Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 460593, 6 pages
http://dx.doi.org/10.1155/2014/460593

Research Article

Novel Web Service Selection Model Based on Discrete Group Search

Jie Zhai, Zhiqing Shao, Yi Guo, and Haiteng Zhang

Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China

Correspondence should be addressed to Zhiqing Shao; zshao@ecust.edu.cn

Received 2 March 2014; Accepted 11 March 2014; Published 15 April 2014

Academic Editor: Yu-Bo Yuan

In our earlier work, we present a novel formal method for the semiautomatic verification of specifications and for describing web service composition components by using abstract concepts. After verification, the instantiations of components were selected to satisfy the complex service performance constraints. However, selecting an optimal instantiation, which comprises different candidate services for each generic service, from a large number of instantiations is difficult. Therefore, we present a new evolutionary approach on the basis of the discrete group search service (D-GSS) model. With regard to obtaining the optimal multiconstraint instantiation of the complex component, the D-GSS model has competitive performance compared with other service selection models in terms of accuracy, efficiency, and ability to solve high-dimensional service composition component problems. We propose the cost function and the discrete group search optimizer (D-GSO) algorithm and study the convergence of the D-GSS model through verification and test cases.

1. Introduction

We have proposed a novel approach for the verification of service composition with contracts [1]. The approach properties of the generic specification [2] in Tecton [3] are verified by the Violet [4] system. After verification, a global optimum is selected from a number of instantiations of web service composition components with multiple QoS constraints. Compared with other algorithms that evaluate all feasible composition instantiations (e.g., integer programming [5]), evolutionary algorithms (EAs) (e.g., genetic algorithm [6]), which are nature-inspired optimization algorithms, are simple and flexible. Given their characteristics, EAs have been used to solve the service selection problem. We proposed a novel optimization model named discrete group search service (D-GSS) that mainly employs the group search optimizer (GSO) algorithm [7]. The D-GSS model has competitive performance compared with other EAs in terms of accuracy, convergence speed, and ability to solve high-dimensional multimodal problems. On the basis that GSO can solve continuous optimization problems and that service selection can solve discrete instantiations, we present an evolutionary algorithm called discrete group search optimizer (D-GSO) to select the best instantiation that has the lowest cost evaluated by the cost function. The cost function consists of the utility function and the weight for every QoS attribute. We also verify and simulate results to analyze the convergence of the D-GSS model.

The rest of the paper is organized as follows. Section 2 describes the D-GSS model. Section 3 presents a detailed introduction of the cost function, and Section 4 discusses the D-GSO algorithm and applies the algorithm for the problem on searching for the global optimum from discrete instantiations. Section 5 introduces the convergence analysis of the D-GSS model. Finally, Section 6 concludes the paper.

2. Distribute Group Search Optimizer

In this paper, we present a novel algorithm named D-GSS toward the atomic service selection of composing complex services with multiple QoS constraints. The population of the D-GSO algorithm is called a group searching for unknown optima in the services composition problem and each individual in the population is called a member.

In the n-dimensional search space I about composition component, every dimension represents a class of generic service denoted as I_i. The ith member X_i in the space I is denoted as follows:

$$I = \{I_1, I_2, \ldots, I_n\},$$
$$X_i = \{x_i^1, x_i^2, \ldots, x_i^n\}, \quad (1)$$

where $x_i^j \in I_j$. The ith member X_i at the kth iteration has a current position $X_i^k \in R^n$ and X_i^k is corresponding to an instantiation of services composition component.

A head angle ϕ_i^k is the position of the member; $\phi_i^k = (\phi_{i_1}^k, \phi_{i_2}^k, \ldots, \phi_{i_{(n-1)}}^k) \in R^{n-1}$. The search direction of the ith member, which is a unit vector $D_i^k(\phi_i^k) = (d_{i_1}^k, d_{i_2}^k, \ldots, d_{i_n}^k) \in R^n$ that can be calculated from ϕ_i^k via a polar to Cartesian coordinate transformation [7]:

$$d_{i_1}^k = \prod_{q=1}^{n-1} \cos\left(\phi_{i_q}^k\right),$$
$$d_{i_j}^k = \sin\left(\phi_{i_{(j-1)}}^k\right) \cdot \prod_{q=1}^{n-1} \cos\left(\phi_{i_q}^k\right) \quad (j = 2, \ldots, n-1), \quad (2)$$
$$d_{i_n}^k = \sin\left(\phi_{i_{(n-1)}}^k\right).$$

