Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Applying gravitational search algorithm in the QoS-based Web service selection problem#

Bahareh ZIBANEZHAD[†1], Kamran ZAMANIFAR[†‡1], Razieh Sadat SADJADY[1], Yousef RASTEGARI[2]

([1]*Computer Engineering Department, Islamic Azad University, Najafabad Branch, Isfahan, Iran*)

([2]*Electrical and Computer Engineering Department, Shahid Behehsti University, Tehran, Iran*)

[†]E-mail: b.zibanezhad@gmail.com; zamanifar@eng.ui.ac.ir

Received Sept. 2, 2010; Revision accepted Apr. 2, 2011; Crosschecked July 29, 2011

Abstract: With the growing use of service-oriented architecture for designing next generation software systems, the service composition problem and its execution complexity have become even more important in responding to different user requests. The gravitational search algorithm is one of the latest heuristic algorithms. It has a number of distinguishing features, such as rapid convergence, lower memory usage, and the use of particular parameters, for instance, the distance between the solutions. In this paper, we propose a model for the optimization of the Web service composition problem based on qualitative measures and the gravitational search algorithm. To determine the efficacy of this proposed model we solve the problem with the particle swarm optimization algorithm for comparison. Simulation results show that the gravitational search algorithm has a high potential and substantial efficiency in finding the best combination of Web services.

Key words: Web service composition, Gravitational search algorithm (GSA), Quality of service (QoS), Ontology engineering
doi:10.1631/jzus.C1000305 **Document code:** A **CLC number:** TP311

1 Introduction

With the expansion of Internet services, the demand for business-to-business communication and cooperation has increased. Consequently, new forms of technology such as Web services have come into being. The main focus related to Web services is the ability to establish harmony between distributed, non-centralized, and heterogeneous usage. Considering the rapid increase of Web users and the growing complexity of their demands, simple atomic services are inadequate; the combination of services to render possible complex services is a necessity (Staab *et al*., 2003; Maximilien and Singh, 2004; Leutenmayr, 2007). However, due to a large number of providers, various services are offered which are, in terms of the functions they carry out, the same, and they can be substituted for one another. These services are, of course, different from one another in non-functional properties such as response time, availability, throughput, security, reliability, and execution cost (Benveniste, 2008; Chen and Wang, 2009), and are therefore different in terms of efficiency. Since the services should be chosen in such a way as to enable the best possible quality for the overall combination, the issue of Web service composition leads to the quality-of-service (QoS) engineering problem. It can be said that qualitative measures reflect user needs and satisfaction. Therefore, the response time, for instance, may not be very important for a user, but the service cost may be so, or vice versa (Ismail *et al*., 2009); or a user may want to determine a specific price for the cost measure. Since solving this type of problem through the ordinary method falls within the NP-hard problem category and is very time consuming and costly, it can be modeled as an optimization problem and answered using the heuristic search

‡ Corresponding author

A preliminary version was presented at the International Conference on Innovations in Information Technology, Dec. 15–17, 2009, Al-Ain, United Arab Emirates

algorithms (Chen and Wang, 2007; Liu *et al*., 2009). This study deals with the gravitational search algorithm (GSA) (Rashedi *et al*., 2009), a type of heuristic algorithm, as the answer to finding the best Web services composition on the premise of user priorities and constraints.

2 Combination of Web services based on qualitative measures

The Web service combination problem based on qualitative measures can be explained by Fig. 1.

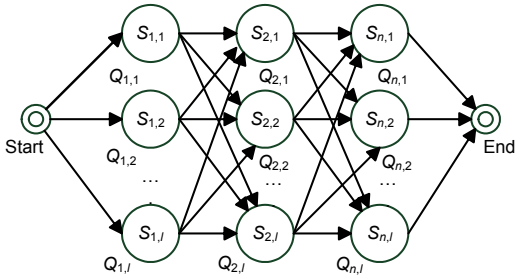

Fig. 1 Web service combination

Different services with different purposes have been published in Universal Description, Discovery and Integration (UDDIs) by service providers. These services cannot respond to different aspects of user requests; this is why we should think about the service composition problem. The main challenges of service composition consist of the following steps:

1. Converting requests to a machine understandable model;

2. Discovering suitable candidate services for each task of the input model;

3. Selecting the best path between candidate services based on required user QoS criteria;

4. Converting the solution (made in step 3) to an executable language such as business process execution language (BPEL).

In the above process, the system receives the importance of each QoS parameter from the user in step 1. The system may find different models which are responsible to user requests in step 3, but it will continue with the best one based on QoS (obtained in step 1) to step 4. Finally, using a BPEL engine in step 4, the system executes the model and responds to the request. Most Web services are described in Web services description language (WSDL), which is syntactic and extensible markup language (XML) based. In recent years, OWL-S, with a new perspective on semantic issues, has been released. Fig. 2 shows different execution operators that are supported in BPEL language, and composition algorithms should support them as well.

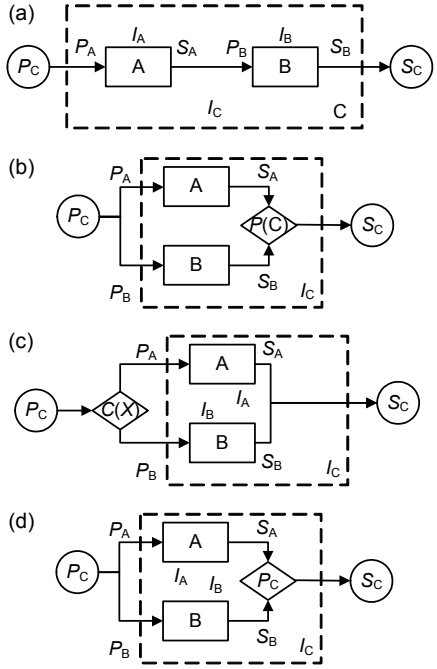

Fig. 2 Some operators supported in BPEL
(a) Sequence; (b) Parallel; (c) Selection; (d) Loop

If we consider the composite Web service as a combination of n atomic services defined as the vector $\boldsymbol{S}=(s_1, s_2, \ldots, s_n)$, according to Fig. 1, for every atomic service s_i there is a group of candidate Web services alike in terms of function, but different in terms of qualitative parameters. The aim now is to find the best combination of Web services with regard to qualitative parameters. By combination we mean a path which would guide us from the start point to the end point (Fig. 1). Note that, for any task, only one of the candidate services is used.

