparison matrix of QoS attributes.

Intensity of importance	Definition
1	Equal importance
3	Moderate importance of one over another
5	Essential or strong importance
7	Very strong importance
9	Extreme importance
2, 4, 6, 8	Intermediate values between the two adjacent judgments

Table 3
Priority weights of QoS attributes.

	Availability	Response time	Reliability	Price	Throughput	Weights vector
Availability	1.00	3.00	2.00	3.00	2.00	34.81%
Response time	0.33	1.00	2.00	3.00	5.00	32.50%
Reliability	0.50	0.50	1.00	0.50	0.33	9.64%
Price	0.33	0.33	2.00	1.00	2.00	14.17%
Throughput	0.50	0.20	3.00	0.50	1.00	13.73%

Table 4
Aggregation functions of QoS attributes.

QoS attribute name	Sequence	Parallel	Loop	Conditional
Price (Pr)	$A_{\text{Pr}} = \sum_{i=1}^{n} Pr\left(s_j^i\right)$	$A_{\text{Pr}} = \sum_{i=1}^{n} Pr\left(s_j^i\right)$	$A_{\text{Pr}} = k * \sum_{i=1}^{n} Pr\left(s_j^i\right)$	$A_{\text{Pr}} = \sum_{i=1}^{n} Pr\left(s_j^i\right) * P_i$
Throughput (Th)	$A_{\text{Th}} = min_{i=1}^{n} Th\left(s_j^i\right)$	$A_{\text{Th}} = min_{i=1}^{n} Th\left(s_j^i\right)$	$A_{\text{Th}} = k * \prod_{i=1}^{n} Th\left(s_j^i\right)$	$A_{\text{Th}} = P_i * \prod_{i=1}^{n} Th\left(s_j^i\right)$
Availability (Av)	$A_{\text{Av}} = \prod_{i=1}^{n} Av\left(s_j^i\right)$	$A_{\text{Av}} = \prod_{i=1}^{n} Av\left(s_j^i\right)$	$A_{\text{Av}} = k * \prod_{i=1}^{n} Av\left(s_j^i\right)$	$A_{\text{Av}} = P_i * \prod_{i=1}^{n} Av\left(s_j^i\right)$
Reliability (Re)	$A_{\text{Re}} = \prod_{i=1}^{n} Re\left(s_j^i\right)$	$A_{\text{Re}} = max_{i=1}^{n} Re\left(s_j^i\right)$	$A_{\text{Re}} = k * \prod_{i=1}^{n} Re\left(s_j^i\right)$	$A_{\text{Re}} = P_i * \prod_{i=1}^{n} Re\left(s_j^i\right)$
Response time (Rt)	$A_{\text{Rt}} = \sum_{i=1}^{n} Rt\left(s_j^i\right)$	$A_{\text{Rt}} = min_{i=1}^{n} Rt\left(s_j^i\right)$	$A_{\text{Rt}} = k * \sum_{i=1}^{n} Rt\left(s_j^i\right)$	$A_{\text{Rt}} = P_i * \sum_{i=1}^{n} Rt\left(s_j^i\right)$

where s_j^i is the service selected for subtask t_i and abbreviations are used for the QoS attributes. The aggregation functions of QoS attributes are described in Table 4, where P_i represents the probability of a branch i and k represents the number of loops. The objective of the composition is to select an optimal candidate service among them to resolve the task t_i, since the selected candidate service influences the quality of composition.

3. MapReduce-based EA/G for big service composition

This section describes the basic concepts of MapReduce, modified Evolutionary Algorithm with Guided Mutation (EA/G) approach, and MapReduce-based EA/G in detailed for Big service composition.

3.1. MapReduce

The MapReduce (MR) model is intended for parallel and distributed processing of large datasets in data intensive applications [16,17]. The paradigm behind MapReduce is to split data into pieces or into equal splits that can be processed in parallel and then gather the results to produce the final output. Conceptually, this model comprises two primitives: Map() function and Reduce() function. The Map() function takes an input as key/value pairs and applies an operator on each key/value pair to construct a set of intermediate key/value pairs. The Reduce() function takes as input intermediate key/value pairs having the same key and produces final key/value pairs.

In our approach, Map() creates a list of candidate services with QoS attributes. For a single service, a set of QoS attributes is sent to the Reduce() function, which computes the best candidate service among the available services. The simplified execution flow of MapReduce-based Big service composition is shown in Fig. 1.

3.2. Modified EA/G approach

The Evolutionary Algorithm with Guided Mutation (EA/G) [18] is a combination of both conventional Genetic Algorithm (GA) and Estimation of Distribution Algorithms (EDA) that uses both global information and local information on solutions found so far to generate offsprings. Global information is used to build a probability model of promising solutions. An offspring is produced by sampling from this model, in combination with partial reuse of parent solutions.

An initial solution is generated randomly. Guided mutation produces an offspring from the parent solution based on a probability

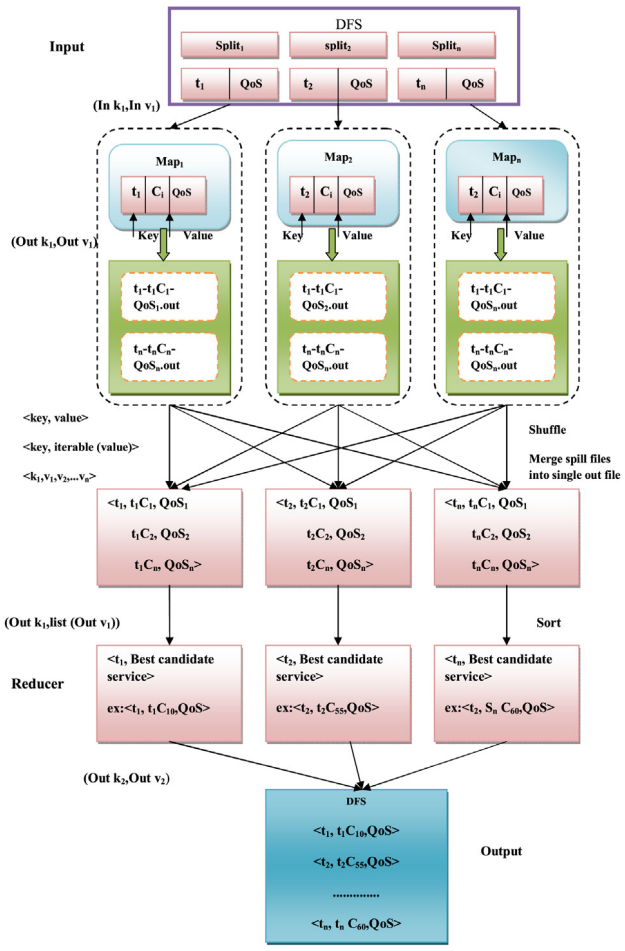

Fig. 1. MapReduce execution flow for big service composition.