In D-GSO based on GSO [7] inspired by animal behavior and animal searching behavior, a group consists of three types of members: only one producer is assumed to have the lowest cost at each searching bout, and the remaining members are assumed to be scroungers and dispersed members. At each iteration, a group member representing the most promising instantiation and conferring the lowest fitness value is chosen as the producer. It then stops and scans the environment to seek optimal instantiation. The scanning field is characterized by maximum pursuit angle $\theta_{\max}$ and maximum pursuit distance $l_{\max}$. The apex is the position of the producer. All scroungers will join the resource found by the producer according to area copying strategy. The rest of the group members will be dispersed from their current positions for randomly distributed better instantiations. To handle the bounded search space, the following strategy is employed: when a member is outside the search space, the member will return into the search space by setting the variables that violated the bounds into their previous values.

The details of D-GSO (see Figure 1) are introduced as follows.

(i) Suppose that n classes of generic services exist in the n-dimensional composition component; each class has N_i ($1 \le i \le n$) candidate services in a special sequence.

(ii) Define the concrete cost function of the specific composition component. The cost function is defined by the QoS attributes of the component services as well as their integration relationships, such as sequential, parallel, conditional, or loop. Generate initial members from all instantiations and evaluate the members according to the cost function.

(iii) Choose a member with the lowest cost as producer. The producer produces on the basis of the discrete GSO algorithm.

(iv) Randomly select 80% of the remaining members to perform scrounging.

(v) The remaining members will be dispersed from their current instantiations to perform ranging.

(vi) Evaluate all members according to the cost function. If no optimal instantiation with multiple QoS constraints is found, reallocate the role of every member on the value of the cost.

3. Cost Function

A "generic service" is a collection of atomic web services with a common functionality, but different nonfunctional properties (e.g., time and quality). Each atomic service may provide a series of QoS parameters, such as service time, cost, reliability, and availability. Users can set the number of QoS values to be considered and can set the weights of the QoS values according to their requirements. In our study, each user has k QoS attribute constraints in their QoS requirements: $Q_c = [Q^1, \ldots, Q^k]$. We focus on the QoS service selection problem, in which multiple QoS constraints must be satisfied. We present the cost function to help in the selection of the best services. The following steps are involved in the creation of the cost function.

(i) Each QoS attribute must be quantitative. Service functionalities can be evaluated by several QoS properties. Some QoS attributes, for example, security and reliability, are difficult to measure quantitatively. For these criteria, we employ the linguistic expression set $L1 = \{\text{VP}, \text{MP}, \text{P}, \text{M}, \text{G}, \text{MG}, \text{VG}\}$, where VP is very poor, MP is medium poor, P is poor, M is medium, G is good, MG is medium good, and VG is very good. When calculating the cost function, set $L1$ is transformed into the corresponding quantitative set $P1 = \{0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1\}$.

(ii) Global QoS attributes ($q_c = [q^1, \ldots, q^k]$) are needed to describe the performance of an instantiation of service composition component. Every global QoS attribute is aggregated by the QoS attributes of all atomic services considering the integration relationships of the global QoS attribute. Each service has four main basic structures: (1) the sequential structure, which represents n services that are invoked one by one; (2) the loop structure, which represents one service that is repeated p times; (3) the conditional structure, which represents only one branch that is selected to be invoked from n branches; (4) the parallel structure, which represents n branches that are invoked simultaneously. The complete structure of the service composition component consists of the above four basis structures. Every global QoS