Much research has been done on the choice of atomic services, some of which is related to service-combination in the design time and some to the atomic combination of services during execution (Li *et al*., 2009). Various algorithms have been used for both purposes. For example, Menasce (2004) studied issues such as the way (between time intervals or by tool) in which qualitative measures should be estimated, who (service providers or independent and

intelligent agents) should estimate the measures, and where (on network edges or the user side) these measures should be estimated in the access point (the location of access) of Web services. Zeng *et al*. (2003) made the presentation of qualitative measures and solution finding by linear programming for choosing the best services combination. Maximilien and Singh (2004) and Lecue and Mehandjiev (2009) enhanced the dynamic choice of services by presenting a QoS ontology. Liu *et al*. (2007) achieved the combination of Web services using the spanning tree algorithm, which enables an optimum choice in path-finding between Web services and consequently the reduction of response time.

In Claro *et al*. (2005), Yu and Lin (2005), and Ai *et al*. (2008), local and global strategies have been used for the best combination choice. This algorithm (Ai *et al*., 2008) first eliminates combinations with a low QoS (local strategy), and then from the remaining combinations, chooses the best one in terms of qualitative measures using the knapsack problem 0-1 algorithm (global strategy). Chen and Wang (2007) used the particle swarm optimization (PSO) algorithm to find best service combination. PSO, as an optimization algorithm, has certain advantages such as simple implementation and rapid convergence.

Semantic Web concepts have been applied in the service composition problem in recent years (Lecue, 2009; Talantikite *et al*., 2009). Talantikite *et al*. (2009) presented a proper semantic framework for QoS-based selection and discovery. Applying Web service semantic annotation, constructing a semantic network of UDDI Web services, and using a customized backward changing algorithm are some of the major innovations presented in Talantikite *et al*. (2009). Annotation process was achieved by using an ontology related to each Web service.

3 Algorithm presentation

3.1 Gravitational search algorithm

GSA is a heuristic type algorithm derived from the concepts of 'mass' and 'gravitational force' and the simulation of Newton's laws. The system space is a multi-dimensional coordinate system in the space defined by the problem: each point of the space is an answer to the problem.

If the system is considered as a collection of num masses, the position of each mass is a space point and a probable answer to the problem. The position of dimension d from mass i is shown by X_i^d:

$$\boldsymbol{X}_i = (X_i^1, ..., X_i^d, ..., X_i^n). \tag{1}$$

In this system, at time t a force $F_{ij}^d(t)$ is exerted on each mass i from mass j in the direction of dimension d. The amount of this force is calculated according to Eq. (2). Mg_j is the gravitational mass of mass j, $G(t)$ is the gravitational constant at time t, ε is a very small number, and R_{ij} is the distance between masses i and j. The Euclidian distance has been used.

$$F_{ij}^d = \frac{G(t)\cdot \mathrm{Mg}_j(t)}{R_{ij}(t)+\varepsilon}(X_j^d(t) - X_i^d(t)), \tag{2}$$

$$R_{ij}(t) = \left\| \boldsymbol{X}_i(t), \boldsymbol{X}_j(t) \right\|_2. \tag{3}$$

The force exerted on mass i in the direction of dimension d at time t is, according to Eq. (4), equal to the sum of all forces exerted on the mass from the other masses of the system.

$$F_i^d(t) = \sum_{j=1, j\neq i}^{\mathrm{num}} r_j F_{ij}^d(t), \tag{4}$$

where r_j is a uniform random variable in the interval [0, 1], used to ensure a random search.

According to Newton's laws of gravitation and motion, the acceleration of any mass in the direction of dimension d is equal to the force exerted on the mass in that direction divided by the inertia mass:

$$a_i^d(t) = \frac{F_i^d(t)}{\mathrm{Mi}_i(t)}, \tag{5}$$

where $a_i^d(t)$ is the acceleration of mass i in the direction of dimension d at time t and Mi_i is the inertia mass of mass i.

The velocity of any mass is equal to the sum of an index of the velocity of the mass and its acceleration, as defined in Eq. (6). The new position of dimension d in relation to mass i is calculated using Eq. (7).

$$V_i^d(t+1) = r_i \cdot V_i^d(t) + a_i^d(t), \tag{6}$$

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1), \tag{7}$$

where r_i is a uniform random variable in the interval [0, 1], used to ensure a random search.

To determine the gravitational index, Eq. (8) is used:

$$G(t)=\beta^{-\alpha t/T}, \tag{8}$$

where the gravitational constant decreases exponentially. α is equal to 20 and β shows linear increase ranging from one to three.

To determine the masses, according to Eqs. (9) and (10), the mass target function is used such that the masses with a higher degree of fitness are accorded a higher quality of mass.

$$\mathrm{Mg}_i=\frac{\mathrm{fit}_i(t)-\mathrm{worst}(t)}{\mathrm{best}(t)-\mathrm{worst}(t)}, \tag{9}$$

$$\mathrm{Mi}_i=\frac{\mathrm{Mg}_i}{\sum_{j=1}^{\mathrm{num}}\mathrm{Mg}_j} \tag{10}$$

where $\mathrm{fit}_i(t)$ represents the degree of fitness in mass i at time t and num is the number of population. In problems involving maximization, Eqs. (11) and (12) can be used to attain the best and the worst variables:

$$\mathrm{Best}(t)=\max\left(\mathrm{fit}_j(t)\right), \tag{11}$$

$$\mathrm{Worst}(t)=\min\left(\mathrm{fit}_j(t)\right). \tag{12}$$

Therefore, the GSA algorithm stages are:

1. Determining the system environment;
2. Initial quantity allocation;
3. Mass evaluation;
4. Updating the parameters G, best, worst, Mi, and Mg;
5. Calculating the amount of force exerted on each mass;
6. Calculating the acceleration and velocity of each mass;
7. Updating the position of masses;
8. Ordering: if the condition of termination is not met, go to step 2;
9. End.

3.2 Particle swarm optimization algorithm

The PSO algorithm is a probability technique of optimization that works on the basis of population (Maximilien and Singh, 2004), initially derived from the group behavior of fish or birds in search of food. In the PSO algorithm, each solution, which is termed a 'particle', is equivalent to a bird in a flock. Each particle has a certain degree of fitness which is determined by a fitness function. The closer is a particle to the target (e.g., food in the bird-flock model) in the search space, the fitter is it deemed. Also, each particle has a velocity which is responsible for its guidance. Each particle continues its movement in the problem space pursuing the particles which are, in any present condition, the best.