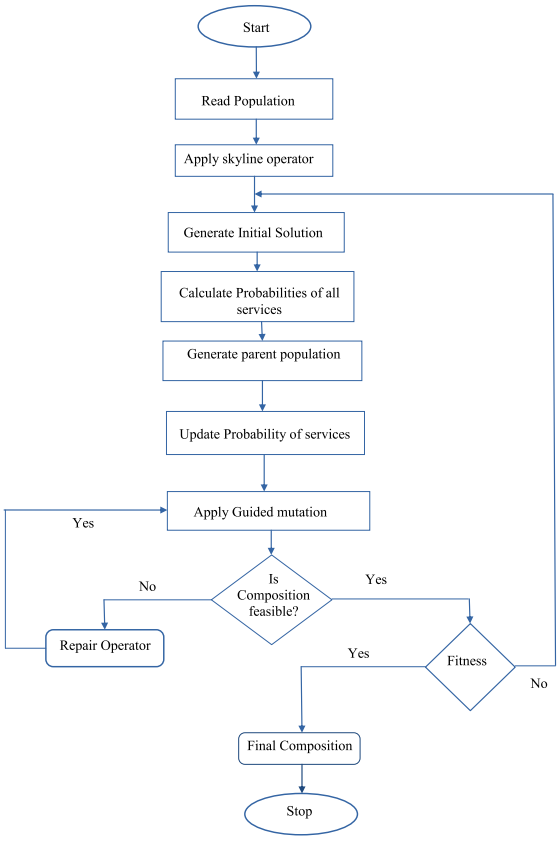

Fig. 2. Flow chart of the modified EA/G algorithm.

guidance model, but the newly generated solution retains a user-specified percentage of the elements of the parent. In such a way, the similarity between an offspring and a parent can be controlled, and the resultant solution can fall in or close to a promising area which is characterized by the probability model.

The problem with this model is the random generation of the initial solution that results in a low convergence rate. To overcome this issue, we propose a modified evolutionary algorithm with guided mutation, where the search space is reduced to avoid redundancy (filter-out redundant services among available services), and the initial solution is generated on the reduced search space using a novel skyline operator. The modified evolutionary algorithm with guided mutation algorithm workflow is shown in Fig. 2.

3.2.1. Solution encoding

The goal is to find a service composition with top k candidate services from the abstract services. The chromosome encoding model is shown in Fig. 3. In Fig. 3, $t_1, \ldots, t_m$ are the abstract services of the composition. The chromosome represented in Fig. 3 contains, for each service t_i, an integer value which represents a concrete service chosen in the set C_i of the candidate services. In the chromosome of Fig. 3, the abstract service t_1 selects the candidate service s_4, the abstract service t_2 selects the candidate service s_1, the abstract service t_3 selects the candidate service s_6, and etc. to include in the composition.

3.2.2. Initial solution using skyline operator

Generally, in genetic algorithms, the initial population is generated randomly. The random selection of a solution spans the entire search space of $\prod_{k=1}^{m} n_k$ possible combinations of one out of n_k alternative services for each of the m abstract services. This results in an initial solution having possibly a low quality. In order to overcome this issue, we propose a novel skyline operator to reduce the search space by selecting only dominant services in a given set, thus improving the convergence speed and quality of the solution.

Let V be a set of candidate services C_i where $C_i = \{s_1^i, \ldots, s_{k_i}^i\}$ in all abstract services t_i, where $i = 1, 2. \ldots, m$. Initially, all the candidate services are assigned to $\Re$. From $\Re$, the first candidate service of initial population is chosen based on the fitness value (described in Section 2). The candidate service C_i with the maximum fitness is added to the initial population, which is denoted by τ. From the set $\Re$, the candidate services that have fitness values nearer to the candidate service C_i in τ are selected for further processing and are added to ζ and are removed from $\Re$. ζ is added to χ which represents the shortlisted services. For services in χ, we check whether there are candidate services from τ, with the nearest fitness values. In such selected candidate services, we find a candidate service with the nearest fitness value among the selected services to add it to τ and delete it from χ. This process is repeated until all the services in the set $\Re$ are covered. The final τ list becomes the initial solution for the given input.

3.2.3. Initial probabilities

In our modified evolutionary algorithm with guided mutation, univariate marginal distribution is applied for the estimation of the distribution of solutions in the search space. A probability vector $P = (p_1, \ldots, p_K)$ is used for characterizing the solution distribution, where $K = |V|$ is the number of candidate services considered. The vector P is initialized starting from the N_c initial solutions. After generating the initial solutions, p_i is set to the

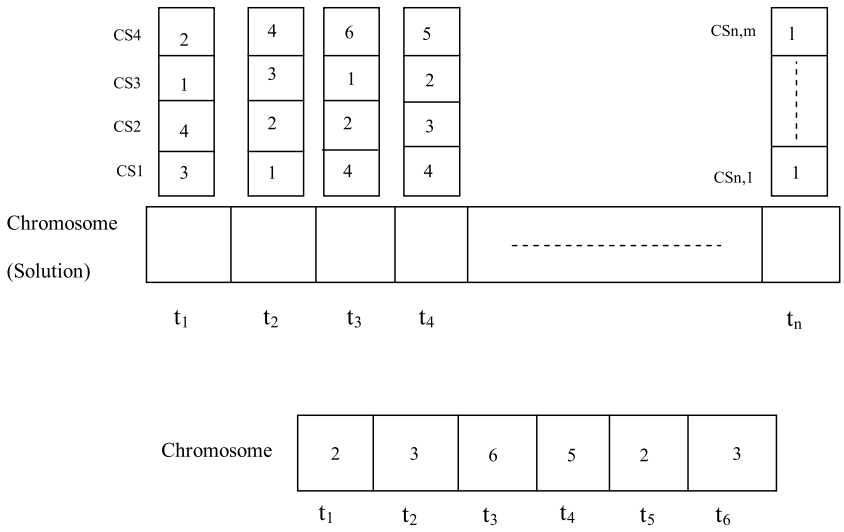

Fig. 3. Chromosome encoding model.

Algorithm 1 initial population generation using skyline operator

Input: V:set of candidate services
Output: dominating set of candidate services.
$\Re$: set of candidate services in each abstract service.
τ: set of services selected for initial solution.
ζ: neighbor services of τ.
χ: neighborhood set of τ.
d_t: Possible dominant set.
for *each population* **do**
  $\Re \leftarrow V$;
  $\chi \leftarrow \phi, \zeta \leftarrow \phi, d_t \leftarrow \phi$;
  $V \leftarrow random(\Re)$;
  Insert V into τ //dominating set;
  Delete V from $\Re$;
  while $\Re \neq \phi$ **do**
    Compute neighbor services (ζ) for V;
    for *each service j in* $\Re$ **do**
      if $V_{fitness}$ - $j_{fitness} < c1$ **then**
        $\zeta \leftarrow j$;
      end
    end
    $\chi \leftarrow \zeta$;
    Select neighbor services which has at least one service with nearest fitness as neighbor ;
    for *each service k in* χ **do**
      for *each service j in* $\Re$ **do**
        if $k_{fitness} - j_{fitness} < c1$ **then**
          $\zeta \leftarrow j$;
        end
      end
      if *neighbor is present* **then**
        $d_t \leftarrow k$;
      end
    end
    χ.remove(service);
    $\chi \leftarrow \zeta$;
    Select a service from χ **with fitness value nearest to** τ;
    $\tau \leftarrow service$;
  end
end

frequency of occurrence of the candidate service s_i, i.e., counting the number of solutions containing s_i and dividing by N_c. The vector P is used in the guided mutation operator, and it is further updated after every new generation.