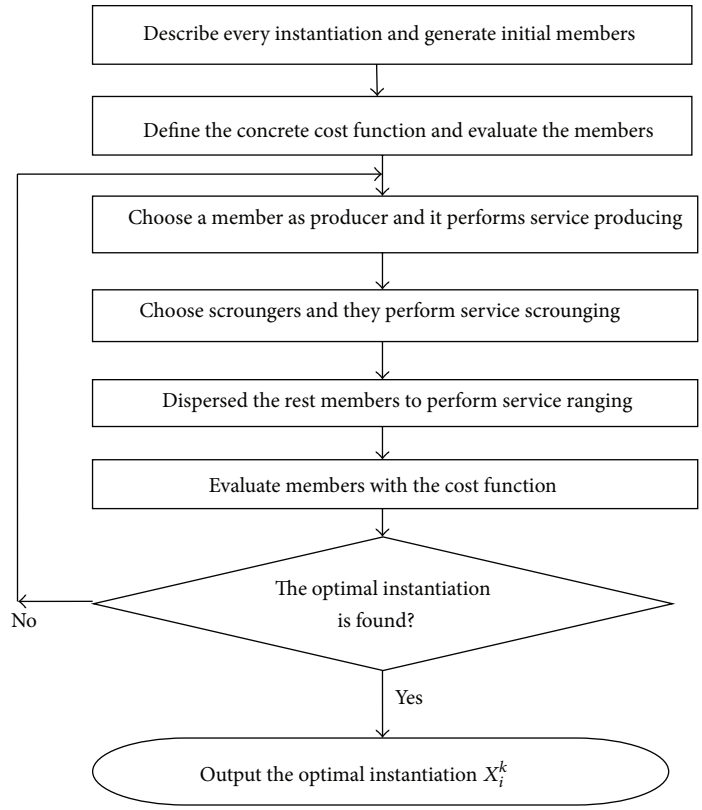

Figure 1: Flowchart of the D-GSS model.

Table 1: Aggregated methods for global QoS attributes.

Method	Sequential	Loop	Choice	Parallel
Summation	$\sum_{i=1}^{n} q^i$	pq^i	$\sum_{i=1}^{n} c^i q^i$	$\sum_{i=1}^{n} q^i$
Continued multiplication	$\prod_{i=1}^{n} q^i$	q^i	$\sum_{i=1}^{n} c^i q^i$	$\prod_{i=1}^{n} q^i$
Average	$\frac{1}{n}\sum_{i=1}^{n} q^i$	q^i	$\sum_{i=1}^{n} c^i q^i$	$\frac{1}{n}\sum_{i=1}^{n} q^i$

attribute has its own aggregated method. We sort the QoS aggregated methods into three types: (1) the summation method (e.g., cost), in which the fees must be accumulated by the user to pay for invoking the services; (2) the continued multiplication method (e.g., availability), in which global availability can be computed as the product of the ratios of all atomic service availability; (3) the average method (e.g., reputation), in which global reputation is the average value of the related service reputation. We present all particulars (see Table 1) of these three methods with sequential, parallel, conditional, or loop structures. In Table 1, c^i is a 0-1 variable. If condition c^i is satisfied, then we define $c^i = 1$; otherwise, $c^i = 0$.

(iii) After the values of $[q^1, \ldots, q^k]$ and $[Q^1, \ldots, Q^k]$ are evaluated, we present a utility function to describe the relationship between q^i and Q^i. Two types of QoS criteria are available, that is, cost and benefit. In the cost criterion, variables (e.g., response time) with higher values have lower qualities. In the benefit criterion, variables (e.g., availability) with higher values have higher qualities. The utility function synthesizes the cost and benefit criteria.

Definition 1 (utility function). Suppose that a global QoS attribute q^i $(1 \leq i \leq k)$ and its constraint Q^i of an instantiation S^j exist, the utility function is defined as follows:

$$U\left(q_i^j, Q_i^j\right) = \begin{cases} \dfrac{q_i^j}{Q_i^j}, & \text{if } q_i^j \text{ is the cost criterion,} \\ 2 - \dfrac{q_i^j}{Q_i^j}, & \text{if } q_i^j \text{ is the benefit criterion.} \end{cases} \quad (3)$$

If the global QoS attribute q^i satisfies the requirement of the QoS constraint Q^i, then $U(q_i^j, Q_i^j) \leq 1$; otherwise $U(q_i^j, Q_i^j) > 1$.

(iv) The cost function is based on the values of the utility function and the weights the user defined. The better the instantiation is, the lower the quality of the cost function result becomes.