In the beginning, a number of particles (i.e., solutions) are randomly created, and the aim is to find the best solution by updating the generations. In each step, each particle is updated with regard to two positions: one is the best position it has gained thus far and is known and kept as pbest, which is the particle's best local position, and the other is the best position attained by the swarm of particles thus far, which is the best global position of the particles, shown as gbest. After finding the best quantities, the velocity and location of each particle are updated using

$$\begin{aligned}V_i^d(t+1)=&w\cdot V_i^d(t)+C_1\cdot r_1\cdot(X^d_{\mathrm{pbest}i}(t)-X_i^d(t))\\&+C_2\cdot r_2\cdot(X^d_{\mathrm{gbest}}(t)-X_i^d(t)),\end{aligned} \tag{13}$$

$$X_i^d(t+1)=X_i^d(t)+V_i^d(t+1), \tag{14}$$

where V_i^d is the particle velocity and X_i^d is the present location of the particle, each being an array with a length equal to the problem dimensionality, r_1 and r_2 are two random numbers in [0, 1], and C_1 and C_2 are learning factors. Usually C_1 and C_2 are supposed equal and equal to 2 (i.e., C_1=C_2=2). The right-hand side of Eq. (13) is composed of three parts: the first is the present velocity of the particle and the second and third parts relate to the change in velocity and the change towards the best individual and collective experience, respectively. Actually, the aim in the combination of these two factors is to achieve a balance between local and global search. Accordingly, the PSO algorithm stages are as follows:

1. Determining the system environment;
2. Initial quantity allocation;
3. Particle evaluation;
4. Particle velocity calculation;
5. Updating particle position;
6. Ordering: if the condition of termination is not met, go to step 2;
7. End.

4 Finding the best QoS awareness combination from Web services

In this part we first design a fitness function and present the concept of optimization and then propose our models.

4.1 Designing the fitness function

A composite service is constituted by a number of atomic services each with a specific duty. Considering the expansion of the Internet and the variety of providers for any task t_i, there are a number of candidate services similar in function, but different in terms of qualitative measures. In the following formula, S_i shows a set of l services for task t_i:

$$S_i = \{S_{i1}, S_{i2}, ..., S_{il}\}. \quad (15)$$

The integration of Web service quality vectors yields matrix $\boldsymbol{Q}$:

$$\boldsymbol{Q} = (Q_{ij}^k),\ 1 \le i \le n,\ 1 \le j \le l,\ 1 \le k \le m, \quad (16)$$

where n is the number of tasks, l is the number of candidate services ready to carry out any task, and m is the number of qualitative measures for each service.

In matrix $\boldsymbol{Q}$ each line corresponds to a set of candidate Web services and each $\boldsymbol{Q}_{ij}$ element with the qualitative measures of service S_{ij}; k shows the kth QoS measure.

Some qualitative measures such as execution time and cost are in inverse relationship with the quality parameter (i.e., a higher level shows a lower quality), whereas some measures such as reliability and availability are in direct relationship with it (i.e., a higher level in these measures shows a higher degree of quality). Since we need a target function composed of the above measures, we use Eq. (17) for the measures with inverse relationship and Eq. (18) for those with direct relationship as the functions for the interval [0, 1].

$$V(Q_{ij}^k) = \begin{cases} \dfrac{Q_{ij}^k - \min_v(Q_{iv}^k)}{\max_v(Q_{iv}^k) - \min_v(Q_{iv}^k)}, \\ \qquad \max_v(Q_{iv}^k) \ne \min_v(Q_{iv}^k), \\ 1, \\ \qquad \max_v(Q_{iv}^k) = \min_v(Q_{iv}^k), \end{cases} \quad (17)$$

$$V(Q_{ij}^k) = \begin{cases} \dfrac{\max_v(Q_{iv}^k) - Q_{ij}^k}{\max_v(Q_{iv}^k) - \min_v(Q_{iv}^k)}, \\ \qquad \max_v(Q_{iv}^k) \ne \min_v(Q_{iv}^k), \\ 1, \\ \qquad \max_v(Q_{iv}^k) = \min_v(Q_{iv}^k). \end{cases} \quad (18)$$

The quality of the resulting composite service is a determining factor for consumer satisfaction. Various users may prefer different qualities. For instance, one user may need minimum execution time and want specific limits for cost and validity, whereas for another user, cost may be more significant than execution time. Therefore, the aim is to determine the qualitative measures of composite services based on the user's constraints and priorities. For this purpose, a weight is allotted to each qualitative measure. The amount is determined by the circumstances of the user. As a result, with regard to Eqs. (17) and (18), the target function is derived from

$$F_{ij}^k = V_{ij}^k W_k,\quad 0 \le W_k \le 1,\quad \sum W_k = 1, \quad (19)$$

where W_k is the qualitative measure weight determined by the user, V_{ij}^k is the standardized form of impact of the kth QoS criterion of the jth candidate Web service for the ith task, and F_{ij}^k is the standardized form of impact of the kth QoS criterion of the jth candidate Web service for the ith task, by considering each criterion from the user's point of view.

The standardized quality of the jth candidate Web service of the ith task is calculated using

$$\text{article}_{ij} = \sum_k F_{ij}^k. \quad (20)$$

Finally, by the help of article, the fitness of each path is calculated. Our objective is finding a path with the highest fitness value. Fitness values are calculated in different cases based on the relation between Web services in the path (operators used in the path). Thus, Web services are divided into blocks based on their related operators, and then the fitness value of each block is calculated based on the composition operator used. As an example, if Sequence is the operator used among all Web services in the path, the fitness value is calculated as

$$\text{fitnesspath} = \sum \text{article}_{ij}, \tag{21}$$

when candidate service j for task i is anticipated in the path.

To compare the PSO and GSA algorithms, the designed evaluation function has been used for both.

4.2 Optimum version of the algorithm

The concept of optimization is the same in PSO and GSA algorithms and is defined as follows.

In the optimum version of GSA, only k fitter masses in the population have the capability of exerting force on the remainder of the masses. That is, in each repetition of the algorithm, the force exerted on any mass is equivalent to the overall sum of forces exerted on it from the k fitter masses in the population. Therefore, Eq. (4) is changed to

$$F_i^d(t) = \sum_{j \in \text{kbest}, j \neq i} r_1 \cdot F_{ij}^d(t), \tag{22}$$

where kbest represents the k fitter masses within the population.