3.2.4. Probability update

At each generation, a parent population is formed by selecting the best $N_c/4$ solutions from the current population set. Where N_c is an active solution. The parent population is used for updating P. **The pseudocode for probability updation is described in Algorithm 2**, where $\lambda \in [0, 1]$ is the learning rate that governs the parent population contribution. With high values of λ, the updated P reflects more the characteristics of the parent population and vice versa. The change in the probability of a candidate service depends on the frequency of occurrence of that service in the parent population.

Algorithm 2 Update Probability

Input: initial P, parent population, population.
Output: updated P.
for *each candidate service* s_i *in population* **do**
  $n \leftarrow$ number of parent population solutions containing s_i;
  $p'_i = (1 - \lambda) * p_i + (\lambda * (4n/N_c))$;
end
$P \leftarrow P'$

3.2.5. Guided mutation

The Guided Mutation operator combines the global statistical information in P and the generated solutions to generate new solutions.

Algorithm 3 explains the process of generating a new solution, where $\beta \in [0, 1]$ and d_t is the new solution generated whose services are either sampled from the model given by P or retain those of the best solution in the parent. This process is controlled by a parameter β is used to control the contributions of P and parent population. As the value of β decreases, the parent population contribution to the new solution increases whereas that of P decreases and vice-versa. The generated solution d_t may not be valid. Hence it needs to be checked for feasibility and, if required, a repair operator is applied to it.

3.2.6. Repair operator

The repair operator is applied only on infeasible solutions. In this repair operator, all the services from $\Re$ that are not present in the χ list are selected and $\Re$ list is reinitialized with those selected services. A service from $\Re$ is selected randomly, and its nearest neighborhood is calculated. The repair operator selects a service from this set and adds to the infeasible solution set. Then, the

Algorithm 3 Guided Mutation

Input: population, P, and best solution
Output: Generated solution d_t
initialize β ;
for *each service* s_i *in population* **do**
  generate a random value $r \in [0, 1]$;
  if $r < \beta$ **then**
    if $p_i <$ *threshold_probability* **then**
      $d_t \leftarrow d_t \cup s_i$;
    end
    else
      if $s_i \in$ *_bestsolution* **then**
        $d_t \leftarrow d_t \cup s_i$;
      end
    end
  end
end

selected service is deleted from $\Re$. This process is repeated until the set becomes feasible, i.e., until $\Re$ becomes null. The selection of a service is based on the probability threshold value or random selection. This helps to maintain the diversity of the population. The pseudocode of repair operator is shown in Algorithm 4.

Our proposed Modified Evolutionary Algorithm with Guided mutation (EA/G) (see Algorithm 5) uses the conventional EA/G with a novel skyline operator to generate solutions at faster convergence rate. In this algorithm, the initial population is generated by using the skyline operator. The skyline operator has the ability to remove redundant services. Further, N_c number of initial solutions are generated. The probability of each candidate service is computed using these solutions generated. Later, the probability of each candidate service is updated using the best $N_c/4$ number of parent populations. The composition is generated using the modified EA/G based on both parent solutions and P. Further, the composition generated is tested for feasibility. If the composition generated is not feasible, then the repair operator is applied to make it feasible. This process is repeated for 'N' number of times until the optimal composition with the best fitness is obtained. The pseudocode of modified EA/G algorithm is presented in Algorithm 5. The evolutionary strategy flow diagram of our proposed approach is illustrated in Fig. 2.

Algorithm 4 Repair Operator

Input: d_t, population, *Prob*
Output: d_t
while $W_n \neq \phi$ **do**
  $v \leftarrow$ random service from population;
  if v *not in* d_t **then**
    if $V\_$ *probability* $<$ *threshold_ probability* **then**
      $d_t \leftarrow V$;
      delete V from population;
    end
  end
end
return d_t;

3.3. MapReduce based EA/G

This sub-section describes the MapReduce-based implementation of our Modified EA/G algorithm to solve the challenge of big service composition with scalability and robustness. The algorithm works in 3 phases: Initialization phase, MapReduce phase, and Repair phase (as illustrated in Fig. 4).

3.3.1. Initialization phase

In order to reduce the search space of candidate services and filter-out the optimal candidate services for each abstract service,

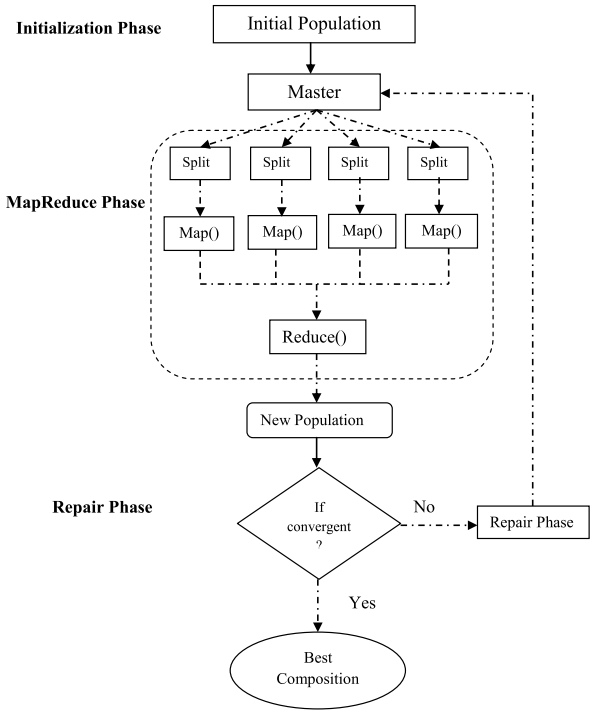

Fig. 4. Flowchart of MR-EA/G.

Algorithm 5 Modified EA/G

Input: population, P
Output: best service composition
iterations $\leftarrow 0$;
do
  iterations $\leftarrow$ *iterations* $+ 1$;
  Generate N_c initial solutions for g by using Algorithm 1;
  Calculate initial probabilities for all the services in generation;
  Generate the parent population;
  Update P based on parent population by using Algorithm 2;
  do
    Apply Guided Mutation by using Algorithm 3;
  while *services* $\neq$ *end*;
  Verify feasibility of the generated composition;
  if *composition is not feasible* **then**
    Apply repair operator on obtained solution using Algorithm 4;
  end
  Reinitialize generation g;
while *iterations* $\leq N$;

we use the following skyline operator in the map reduce framework.