Definition 2 (cost function). Suppose that an instantiation S^j exists in the QoS attributes $q_c = [q_1^j, \ldots, q_k^j]$, QoS constraints $Q_c = [Q_1^j, \ldots, Q_k^j]$, and the weights for each QoS attribute; then the cost function is defined as follows:

$$F\left(X_j, q_c, Q_c\right) = \sum_{i=1}^{k} w_i^j U\left(q_i^j, Q_i^j\right), \tag{4}$$

where $\sum_{i=1}^{k} w_i^j = 1$ and $U(q_i^j, Q_i^j) \leq 1$ $(1 \leq i \leq k)$.

The objective of this paper is to employ D-GSO to get the optimal solution of the following model:

$$\min\left(F\left(X_j\right)\right) = \sum_{i=1}^{k} w_i^j U\left(q_i^j, Q_i^j\right), \tag{5}$$

where $X_j \in R^n$.

4. D-GSO Algorithm

The GSO algorithm [7] designs optimum searching strategies to solve continuous optimization problems. However, service selection is a discrete problem. Therefore, we present an evolutionary algorithm named D-GSO that can handle composition components with discrete atomic services. The steps of the D-GSO algorithm are described in Algorithm 1. In the D-GSO algorithm, round(x) represents a round function for half adjust result. Suppose that subX_i^h. represents $[1_i^h, 2_i^h, \ldots, n_i^h]$, which are the subscripts of atomic services composing an instantiation X_i^h about the ith member X_i at the hth iteration. At the $(h + 1)$ iteration, the transformation of the subscripts by the following formulas is $[1_i^{h+1}, 2_i^{h+1}, \ldots, n_i^{h+1}]$ relating to a new instantiation (see Algorithm 1).

5. Convergence Analysis of the D-GSS Model

5.1. Convergence Verification. In this section, we verified the convergence of the D-GSS model. After n iterations, the best instantiation with the lowest cost can be determined with the cooperation of the producer and some scroungers and rangers.

Lemma 3. *If X represents the space of all instantiations X_i^k and P represents the space of the producer, then $X = P$.*

Proof. (1) $l_{\max}$ denotes the maximum distance between two points in space X. By using (3) to (6), we can equate space P to a sphere that has center X_h^p possessing sub(X_h^p) and radius $l_{\max}$. Thus, $X \subset P$.

(2) The following strategy is employed by using the D-GSS model: when a member in space P is outside space X, the member will return into space X by setting the variables that violated the bounds to their previous values. Therefore, $P \subset X$.

(3) Thus, we conclude that $X = P$. □

Theorem 4. *The costs of instantiations in the group will converge to the global optimum that corresponds to the best instantiation with the lowest cost.*

Proof. In the D-GSS model at the hth iteration,

(1) the producer S^p behaves according to (ii)–(iv) in Algorithm 1. By applying the D-GSO algorithm, we can derive the following:

$$\begin{aligned}&\operatorname{cost}\left(X_{h+1}^p\right)\\ &= \min\left(\operatorname{cost}\left(X_h^p\right), \operatorname{cost}\left(X_z\right), \operatorname{cost}\left(X_r\right), \operatorname{cost}\left(X_l\right)\right),\end{aligned} \tag{6}$$

(2) the scroungers X_{h+1}^s will approach the producer through (vii) in Algorithm 1,

(3) the rangers X_{h+1}^r will disperse from a group to perform random walks via (viii) and (ix) in Algorithm 1 to avoid entrapments in the local minima,

(4) finally, we calculate the costs of all instantiations in the group and reallocate their roles. The cost of the new producer is shown as follows:

$$\operatorname{cost}\left(X_{h+1}^p\right) = \min\left(\operatorname{cost}\left(X_{h+1}^p\right), \operatorname{cost}\left(X_{h+1}^s\right), \operatorname{cost}\left(X_{h+1}^r\right)\right). \tag{7}$$