In our model, the optimum masses are solutions that meet the global constraints set by users. For example, a user may decide on a 1500 USD limit for the overall cost of the chosen services. As mentioned before, for any task t_i, there is a set of candidate Web services $S_i = \{S_{i1}, S_{i2}, ..., S_{il}\}$ from which only one of the services can be chosen. In the following, X_{ij} shows the choice of service S_{ij} for the execution of task t_i:

$$\sum_{j=1}^{l} X_{ij} = 1, \quad X_{ij} \in \{0,1\}. \tag{23}$$

X_{ij} shows whether or not a Web service belongs to a composition. Suppose the number of Web services in S_i is l. The user constraint is exercised over the composite service according to

$$\sum_{i=1}^{n} \sum_{j=1}^{l} X_{ij} Q_{ij}^k \leq Q^k. \tag{24}$$

This constraint shows that the cost of the composite service should not extend the fixed amount Q_k. Therefore, in each execution of the algorithm only optimum masses (i.e., those that also answer global constraints) influence each other.

In the PSO algorithm, also, only the particles that answer users' global constraints are given participation in any stage.

4.3 Hypothetical model for Web service combination based on qualitative measures and GSA

Algorithm details are given as the following.

1. Determining the system environment.

To solve the problem of Web service combination with GSA, it is necessary to define the system environment beforehand. In our proposed model, each mass is a problem solution defined as

$$\boldsymbol{X}_i = (X_i^1, X_i^2, ..., X_i^j, ..., X_i^n), \tag{25}$$

where X_i^j (j=1, 2, …, n) is the service number used for task j in path i and n shows the overall number of tasks in a path. At this step, Article matrix elements are calculated.

2. Initial quantity allocation.

To determine the initial population, we randomly create a number of paths, i.e., a number of vectors that will take us from the start point to the end point (Fig. 1). Considering the fact that we are using the optimum version of GSA, in this stage the paths that do not answer the user's global constraint are substituted with new paths and the constraint is again reconsidered for new paths. Therefore, at the end of this stage all members of the population are optimal; that is, they answer the user's global constraint.

3. Mass evaluation.

In the proposed model, each mass is a path consisting of some Web services. These Web services may relate to each other based on different composition operators. The path is divided into blocks and in each block the fitness value is calculated based on the composition operator. Finally, the path fitness is calculated.

4. Updating the parameters G, best, worst, Mi, and Mg.

Using Eqs. (9) and (10) Mi and Mg are calculated for each mass. The path with the highest fitness among the paths determined is chosen as the best and that with the least fitness, as the worst. Since the discrete version of this algorithm is used, the formula for $G(t)$ is reduced to a linear form:

$$G(t) = 2 - \frac{t}{T}, \tag{26}$$

where T represents the total number of iterations.

5. Calculation of force.

As mentioned in Section 3.1, to calculate the force between two masses, Eq. (2) is used; for the purpose of correlation between the equation and Web service combination, the distance between two masses i and j in dimension d is defined as

$$X_j^d(t)-X_i^d(t)=\begin{cases}0, & X_j^d(t)=X_i^d(t),\\ 1, & X_j^d(t)\neq X_i^d(t),\end{cases} \tag{27}$$

where d shows the path dimension or task d and may be explicated as follows: since our aim is to calculate the forces that masses exert on each other, in each dimension (for the execution of each task), if the service numbers of the two masses are the same, which means they use the same service to carry out that task and $X_j^d(t)=X_i^d(t)$, then $F_{ij}^d=0$ and they do not exert any force on one another. As a result, they use the same service number as before to execute the task. Otherwise, the two masses use different services to execute the same task and must exert a force in proportion to their Mg on each other.

In Eq. (3) which is used to calculate the amount of force, R_{ij} is the Euclidian distance between the two masses, and in our model, it is a matrix in which the rows and columns are equivalent to the number of population members and each element of R_{ij}, which is different between the path of any row i and the path of any column j, shows the number of services.

Therefore, to calculate the force existing between the two paths i and j for task d, if they use the same service number, they will not exert any force on each other; otherwise, they will exert a force equivalent to a fraction of the number of different services. By adding the forces present in dimension d, the forces exerted on mass i in dimension d are calculated using Eq. (4).

6. Calculating the acceleration and velocity of masses.

Since masses exert force on one another, each causes acceleration and brings about a change in the velocity of the other. Using Eqs. (5) and (6) mass acceleration and velocity in each dimension (each task) can be calculated.

7. Updating the position of masses.

Any two masses exert force on each other, mutually causing change in acceleration and velocity. The mass with a higher degree of fitness, however, exerts a stronger force on that with a lower degree. For this reason, and also due to the fact that the problem is defined for a discrete space, update the mass position in each dimension by

$$X_i^d(t+1)=\begin{cases}\left[\left\lfloor X_i^d(t)+V_i^d(t+1)\right\rfloor \%l\right]+1, & V_i^d(t+1)\geq\alpha,\\ X_i^d(t), & V_i^d(t+1)<\alpha.\end{cases} \tag{28}$$

The first rule is used for masses with a higher fitness degree, and the second for those lower in terms of fitness.

In other words, after calculating $V_i^d(t+1)$, its value is compared to α. If it is less than α, this means the change in velocity is small and there is no need for change in position (i.e., change in the number of the services used); otherwise, the service number is added to the calculated velocity and the result is divided by the total number of candidate services and the remainder of this division is added to one, and thus another service is chosen randomly to be used for the task. Based on experiment, the most suitable value for α was set to be 0.5.

8. If the condition for termination is not met, go to step 2. The condition for termination was considered as a specific number of repetitions, T.

9. End.

Fig. 3 gives the pseudo code of this algorithm, which is an extension of the algorithm presented in our previous work (Zibanezhad *et al*., 2009).

4.4 Hypothetical model for Web services combination based on qualitative measures and the PSO algorithm

The first three stages of the PSO algorithm are exactly the same as those of GSA. To calculate particle velocity we employ the following procedure.