Let s_1^i and s_2^i be two candidate services for the abstract service t_i, considered with their normalized QoS parameters. The Pareto dominance is described as follows. The service s_1^i is said to dominate s_2^i if and only if the following two conditions are true:

$$\begin{cases} \forall q \in Q,\ v_q(1) \geq v_q(2) \\ \exists q \in Q : v_q(1) > v_q(2) \end{cases}$$

that is, s_1^i is better than or equal to s_2^i for all QoS attributes and s_1^i is strictly better than s_2^i on at least two QoS attributes.

We adopted the block elimination method [19] to process our skyline method to reduce the redundant services in service composition. The service files are stored as a $\langle key, value \rangle$ pair structure in a distributed file system, where the key is the service ID and value is the service information. This file is used as input to the MapReduce job in MapReduce phase. The structure of $\langle key, value \rangle$ pair is used

I: S_n	S_nC_n: Candidate service; C_n: QoS attribute; Fitness(n)
Key	**Value**

Fig. 5. Representation of subpopulation.

in MR-EA/G is shown in Fig. 5. The components of the services are separated by a semicolon. These candidate services are given as initial population. From the given population, the initial solution is generated by evaluating the fitness of the services. Initially, the service with the maximum fitness value is selected and added to the solution. Then, we proceed further by selecting the nearest fitness values with respect to the prior service.

3.3.2. MapReduce phase

This section describes the iterative process of MapReduce jobs, where each MapReduce job represents one iteration in MR-EA/G algorithm. The result of each MapReduce job is an updated population. This updated population is used as input to the next MapReduce phase.

The master splits the data into $n = 12$ splits based on replication factor of 3 and stored into the distributed file system, where the number of map and reduce tasks is 1. Hence, the swarm is divided into q number of populations. Then, each map task is updated with its subpopulation. These subpopulations are stored as $\langle key, value \rangle$ pairs, where the key is the abstract service and the value is the candidate service with its QoS attributes. The output of the different tasks with their key is sent to the Reduce() function as its input.

The reducer takes all information about each abstract services at a time by using parallel batch processing. Then, the reducer combines all information of iterable subpopulation. All these subpopulations are stored in a list. The entire list is passed to MR-EA/G algorithm. The MR-EA/G algorithm generates the output as (out key, dominant population) or dominant population for each service and the best optimal service composition with scalability and robustness.

3.3.3. Repair phase

The solution generated from MapReduce phase is further checked for feasibility. A solution is called infeasible if it has low diversity and evolution. Such solutions are passed to repair operator to enhance their diversity and evolution where infeasible solutions are repaired by rechecking and inserting left out candidate services in $\Re$ list. The repair operator is carried on a particular solution until it becomes feasible.

If a solution is feasible, then it is directly passed to next MapReduce job else it is given to repair phase. The resulting solution is given to the next MapReduce job, replacing the previous population in the distributed file system.

4. Performance evaluation

Our proposed MapReduce-based Evolutionary Algorithm with Guided Mutation (MR-EA/G) is implemented using Java on a Hadoop cluster consisting of 12 nodes provided with Ubuntu 14.10 (OS), 16 GB RAM, Intel I7 processor, and 1TB Memory space. One node is set up as a master node and the remaining 11 nodes set up as slave nodes. For experimental analysis, we considered 100 abstract services and each consisting 10 000 candidate services. For the purpose of performance evaluation of MR-EA/G, we require the input dataset consists of QoS attributes such as availability, cost, throughput, response time, and reliability. The

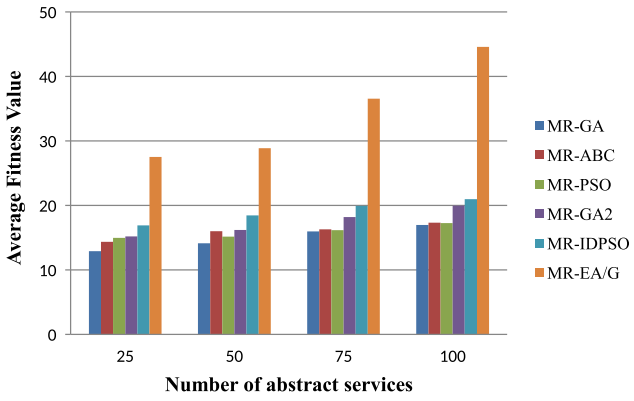

Fig. 6. Average fitness values for 500 candidate services.

dataset used in our experiments is synthetically generated (similar to [20,21,10]). We tested our QWS dataset through Shapiro–Wilk test [22] to determine whether our QWS dataset is normally distributed or not. The test results revealed that the given dataset is normally distributed. The weights of QoS attributes were set to (34.81, 14.17, 13.73, 32.50, 9.64) based on user preferences and AHP with the MNV (mean of normalized values) method. To produce high-quality solutions for a tight convergence settings, we adopt $N_c = 16$, $\beta = 0.035$, $\lambda = 0.34$, probability threshold = 0.55, $C_1 = 1.58$, and fitness threshold = 0.29 (used in checking feasibility of solution) for our proposed approach. All of these values are selected empirically after several experiments.

We compare our proposed MapReduce-based EA/G with MapReduce-based Genetic Algorithm (MR-GA) [23], MapReduce-based Artificial Bee Colony (MR-ABC) [24], MapReduce-based Particle Swarm Optimization (MR-PSO) [25], MapReduce-clonal selection based Genetic Algorithm (MR-GA2) [5], and MapReduce-based Improved Discrete Particle Swarm Optimization (MR-IDPSO) [26]. To evaluate the optimality (average fitness value) and the computation time of our proposed approach, we consider the following three scenarios: (i) Evaluating average fitness values by varying abstract services and candidate services, (ii) Evaluating average fitness values by varying number of iterations, and (iii) Evaluating execution time by varying candidate services.

Figs. 6–9 illustrate the average fitness values with the number of abstract services as 25, 50, 75, and 100, while for each abstract service, the number of candidate services ranges as 500, 1000, 5000, and 10 000. As observed in Figs. 6–9, the average fitness values of abstract services rise with the increase in the number of abstract services. In our proposed approach, this increase is exponential, while the increase is linear in other approaches. The proposed approach uses a probability method to generate offsprings, which in turn, ensures the distribution of promising solutions in each generation. Instead of directly using the position values, our approach updates the probability values for each generation based on the global statistical information and uses the guided mutation. Thus, our proposed approach performs better than the other approaches. Due to the randomness of algorithms, these algorithms are run for 30 times independently.