We conclude that $\operatorname{cost}(X_{h+1}^p) \leq \operatorname{cost}(X_h^p)$ by using (6) and (7), which means that the cost of the producer is monotonically decreasing. A global optimum, which has the lowest cost in all instantiations, exists. As stated in the proof of Lemma 3, $X = P$. Therefore, the infimum of $\operatorname{cost}(X^p)$ is cost (global optimum); that is, after n iterations, the instantiation X^p converges to the global optimum. □

5.2. Simulation Convergence Results. The parameter setting of the D-GSS model is summarized as follows. M classes of generic services are present in the complex composition component, in which each class has 50 candidate services that has 10 QoS attributes. The service requestor provides 10 QoS attribute constraints as well as the weights for each QoS attribute. Overall, 51 initial instantiations X^i with $U(q_t^i, Q_t^i) \leq 1$ $(1 \leq t \leq 10)$ are selected at random in all instantiations. The initial head angle ϕ^0 of each individual is set to $(\pi/4, \ldots, \pi/4)$. The constant a is given by round($\sqrt{n+1}$). The maximum pursuit angle $\theta_{\max}$ is π/a^2. The maximum turning angle $\alpha_{\max}$ is set to $\theta_{\max}/2$. Suppose $n = 10, 100$; the relations between the cost of the producer and the iteration times within 500 runs are shown in Figure 2. The experimental results show that the cost of the producer always converges to the optimum of the low- or high-dimensional service composition component.

Algorithm. D-GSO

Step 1. N classes of generic services are present in the composition component, where each class has N_i $(1 \le i \le n)$ candidate services. The maximum pursuit distance $l_{\max}$ is calculated from the following equation:

$$l_{\max} = \sqrt{\sum_{i=1}^{n} N_i^2}, \quad \text{(i)}$$

Each candidate service has k QoS attributes, which are rearranged into $[q_{\text{des}}^1, \ldots, q_{\text{des}}^k]$ according to the weights in descending sequence. The candidate services of generic service I_i $(1 \le i \le n)$ are reordered into a set $I_i^{\text{order}} = [x_i^0, \ldots, x_i^{N_i}]$ with reference to $q_{\text{des}}^1, \ldots, q_{\text{des}}^k$ in turn.

Step 2. Set $h := 0$;
Randomly initialize r instantiations X_i $[x_i^1, x_i^2, \ldots, x_i^n]$ $(1 \le i \le r)$ with $U(q_t^i, Q_t^i) \le 1$ $(1 \le t \le k)$ of services composition component and head angle ϕ_i of all initial instantiations;
Calculate the values of initial instantiations according to the cost function;
WHILE (the best instantiation is not found)
FOR (each instantiation X^i where $U(q_t^i, Q_t^i) \le 1$ $(1 \le t \le k)$ in the group)
Choose the producer:
Find the producer X^p with the lowest cost in the group;
Perform producing:
(a) The producer will scan at zero degree and then scan laterally by randomly sampling three instantiations in the scanning field: one instantiation at zero degree, one instantiation in the right-hand side of the hypercube, and one instantiation in the left-hand side of the hypercube. $r_1 \in R^1$ is a normally distributed random number with mean 0 and standard deviation 1, where as $r_2 \in R^{n-1}$ is a uni-formly distributed random sequence in the range $(0, 1)$;

$$\text{sub}\left(X_z\right) = \text{sub}\left(X_h^p\right) + \text{round}(r_1 l_{\max} D_h^p(\phi_h)), \quad \text{(ii)}$$

$$\text{sub}\left(X_r\right) = \text{sub}\left(X_h^p\right) + \text{round}\left(r_1 l_{\max} D_h^p\left(\phi_h + r_2 \frac{\theta_{\max}}{2}\right)\right), \quad \text{(iii)}$$

$$\text{sub}\left(X_l\right) = \text{sub}\left(X_h^p\right) + \text{round}\left(r_1 l_{\max} D_h^p\left(\phi_h - r_2 \frac{\theta_{\max}}{2}\right)\right), \quad \text{(iv)}$$