Since the binary version of the PSO algorithm is used, to calculate the particle velocity in dimension d, Eq. (13) is used. According to the explanation given, the first part of the right-hand side of Eq. (13) calculates an index of the velocity in the previous stage. The second part calculates the difference between the particle's position and its best previously attained position. As mentioned, index d of vector $\boldsymbol{X}$ shows the number of the services used for task d. Thus, according to Eq. (29):

```
INPUT: QoS_Matrix, Weight_Matrix, GlobalConstraints_
       Matrix, composite Web service structure
OUTPUT: Solution, the best path for creating the composite
        Web service

BEGIN
// generate the first population randomly (paths for creating
// the composite Web service).
Initialize ();

// examine global constraints for selected gifted people.
FOR total paths DO
  LABEL 1: FOR each QoS measure DO
    IF (the measure is direct AND
    ∑_path QoS_Matrix_[path,k] > global_constraint[k])
    OR (the measure is indirect AND
    ∑_path QoS_Matrix_[path,k] < global_constraint[k])
    THEN
      Generate another random path and examine its
      constraints again;
    FI
  OD
OD

// compute for each atomic service its total quality due to
// the Weight_Matrix which is determined by the user.
F_ij^k = V_ij^k W_k,  0 ≤ W_k ≤ 1,  ∑W_k = 1;

For T times DO
// the fitness of each path is evaluated according to its
// operators (aggregation functions) and blocks.
// using Eqs. (9) and (10) to compute Mg and Mi for each path.
// using Eq. (2) to compute F_ij^d for each pair of services.
  IF (X_j^d(t) ≠ X_i^d(t)) THEN
    F_ij^d = G(t)Mg_j(t) / (R_ij(t) + ε);
  FI
// using Eqs. (4) and (5) to compute acceleration for each
// given dimension (task) of the path.
  a_i^d(t) = (∑_{j=1, j≠i}^{num} r_j F_ij^d(t)) / Mi_i(t);
// using Eqs. (6) and (7) to compute velocity and new position
// for each given dimension (task) of the path.
  IF (V_i^d(t+1) ≥ α) THEN
    X_i^d(t+1) = ⌈⌊X_i^d(t) + V_i^d(t+1)⌋ %l⌉ + 1;
  ELSE
    X_i^d(t+1) = X_i^d(t);
  FI
// examine global constraints for selected gifted people;
// this step is similar to LABEL 1.
OD

END
```

Fig. 3 Pseudo code of the gravitational search algorithm

$$X^d_{\text{pbest}i}(t) - X^d_i(t) = \begin{cases} 0, & X^d_{\text{pbest}i}(t) = X^d_i(t), \\ 1, & X^d_{\text{pbest}i}(t) \neq X^d_i(t), \end{cases} \quad (29)$$

if, to execute task d of particle i, the service number of its best previous location is used, then the second part of the right-hand side of Eq. (13) will equal zero; if a different service number is used, its velocity will be increased in proportion to $C_1 \cdot \text{rand}()$.

The third part of the right-hand side of Eq. (13) enables the calculation of the difference between the particle position and the best overall position. Therefore, according to Eq. (30):

$$X^d_{\text{gbest}}(t) - X^d_i(t) = \begin{cases} 0, & X^d_{\text{gbest}}(t) = X^d_i(t), \\ 1, & X^d_{\text{gbest}}(t) \neq X^d_i(t), \end{cases} \quad (30)$$

if the service number of the best global position is used for the execution of task d of particle i, the third part will equal zero; if a different service number is used, its velocity will be increased in proportion to $C_2 \cdot \text{rand}()$.

Updating particle position:

With regard to the velocity calculated in the previous stage, we update the new position of the particle in dimension d. For this purpose, Eq. (31) is used. Its explication is the same as in GSA.

$$X^d_i(t+1) = X^d_i(t) + V^d_i(t+1). \quad (31)$$

5 Evaluation

This algorithm was implemented using a Pentium IV computer with 3 GHz CPU and 1 GB memory, a windows operating system with MATLAB software. Note that finding global optimization quality is our main objective and that inappropriate masses are substituted with other masses randomly. The premise of implementation is that the composite service is constituted by n atomic services, each service having l candidate services. m shows the qualitative parameters that are, for this model, the following six measures:

1. Response time (ms): time taken to send a request and receive a response;

2. Availability (%): the ratio of the number of successful invocations to the number of total invocations;

3. Throughput (s^{-1}): the total number of invocations for a given period of time;

4. Success ability (%): the ratio of the number of responses to the number of request messages;

5. Reliability (%): the ratio of the number of error messages to the number of total messages;

6. Latency (ms): time taken for the server to process a given request.

Measures 1 and 6 are indirect measures and the remainder, direct. For the quantity allocation of the service qualitative parameters, a quality of Web service (QWS) Internet dataset (Liu *et al*., 2007) was used (Al-Masri and Mahmoud, 2007a; 2007b; 2008). In this dataset, the degree of the qualitative parameters of a set of services was measured. For utilization, the similar services in terms of function were grouped together.

In the following, num represents the population size, l the number of candidate services for each task, n the number of atomic services, and m the number of qualitative measures.

Fig. 4 shows that, with the initial premise num=10, l=10, n=10, and m=6, with an increase in the number of algorithm repetitions, fitness increases significantly. Since this algorithm is of the collective-intellect type, with a higher number of program executions and the passage of time, masses have a better mutual impact on each other, and the problem moves more quickly towards better fitness.

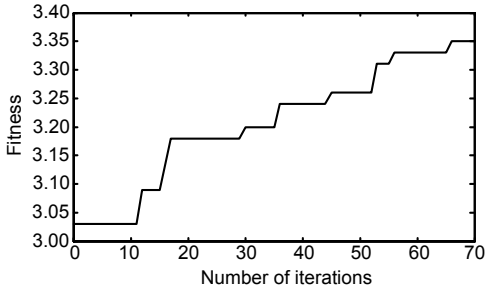

Fig. 4 Fitness change with increase in the number of algorithm repetitions, with num=10, l=10, n=10, and m=6

Fig. 5 shows that, with the premise num=10, T=10, n=10, m=6, there is no specific relation between the change in the number of candidate services and the fitness of the composite service. Considering the present expansion of the Internet and the regular increase of Web service providers, with regard to the fact that this algorithm is unyielding to increase in the number of candidate services, it is, consequently, an efficient, practical algorithm.

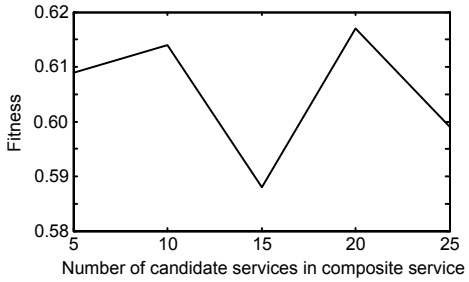

Fig. 5 Fitness change with increase in the number of candidate services, with num=10, T=10, n=10, and m=6

Fig. 6 shows that, with the increase in the number of atomic services, the algorithm execution time increases, which is normal. For additional information concerning the functional features of GSA, the PSO algorithm was simulated in a very similar way. The reason for using the PSO algorithm for comparison is the inherent similarity between these two algorithms: in both algorithms the searcher agents that mutually affect each other play the main role. We present a number of example results.