We evaluated the average fitness values for 100 abstract services with respect to candidate 100, 500, 1000, 5000, and 10 000 using various approaches as shown in Fig. 10. In Fig. 10, we observe that the best average fitness value obtained by our proposed approach for 100 abstract services is 65.58824, whereas the best average fitness values found by MR-GA, MR-GA2, MR-PSO, MR-IDPSO, and MR-ABC are 19.3241, 20.8705, 21.7689, 24.4896, 29.3672 respectively. Because our proposed approach uses a probability method that ensures the distributing of promising solutions

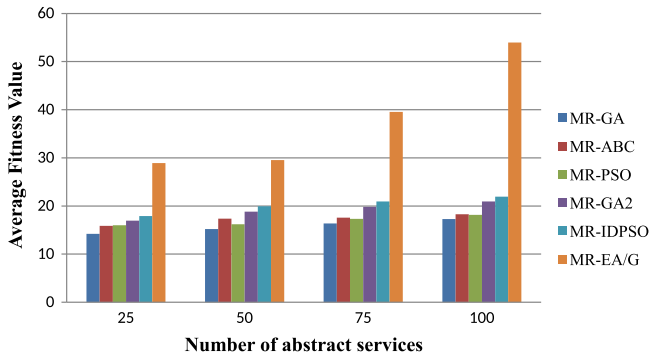

Fig. 7. Average fitness values for 1000 candidate services.

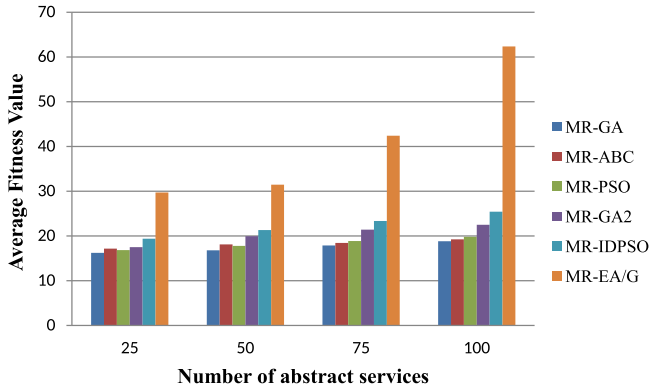

Fig. 8. Average fitness values for 5000 candidate services.

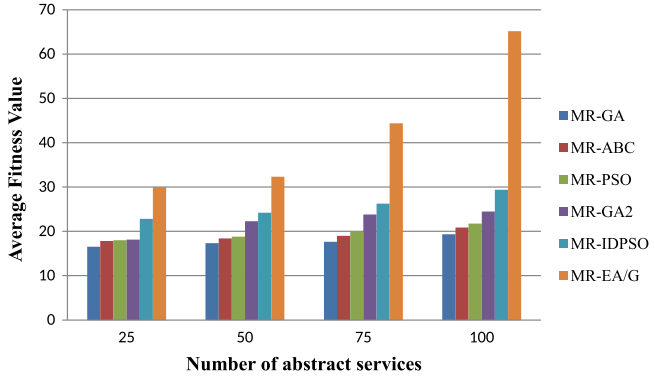

Fig. 9. Average fitness values for 10 000 candidate services.

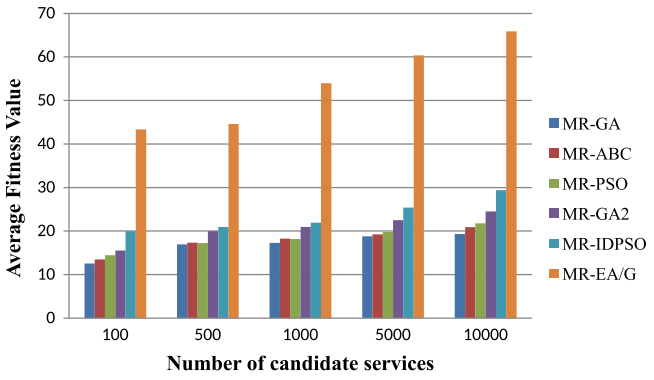

Fig. 10. Average fitness values for 100 abstract services.

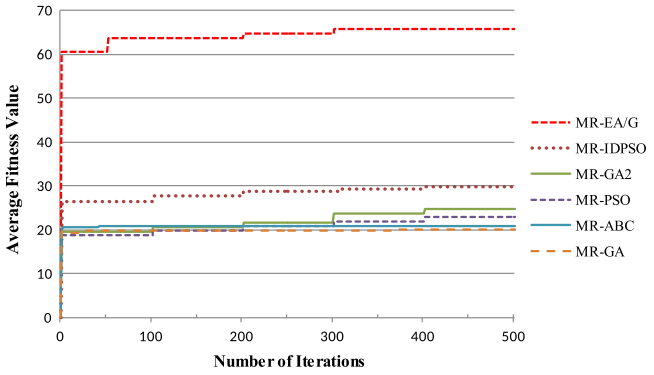

Fig. 11. Average fitness values for 100 abstract services (10 000 candidate services) by increasing the number of iterations.

in each generation and updates the probability values for each generation based on the global statistical information of the population.

Fig. 11 depicts the evaluation of fitness values by increasing the number of iterations for 100 abstract services, where each abstract service has 10 000 candidate services. In Fig. 11, we observe that the average fitness value is slightly increasing with number of iterations for all approaches. However, our proposed approach converges very quickly and gives the satisfactory results in lesser time than the other approaches.

Table 5 represents the execution time of different approaches for 100 abstract services. The execution time to obtain the best solutions by MR-GA, MR-ABC, MR-PSO, MR-GA2, and MR-IDPSO are 62.58, 115.59, 61.08, 57.86, and 61.95 s respectively, while the execution time for MR-EA/G is 30.23 s. The time complexity of MR-EA/G, MR-GA, MR-PSO, MR-IDPSO, MR-GA2, and MR-ABC are $O(n)$, $O(n^2)$, $O(n^2)$, $O(n.logn)$, $O(n.logn)$ and $O(n^2)$ respectively.

In MR-GA, for each iteration, the new population is generated, and the individual fitnesses are evaluated (based on fixed and predetermined crossover and mutation). The whole process consumes more than twice the time compared to our proposed approach. Similarly, in MR-PSO and MR-IDPSO, for each iteration, the new population is generated, and the particle fitness is evaluated (based on velocity and position of each particle) which gets changed in each iteration. The change decreases the convergence rate and premature rate, which in turn, consumes twice the time compared to our proposed approach. Likewise, in MR-GA2, the antibody repertoire is randomly generated first, and then low-affinity antibodies have replaced by new random antibodies during the mutation process in each iteration which consumes more time. In MR-ABC, the initial population is generated randomly. Further, for each iteration the population is subjected to repeat the cycles of the search processes of the employed, onlooker, and scout bees, respectively. The whole process consumes more than twice the time compared to our proposed approach. In our proposed approach, the MR-skyline operator is developed for abandoning some candidate services with a non-optimal solution and thereby, gradually reducing the valid search space. We used the guided mutation method, in which, an offspring is produced by combining both statistical information about search space and local information about parent solution. Hence, our proposed approach reduces the search space and consumes less time for execution.

We compared our proposed approach with each of other approaches by performing the statistical tests (parametric and

Table 5
Execution time (s) for 100 abstract services.