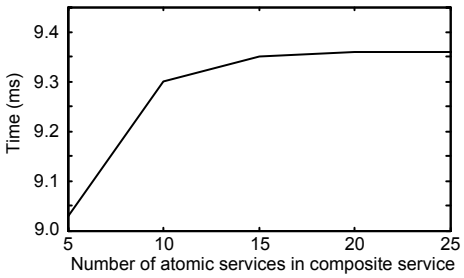

Fig. 6 Fitness change with increase in the number of atomic services, with l=10, num=10, m=6, and T=80

Fig. 7 shows that, in the simulation of the two algorithms, with similar initial premises, an increase in the number of algorithm repetitions results in GSA moving much more quickly towards the convergence point (i.e., finding the fitter composite Web service) in comparison with the PSO algorithm. This is one of the most important reasons for the superiority of GSA in this case.

As can be seen in Fig. 7, since both algorithms are of the collective-intellect type, a higher number of program executions and the passage of time cause a higher degree of mutual effect between the agents and

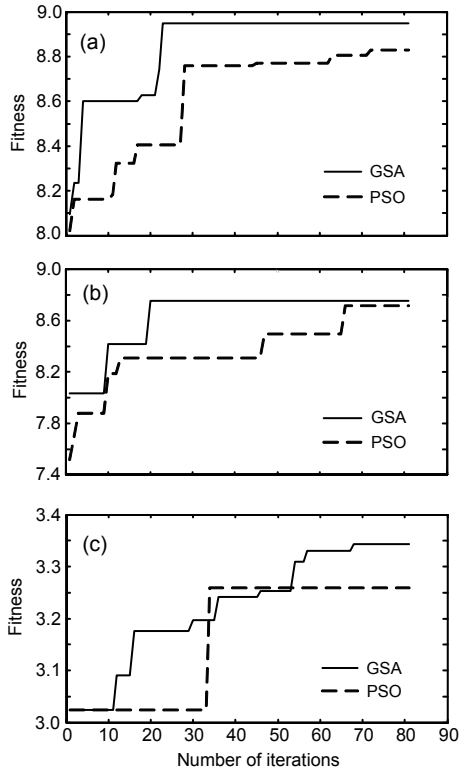

Fig. 7 Fitness change with increase in the number of algorithm repetitions

(a) l=15, m=6, n=15; (b) l=10, m=6, n=15; (c) l=10, m=6, n=5

the movement of the problem towards the convergence point and higher fitness. In this regard, however, GSA is significantly better than PSO. Two of the most important reasons for this superiority are: (1) In the PSO algorithm, the direction of an agent is determined by calculating the best global position of all agents and the best local position of that agent. In GSA, the direction of agents is determined on the basis of all forces exerted by all agents, which enables a more speedy convergence, i.e., finding the fitter combination in a shorter time. (2) In the PSO algorithm, updating the location of masses occurs without regard to the distance between the solutions, whereas in GSA, the force parameter is in inverse relationship with the distance between the solutions. Therefore, the difference between solutions (the number of different services in two compositions) is also effective in updating mass position and this causes a more rapid convergence.

Fig. 8 shows the change in fitness with an increase in the number of candidate services. In the simulation, both algorithms are resistant to the increase in the number of candidate services. In this regard, the behaviors of these two algorithms are similar. In most cases, GSA fitness is of a higher level in comparison with the PSO algorithm.

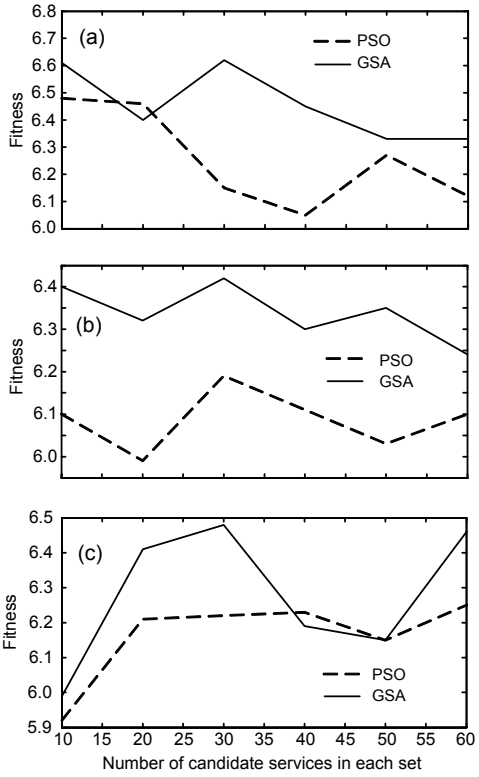

Fig. 8 Fitness change with increase in the number of candidate services

(a) n=5, num=10, T=80, m=6; (b) n=10, num=10, T=80, m=6; (c) n=15, num=10, T=80, m=6

Fig. 9 shows the change in fitness on the basis of the number of atomic services. With an increase in the number of atomic services, the average of the resulting fitness decreases; this is normal. What is interesting is that the gradient showing the average decrease in fitness in the PSO algorithm is significantly sharper than that in GSA. This means that GSA shows a more logical reaction to the increase of atomic services. Also, the resulting fitness for GSA is higher in comparison with the PSO algorithm.

We investigated composition operators such as sequence, parallel, loop, and selection. Fig. 10 presents the sample on which we examined our algorithm. In our proposed algorithm, the composite algorithm structure is divided into blocks. The fitness value is

calculated for each block based on the operator applied, and the total fitness is calculated at last. Table 1 presents the details of service-to-blocks division and finally composing blocks to calculate the fitness value of the sample.

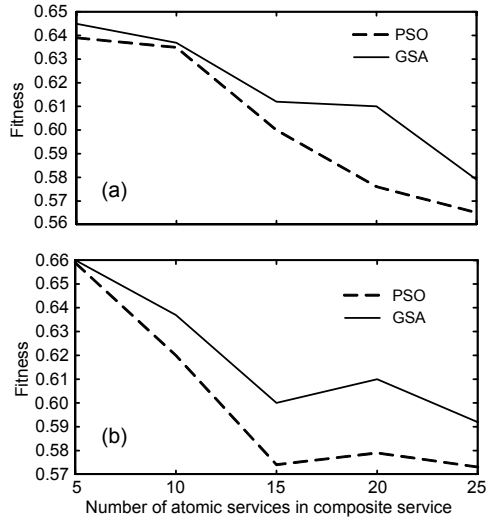

Fig. 9 Fitness change with increase in the number of atomic services

(a) l=10, num=10, T=80, m=6; (b) l=15, num=10, T=80, m=6

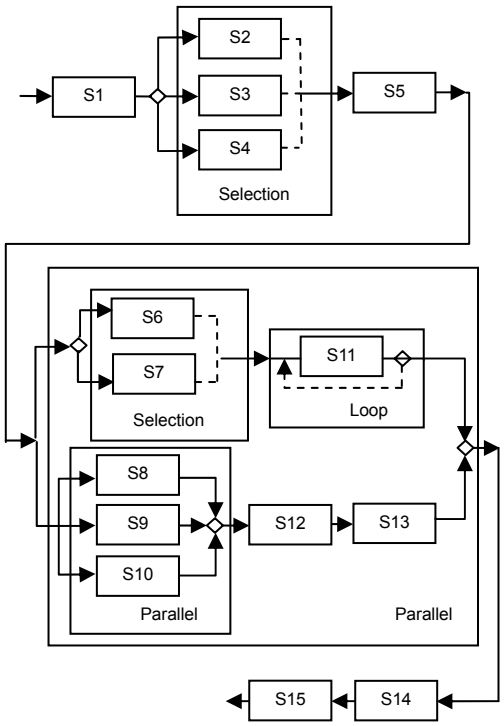

Fig. 10 A sample with different operators on which our proposed algorithm was evaluated

Table 1 Details of each step of the proposed algorithm executed on the sample given in Fig. 10

Step	Block	n_{cws}	n_b	List of Web services and blocks	Operator
1	B1	3	0	S2, S3, S4	Selection
2	B2	2	0	S6, S7	Selection
3	B3	1	0	S11	Loop
4	B4	0	2	B2, B3	Sequence
5	B5	3	0	S8, S9, S10	Parallel
6	B6	2	1	B5, S12, S13	Sequence
7	B7	0	2	B4, B6	Parallel
8	B8	4	2	S1, B1, S5, B7, S14, S15	Sequence

n_{cws}: number of candidate Web services; n_b: number of blocks

As mentioned in Chen and Wang (2007), the PSO algorithm is better in finding the optimized selection with higher fitness than the genetic algorithm (GA). In this work, GSA was evaluated under the same conditions and with the same fitness function. Figs. 11 and 12 show the results of executing GSA and the PSO algorithm on the sample given in Fig. 10. Even by considering composition operators in Web services, the results improved more than before. This is because of the ability of GSA in producing more varied random numbers and indeed moving in a wider domain of response.

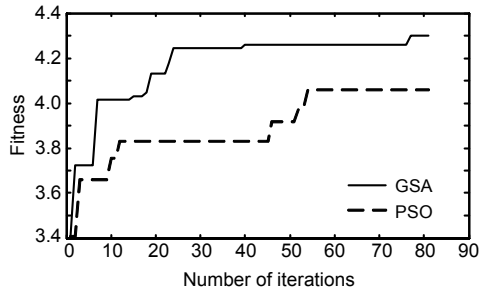

Fig. 11 Fitness change with increase in the number of algorithm repetitions, with n=10, l=12, m=6, num=4, and T=80

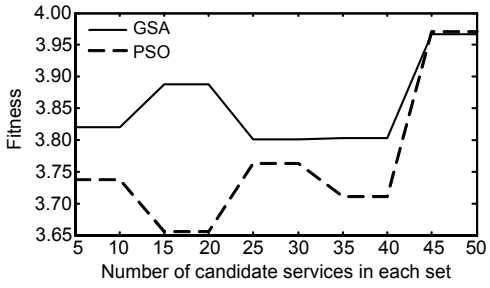

Fig. 12 Fitness change with increase in the number of candidate services, with n=10, m=6, num=10, and T=80

However, the computation complexity of GSA is higher than that of PSO; thus, GSA is more time-consuming than PSO (Figs. 13 and 14).

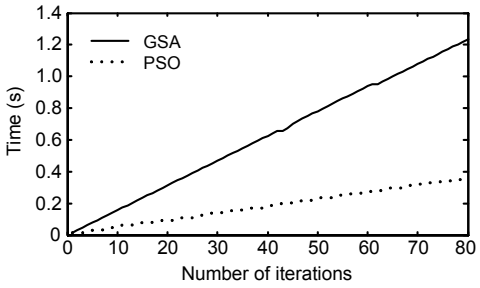

Fig. 13 Change of time with increase in the number of algorithm repetitions, with n=10, m=6, num=10, and l=10

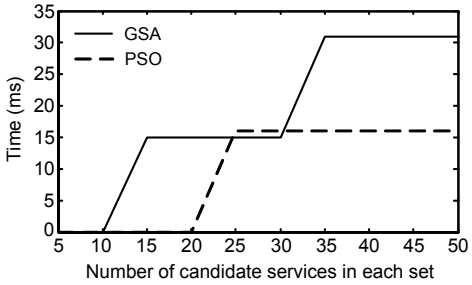

Fig. 14 Change of time with increase in the number of candidate services, with n=5, m=6, num=10, and T=80

6 Conclusions

In this research, the gravitational search algorithm, a natural type algorithm derived from the concept of gravitational force between masses, is used to determine the best Web services combination. As mentioned in Chen and Wang (2007), the PSO algorithm is better in finding the optimized selection than GA. We illustrate that GSA is much better than PSO under the same conditions and with the same fitness function. GSA considers the difference between solutions, and therefore moves quickly towards the convergence point in determining the best Web services combination. In addition, it does not use any marginal memory to save probable modes. Also, in designing the targeted evaluation function, the ideal combination is derived from the user's standpoint by giving weights to qualitative parameters and including the global constraints of the user. Simulation results showed that GSA has, in comparison with the PSO algorithm, a significant potential for finding the ideal user combination in the least time and with least memory use, and is therefore a good choice for practical use.

7 Future work

The combination of this algorithm with services interface comparison algorithms is a future aim concerned with the probability of atomic services combination in creating composite services. This issue is important with regard to the daily expansion of the Web and the number of Internet users. By using multi-purpose functions (Yu and Lin, 2005), the algorithm can be extended, and also by using the framework presented in Canfora *et al.* (2008) and a comparison with the genetic algorithm (Canfora *et al.*, 2005; Ma and Zhang, 2008), the evaluation results will be clearer and more comparable.