Approaches/candidate services	100	500	1000	5000	10 000
MR-GA	58.01	63.22	57.32	61.27	62.58
MR-ABC	180.01	120.2	120.54	80.86	115.59
MR-PSO	52.33	48.76	52.19	40.89	61.08
MR-GA2	56.21	56.52	55.12	58.36	57.86
MR-IDPSO	54.65	51.08	55.38	59.36	61.95
MR-EA/G	**29.69**	**30.18**	**33.09**	**30.19**	**30.23**

Table 6
T-test statistical analysis results for abstract services.

Abstract services	Approaches		MR-GA	MR-IDPSO	MR-PSO	MR-ABC	MR-GA2	MR-EA/G
25	MR-GA	*T*-value/*P*-value	*	*	*	*	*	*
	MR-IDPSO		249.219 (0)	*	*	*	*	*
	MR-PSO		21.3605 (0)	83.8613 (0)	*	*	*	*
	MR-ABC		30.4911 (0)	114.6498(0)	0.5161(0.6076)	*	*	*
	MR-GA2		68.7899 (0)	129.1001 (0)	9.6099 (0)	13.7649 (0)	*	*
	MR-EA/G		422.5523(0)	188.5225 (0)	197.536 (0)	257.54 (0)	297.6201 (0)	*
50	MR-GA	*T*-value/*P*-value	*	*	*	*	*	*
	MR-IDPSO		171.9023 (0)	*	*	*	*	*
	MR-PSO		28.7490 (0)	107.4319 (0)	*	*	*	*
	MR-ABC		32.1787 (0)	123.105 (0)	2.2617 (0.2747)	*	*	*
	MR-GA2		249.3636(0)	55.5187 (0)	94.9382 (0)	126.0214 (0)	*	*
	MR-EA/G		560.1384 (0)	184.9778 (0)	310.2354 (0)	376.5437 (0)	467.4155 (0)	*
75	MR-GA	*T*-value/*P*-value	*	*	*	*	*	*
	MR-IDPSO		247.276 (0)	*	*	*	*	*
	MR-PSO		41.6562 (0)	119.4937 (0)	*	*	*	*
	MR-ABC		38.88 (0)	146.9068 (0)	7.8567 (0)	*	*	*
	MR-GA2		132.9658(0)	56.0843 (0)	61.4957 (0)	77.1474 (0)	*	*
	MR-EA/G		373.925 (0)	232.5213 (0)	293.0033 (0)	316.9459 (0)	256.1271 (0)	*
100	MR-GA	*T*-value /*P*-value	*	*	*	*	*	*
	MR-IDPSO		348.7109 (0)	*	*	*	*	*
	MR-PSO		57.5519 (0)	187.2576 (0)	*	*	*	*
	MR-ABC		28.2106 (0)	180.2931 (0)	16.6644 (0)	*	*	*
	MR-GA2		226.3003(0)	177.5601 (0)	74.6589 (0)	82.7483 (0)	*	*
	MR-EA/G		319.683 (0)	246.8641 (0)	297.448 (0)	299.4491 (0)	284.5778 (0)	*

non-parametric). The results of paired two-tailed *T*-test [27] and Wilcoxon signed rank test [28,29] justify whether the obtained best mean values of all algorithms have a distinguished difference with 58 degrees of freedom at 1% level of significance. *T*-test and Wilcoxon signed rank test results are presented in Tables 6 and 7. Based on Table 6, the obtained values are statistically significant (all T-values are positive and P-values are 0.0000). Based on Table 7, the obtained values are statistically significant (all Z-values are obtained by positive ranks, and *P* values are 0.000). Therefore, our proposed method is statistically more significant than other methods.

5. Related work

QoS-aware big service composition is an emerging research topic in service-oriented computing [2]. Due to the seamless proliferation of services, it is difficult to select an optimal service from available services. To reduce search space and select an optimal service, a skyline operator is used that prunes the number of candidate services from the set of services based on their QoS attributes [30]. There are several researches proposed using skyline operators [31–34] to reduce search space and accelerate the service composition to attain high QoS. Limin et al. [35] proposed a MapReduce-based service selection method to improve the efficiency and reduce search space. Akrivi et al. [36] developed a novel method to improve the skyline operator efficiency using MapReduce angle-based space partitioning method. Jian et al. [37] proposed a MapReduce Skyline operator to prune the candidate services and find the optimal solution for QoS-aware service composition. Kasper et al. [38] proposed an efficient skyline approach to prune the search space using MapReduce grid partitioning approach. However, all of these mentioned approaches consider either a single QoS attribute or multiple QoS attributes with one QoS attribute is a prime attribute. Thus, it is not possible to model the scalability of service composition. Another disadvantage is that the services involved in composition proliferate with the number of tasks (or abstract services) and the associated services (or candidate services). Hence, these approaches cannot be solved in a polynomial time. In order to solve this problem, we presented a novel MR-skyline operator with multiple QoS attributes having two prime attributes (eg. response time and price) to reduce search space as well as increase convergence rate and premature rate.

A Genetic algorithm (GA) is effective to address the obstacle in the optimization process of service composition. Canfora et al. [3] investigated a genetic algorithm based method to solve QoS-ware web service composition. Yilmaz et al. [6] introduced an improved GA to solve service composition with minimum global QoS and improve scalability. Quanwang et al. [7] proposed a novel method to optimize the overall QoS values using a backtracking-based algorithm and an extended genetic algorithm for QoS-aware web service composition. Ludwig [5] proposed a clonal selection based algorithm to solve workflow service selection with high solution quality.

All of these methods are based on single point crossover and mutation. These methods proposed modifications to crossover operator and selection of parent chromosome to escape from local optima. However, if the number of abstract services and candidate services grows for each abstract service, chromosomes become very long. Hence, the GA method results in poor readability of the chromosome, crossover, mutation and fails to predict the information related to the semantics of services. Further, it results in low convergence rate and low premature rate in local optima [39].

Table 7
Wilcoxon signed rank test results for abstract services.

Abstract services	Approaches		MR-GA	MR-IDPSO	MR-PSO	MR-ABC	MR-GA2	MR-EA/G
25	MR-GA	*Z*-value /*P*-value	*	*	*	*	*	*
	MR-IDPSO		−4.7821 (0)	*	*	*	*	*
	MR-PSO		−4.7821 (0)	−4.7821 (0)	*	*	*	*
	MR-ABC		−4.7821 (0)	−4.7821 (0)	−0.7919 (0.4295)	*	*	*
	MR-GA2		−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	*	*
	MR-EA/G		−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	*
50	MR-GA	*Z*-value /*P*-value	*	*	*	*	*	*
	MR-IDPSO		−4.7821 (0)	*	*	*	*	*
	MR-PSO		−4.7821 (0)	−4.7821 (0)	*	*	*	*
	MR-ABC		−4.7821 (0)	−4.7821 (0)	−0.977 (0.327)	*	*	*
	MR-GA2		−4.7821 (0)	−4.7821 (0)	−0.4628 (0.64552)	−4.7821 (0)	*	*
	MR-EA/G		−4.7821(0)	−4.7821 (0)	−4.7821 0)	−4.7821 (0)	−4.7821 (0)	*
75	MR-GA	*Z*-value/*P*-value	*	*	*	*	*	*
	MR-IDPSO		−4.7821 (0)	*	*	*	*	*
	MR-PSO		−4.7821 (0)	−4.7821 (0)	*	*	*	*
	MR-ABC		−2.9104 (0.0036)	−4.7821 (0)	−1.43 (0.1527)	*	*	*
	MR-GA2		−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	−1.6352 (0.101)	*	−4.7821 (0)
	MR-EA/G		−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	*
100	MR-GA	*Z*-value/*P*-value	*	*	*	*	*	*
	MR-IDPSO		−4.7821 (0)	*	*	*	*	*
	MR-PSO		−4.7821 (0)	−4.7821 (0)	*	*	*	*
	MR-ABC		−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	*	*	*
	MR-GA2		−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	−4.7821 (0)	*	−4.7821 (0)
	MR-EA/G		−4.7821(0)	−4.7821 (0)	−4.7821 0)	−4.7821 (0)	−4.7821 (0)	*

Generally, genetic algorithms use crossover and mutation operators to produce offsprings (solutions) from the selected parents without considering search space global information using the location information of the solutions found so far. Unlike genetic algorithms, EA/G uses a probability method to generate offsprings, that ensures the distributing of promising solutions in each generation from which off-springs are generated by sampling. Instead of directly using the position values, EA/G updates the probability values for each generation based on the global statistical information extracted from the members of the population. To handle the complementary aspect of GAs and EDAs (estimation of distribution algorithms), Zhang et al. [18] proposed a novel evolutionary algorithm with guided mutation (EA/G) that employs the local information of the solutions (like GAs) and global information about the search space (EDAs) while generates an offsprings (solutions). EA/G uses a mutation operator, named as guided mutation, where an offspring is produced by combining both global statistical information about search space and local information about parent solution.

We proposed a novel MapReduce-based EA/G algorithm for QoS-aware Big service composition. To the best of our knowledge, this is the first paper based on MR-EA/G for QoS-aware Big service composition using MR-Skyline operator. The empirical analysis of our proposed method shows the best performance in terms of feasibility, scalability, and optimality with different QoS attributes for solving Big service composition.

6. Conclusion and future work

With the proliferation of services, it is difficult to select and compose the optimal services to fulfill the user requirements. In this paper, we proposed a novel MR based evolutionary algorithm with guided mutation (MR-EA/G) for addressing QoS-aware Big service composition. The QoS based service pre-selection process is accomplished by using a MR-based skyline operator. To improve the searching efficiency of service selection and composition, we presented a novel EA/G with MapReduce framework. Based on the experiments, we infer that our proposed approach outperforms against all other approaches.

In future, we plan to improve our proposed approach, making it more effective and efficient by using other machine learning and meta-heuristic algorithms. In addition, other aspects such as reputation-based service selection [40] could be integrated. The efficient scheduling tasks on cloud systems is a complex problem which has attracted many research efforts [41]. Models for scheduling become more sophisticated, incorporating such notions as energy-efficiency [42,43], network-awareness [44], and heterogeneity of systems [45]. Pricing models can be expected to evolve in the same direction, including explicitly these concepts [46]. Such a scenario of multifaceted pricing models offers plenty of opportunities to extend our approach.

References

[1] X. Xu, Q.Z. Sheng, L.-J. Zhang, Y. Fan, S. Dustdar, From big data to big service, Computer 48 (7) (2015) 80–83.
[2] J. Chandrashekar, G.R. Gangadharan, R. Buyya, Computational intelligence based qos-aware web service composition: a systematic literature review, IEEE Trans. Serv. Comput. (2016). http://dx.doi.org/10.1109/TSC.2015.2473840.
[3] G. Canfora, M. Di Penta, R. Esposito, M.L. Villani, An approach for QoS-aware service composition based on genetic algorithms, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO, 2005, pp. 1069–1075.
[4] M. Tang, L. Ai, A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2010, pp. 1–8.
[5] S.A. Ludwig, Clonal selection based genetic algorithm for workflow service selection, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–7.
[6] A.E. Yilmaz, P. Karagoz, Improved genetic algorithm based approach for qos aware web service composition, in: Proceedings of the IEEE International Conference on Web Services, 2014, pp. 463–470.
[7] Q. Wu, F. Ishikawa, Q. Zhu, D.H. Shin, QoS-aware multigranularity service composition: Modeling and optimization, IEEE Trans. Syst. Man Cybern. (2016) 1–13. http://dx.doi.org/10.1109/TSMC.2015.2503384.
[8] X.-Q. Fan, X.-W. Fang, C.-J. Jiang, Research on web service selection based on cooperative evolution, Expert Syst. Appl. 38 (8) (2011) 9736–9743.
[9] T.G. Crainic, M. Toulouse, Parallel meta-heuristics, in: Handbook of Meta-heuristics, Springer, 2010, pp. 497–541.
[10] J.A. Parejo, S. Segura, P. Fernandez, A. Ruiz-Cortes, QoS-aware web services composition using GRASP with Path Relinking, Expert Syst. Appl. 41 (9) (2014) 4211–4223.
[11] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, QoS-aware middleware for Web services composition, IEEE Trans. Serv. Comput. 30 (5) (2004) 311–327.
[12] W. Dou, X. Zhang, J. Liu, J. Chen, HireSome-II: Towards privacy-aware cross-cloud service composition for big data applications, IEEE Trans. Parallel Distrib. Syst. 26 (2) (2015) 455–466.