In the future, we will apply the ontology engineering and Web service annotation techniques to increase the accuracy and improve algorithm complexity. The system can decrease the number of candidate Web services in the discovery phase and find a better path in the selection phase. As one can see in Fig. 15, Web services are annotated and the system can calculate the similarity measure (numbers above each edge) between their inputs and outputs. In this case, the system can find the best selection path by both syntactic matching and semantic measuring.

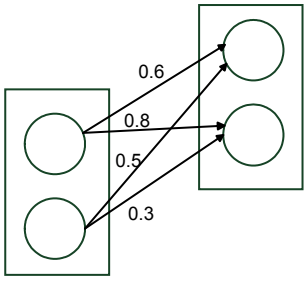

Fig. 15 Web service annotation and calculation of semantic similarity measures between Web services (represented by circles) which would lead to higher accuracy and lower complexity

References

Ai, W., Huang, Y., Zhang, H., Zhou, N., 2008. Web Services Composition and Optimizing Algorithm Based on QoS. 4th Int. Conf. on Wireless Communications, Networking and Mobile Computing, p.1-4. [doi:10.1109/WiCom.2008.2001]

Al-Masri, E., Mahmoud, Q.H., 2007a. Discovering the Best Web Service. 16th Int. Conf. on World Wide Web, p.1257-1258.

Al-Masri, E., Mahmoud, Q.H., 2007b. QoS-Based Discovery and Ranking of Web Services. IEEE 16th Int. Conf. on Computer Communications and Networks, p.529-534.

Al-Masri, E., Mahmoud, Q.H., 2008. Investigating Web Services on the World Wide Web. 17th Int. Conf. on World Wide Web, p.795-804.

Benveniste, A., 2008. Composing Web Services in an Open World: QoS Issues. Proc. 5th Int. Conf. on Quantitative Evaluation of Systems, p.121. [doi:10.1109/QEST.2008.49]

Canfora, G., di Penta, M., Esposito, R., Villani, M.L., 2005. An Approach for QoS-Aware Service Composition Based on Genetic Algorithms. Proc. Conf. on Genetic and Evolutionary Computation, p.1069-1075. [doi:10.1145/1068009.1068189]

Canfora, G., di Penta, M., Esposito, R., Villani, M.L., 2008. A framework for QoS-aware binding and re-binding of composite web services. *J. Syst. Software*, **81**(10):1754-1769. [doi:10.1016/j.jss.2007.12.792]

Chen, M., Wang, Z.W., 2007. An Approach for Web Services Composition Based on QoS and Discrete Particle Swarm Optimization. 8th ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, p.37-41. [doi:10.1109/SNPD.2007.11]

Chen, Z., Wang, H., 2009. An Approach to Optimal Web Service Composition Based on QoS and User Preferences. Int. Joint Conf. on Artificial Intelligence, p.96-101. [doi:10.1109/JCAI.2009.206]

Claro, D.B., Albers, P., Hao, J.K., 2005. Selecting Web Services for Optimal Composition. 2nd Int. Workshop on Semantic and Dynamic Web Processes, p.32-45.

Ismail, A., Yan, J., Shen, J., 2009. Dynamic Service Selection for Service Composition with Time Constraints. Australian Software Engineering Conf., p.183-190. [doi:10.1109/ASWEC.2009.30]

Lecue, F., 2009. Optimizing QoS-Aware Semantic Web Service Composition. Int. Semantic Web Conf., p.375-391.

Lecue, F., Mehandjiev, N., 2009. Towards Scalability of Quality Driven Semantic Web Service Composition. Proc. IEEE Int. Conf. on Web Services, p.469-476. [doi:10.1109/ICWS.2009.88]

Leutenmayr, S., 2007. Selected Languages for Web Services Composition: Survey, Challenges, Outlook. Available from http://www.pms.ifi.lmu.de/publikationen/diplomarbeiten/Stephan.Leutenmayr/Diplomarbeit%20Stephan%20Leutenmayr.pdf

Li, H., Yang, X., Ouyang, Y., 2009. MCHRC: Min-conflict Heuristic Based Web Services Chain Reconfiguration Approach. Int. Conf. on Computational Intelligence and Software Engineering, p.1-4. [doi:10.1109/CISE.2009.5365664]

Liu, A.F., Chen, Z.G., He, H., Gui, W.H., 2007. Treenet: a Web Services Composition Model Based on Spanning Tree. 2nd Int. Conf. on Pervasive Computing and Applications, p.618-623. [doi:10.1109/ICPCA.2007.4365517]

Liu, D., Shao, Z., Yu, C., Fan, G., 2009. A Heuristic QoS-Aware Service Selection Approach to Web Service Composition. 8th IEEE/ACIS Int. Conf. on Computer and Information Science, p.1184-1189. [doi:10.1109/ICIS.2009.76]

Ma, Y., Zhang, C., 2008. Quick convergence of genetic algorithm for QoS-driven web service selection. *Comput. Networks*, **52**(5):1093-1104. [doi:10.1016/j.comnet.2007.12.003]

Maximilien, E.M., Singh, M.P., 2004. A framework and ontology for dynamic Web services selection. *IEEE Internet Comput.*, **8**(5):84-93. [doi:10.1109/MIC.2004.27]

Menasce, D.A., 2004. A composing Web services: a QoS view. *IEEE Internet Comput.*, **8**(6):88-90. [doi:10.1109/MIC.2004.57]

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009. GSA: a gravitational search algorithm. *Inf. Sci.*, **179**(13):2232-2248. [doi:10.1016/j.ins.2009.03.004]

Staab, S., van der Aalst, W., Benjamins, V.R., Sheth, A., Miller, J.A., Bussler, C., Maedche, A., Fensel, D., Gannon, D., 2003. Web services: been there, done that? *IEEE Intell. Syst.*, **18**(1):72-85. [doi:10.1109/MIS.2003.1179197]

Talantikite, H.N., Aissani, D., Boudjlida, N., 2009. Semantic annotations for web services discovery and composition. *Comput. Stand. Interfaces*, **31**(6):1108-1117. [doi:10.1016/j.csi.2008.09.041]

Yu, T., Lin, K.J., 2005. Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. Proc. Int. Conf. on Service Oriented Computing, p.130-143.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z., 2003. Quality Driven Web Services Composition. Proc. 12th Int. Conf. on World Wide Web, p.411-421. [doi:10.1145/775152.775211]

Zibanezhad, B., Zamanifar, K., Nematbakhsh, N., Mardukhi, F., 2009. An Approach for Web Services Composition Based on QoS and Gravitational Search Algorithm. Int. Conf. on Innovations in Information Technology, p.340-344. [doi:10.1109/IIT.2009.5413773]