[13] J. Chandrashekar, G.R. Gangadharan, QoS-aware web service composition using quantum inspired particle swarm optimization, in: Proceedings of the 7th International KES Conference on Intelligent Decision Technologies, Springer, 2015, pp. 255–265.
[14] T. Saaty, Fundamentals of Decision Making and Priority Theory with Analytical Hierarchical Process, vol. 6, RWS Publications, University of Pittsburgh, Pittsburgh, USA, 1980.
[15] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, Y. Fan, An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition, Appl. Soft Comput. 12 (8) (2012) 2208–2216.
[16] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[17] S.J. Kang, S.Y. Lee, K.M. Lee, Performance comparison of OpenMP, MPI, and mapreduce in practical problems, Adv. Multimedia (2015). http://dx.doi.org/10.1155/2015/575687.
[18] Q. Zhang, J. Sun, E. Tsang, An evolutionary algorithm with guided mutation for the maximum clique problem, IEEE Trans. Evol. Comput. 9 (2) (2005) 192–200.
[19] F. Zhang, K. Hwang, S.U. Khan, Q.M. Malluhi, Skyline discovery and composition of multi-cloud mashup services, IEEE Trans. Serv. Comput. 9 (1) (2016) 72–83.
[20] A. Mostafa, M. Zhang, Multi-Objective service composition in uncertain environments, IEEE Trans. Serv. Comput. (2015). http://dx.doi.org/10.1109/TSC.2015.2443785.
[21] D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Trans. Serv. Comput. 33 (6) (2007) 369–384.
[22] J.A. Villasenor Alva, E.G. Estrada, A generalization of Shapiro–Wilk's test for multivariate normality, Comm. Statist. Theory Methods 38 (11) (2009) 1870–1883.
[23] N.E.A. Khalid, A.F.A. Fadzil, M. Manaf, Adapting mapreduce framework for genetic algorithm with large population, in: Proceedings of the IEEE Conference on Systems, Process & Control, IEEE, 2013, pp. 36–41.
[24] X. Wang, Z. Wang, X. Xu, An improved artificial bee colony approach to qos-aware service selection, in: Proceedings of the 20th International Conference on Web Services, IEEE, 2013, pp. 395–402.
[25] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intelligence 1 (1) (2007) 33–57.
[26] Y. Zhang, Z. Jing, Y. Zhang, Mr-idpso: a novel algorithm for large-scale dynamic service composition, Tsinghua Sci. Technol. 20 (6) (2015) 602–612.
[27] D.W. Zimmerman, Teacher's corner: A note on interpretation of the paired-samples t test, J. Educ. Behav Stat. 22 (3) (1997) 349–360.
[28] S. Siegel, Nonparametric Statistics for the Behavioral Sciences, McGraw-Hill, 1956.
[29] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bull. 1 (6) (1945) 80–83.
[30] S. Borzsony, D. Kossmann, K. Stocker, The skyline operator, in: Proceedings of the 17th International Conference on Data Engineering, IEEE, 2001, pp. 421–430.
[31] M. Alrifai, D. Skoutas, T. Risse, Selecting skyline services for QoS-based web service composition, in: Proceedings of the 19th International Conference on World Wide Web, ACM, 2010, pp. 11–20.
[32] H. Han, H. Jung, S. Kim, H.Y. Yeom, A skyline approach to the matchmaking web service, in: Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE, 2009, pp. 436–443.
[33] Q. Yu, A. Bouguettaya, Computing service skyline from uncertain QoWS, IEEE Trans. Serv. Comput. 3 (1) (2010) 16–29.
[34] D. Kossmann, F. Ramsak, S. Rost, Shooting stars in the sky: An online algorithm for skyline queries, in: Proceedings of the 28th International Conference on Very Large Data Bases, 2002, pp. 275–286.
[35] L. Pan, L. Chen, J. Wu, Skyline web service selection with mapreduce, in: Proceedings of the International Conference on Computer Science and Service System, IEEE, 2011, pp. 739–743.
[36] A. Vlachou, C. Doulkeridis, Y. Kotidis, Angle-based space partitioning for efficient parallel skyline computation, in: Proceedings of the SIGMOD International Conference on Management of Data, ACM, 2008, pp. 227–238.
[37] J. Wu, L. Chen, Q. Yu, L. Kuang, Y. Wang, Z. Wu, Selecting skyline services for QoS-aware composition by upgrading MapReduce paradigm, Clust. Comput. 16 (4) (2013) 693–706.
[38] K. Mullesgaard, J.L. Pederseny, H. Lu, Y. Zhou, Efficient skyline computation in mapreduce, in: Proceedings of the 17th International Conference on Extending Database Technology, 2014, pp. 37–48.
[39] Y. Ma, C. Zhang, Quick convergence of genetic algorithm for QoS-driven web service selection, Comput. Netw. 52 (5) (2008) 1093–1104.
[40] V.N. Serbanescu, F. Pop, V. Cristea, O.-M. Achim, Web services allocation guided by reputation in distributed SOA-based environments, in: Proceedings of the 11th International Symposium on Parallel and Distributed Computing, IEEE, 2012, pp. 127–134.
[41] G.V. Iordache, M.S. Boboila, F. Pop, C. Stratan, V. Cristea, A decentralized strategy for genetic scheduling in heterogeneous environments, in: OTM Confederated International Conferences on the Move To Meaningful Internet Systems, Springer, 2006, pp. 1234–1251.
[42] A. James, N. Yaacob, Special issue: Quality of service in grid and cloud 2015, Future Gener. Comput. Syst. 50 (c) (2015) 1–2.
[43] A. Sfrent, F. Pop, Asymptotic scheduling for many task computing in big data platforms, Inform. Sci. 319 (2015) 71–91.
[44] U. Fiore, F. Palmieri, A. Castiglione, A. De Santis, A cluster-based data-centric model for network-aware task scheduling in distributed systems, Int. J. Parallel Program. 42 (5) (2014) 755–775.
[45] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, J. Kołodziej, Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing, Future Gener. Comput. Syst. 51 (2015) 61–71.
[46] C. Chilipirea, A.-C. Petre, C. Dobre, F. Pop, Enabling mobile cloud wide spread through an evolutionary market-based approach, IEEE Syst. J. 10 (2) (2016) 839–846.

Chandrashekar Jatoth received his B.E in Information Technology from Osmania University and M.Tech. in Artificial Intelligence from University of Hyderabad, Hyderabad, India in 2008 and 2010 respectively. Currently, he is working towards the Ph.D. degree in University of Hyderabad and Institute for Development and Research in Banking Technology (IDRBT), Hyderabad, India. His research interests focus on QoS, web service composition, and computational intelligence techniques.

G.R. Gangadharan is an Associate professor at the Institute for Development and Research in Banking Technology, Hyderabad, India. His research interests focus on the interface between technological and business perspectives. Gangadharan received his Ph.D. in information and communication technology from the University of Trento, Italy, and the European University Association. He is a senior member of IEEE and ACM. Contact him at geeyaar@gmail.com.

Ugo Fiore, Ph.D. is with the Department of Molecular Medicine and Medical Biotechnologies at Federico II University, Italy. He is also an adjunct professor at Parthenope University of Naples. Earlier, he worked for a decade in the telco industry. His research interests include nonlinear analysis, deep learning, classification of Internet traffic, optimization, energy-saving and energy-aware systems, covert communications, and security. He has authored or co-authored about 60 papers on international journals and conferences. He is serving as Associate Editor with two journals and is a member of the editorial board in two other journals. He has participated to the organizing committees of numerous conferences.

Rajkumar Buyya is a Fellow of IEEE, Professor of Computer Science and Software Engineering, Future Fellow of the Australian Research Council, and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the University, commercializing its innovations in Grid and Cloud Computing. Dr. Buyya has authored/co-authored over 450 publications. He is one of the highly cited authors in computer science and software engineering worldwide. Microsoft Academic Search Index ranked Dr. Buyya as one of the Top 5 Authors during the last 10 years (2001–2012) and #1 in the world during the last 5 years (2007–2012) in the area of Distributed and Parallel Computing. For further information on Dr. Buyya, please visit: http://www.buyya.com.