(b) The producer will find the best instantiation X^i where $U(q_t^i, Q_t^i) \le 1$ $(1 \le t \le k)$ with the lowest cost.
If the best instantiation has a lower cost compared with the current instantiation, then the best instantiation will be chosen; otherwise, the current instantiation will remain and turn its head to a new randomly generated angle.
$\alpha_{\max} \in R^1$ is the maximum turning angle;

$$\phi^{h+1} = \phi^h + r_2 \alpha_{\max}, \quad \text{(v)}$$

(c) If the producer cannot find a better instantiation after a iterations, then the producer will turn its head back to zero degree;

$$\phi^{h+a} = \phi^h, \quad \text{(vi)}$$

Perform scrounging:
Randomly select 80% members from the rest of the instantiations to perform scrounging.
The area copying behavior of the ith scrounger can be modeled as a random walk toward the producer.
In (vii), $r_3 \in R^n$ is a uniform random sequence in the range $(0, 1)$;

$$\text{sub}\left(X_i^{h+1}\right) = \text{sub}\left(X_i^h\right) + \text{round}\left(r_3{}^{\circ}\left(\text{sub}\left(X_p^h\right) - \text{sub}\left(X_i^h\right)\right)\right), \quad \text{(vii)}$$

Perform dispersion:
The rest of the instantiations will be dispersed to perform ranging: (1) generate a random head angle by using (v); (2) choose a random distance l_i from the Gauss distribution by using (viii); transform into the new instantiation by using (ix);

$$l_i = a \cdot r_1 l_{\max}, \quad \text{(viii)}$$

$$\text{sub}\left(X_i^{h+1}\right) = \text{sub}\left(X_i^h\right) + \text{round}\left(l_i D_i^h\left(\phi^{h+1}\right)\right). \quad \text{(ix)}$$

Calculate fitness:
Calculate the values of the current instantiations according to the cost function;
END FOR
Set $h := h + 1$;
END WHILE

Algorithm 1: Procedure for the D-GSO algorithm.

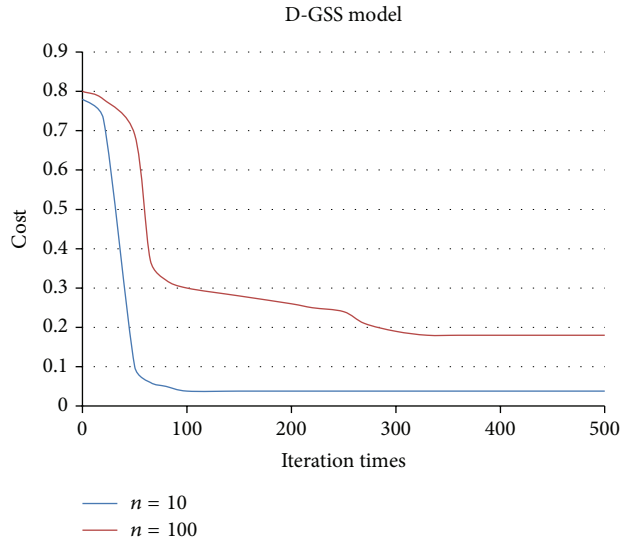

Figure 2: Convergence for $n = 10, 100$.

The experiments were conducted on a PC with 2.50 GHz Intel Processor and 8.0 GB RAM. All programs were written and executed in Java. The operating system was Microsoft Windows 7.

6. Conclusion

In this paper, we describe a new evolutionary approach for multiconstraints service selection on the basis of the D-GSS model. We propose the cost function and the D-GSO algorithm for searching the global optimum from discrete instantiations of the service composition component. The convergence of the D-GSS model is verified via several formal proofs and simulations. This model has an outstanding advantage in terms of solving high-dimensional service composition problems. In the future, we hope to search for the global optimum under a dynamic heterogeneous environment by using the D-GSS model.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by of the National Natural Science Foundation of China under Grant no. 61003126 and no. 61300133.

References

[1] Z. Jie, S. Zhiqing, G. Yi, and Z. Haiteng, "Generic contract-regulated web service composition specification and verification," in *Proceedings of the International Conference on Information Technology and Software Engineering (ITSE '12)*, vol. 3, pp. 137–145, Beijing, China, 2012.

[2] Z. Jie and S. Zhiqing, "Specification and verification of generic web service composition," *Computer Application and Software*, vol. 28, no. 11, pp. 64–68, 2011.

[3] D. R. Musser and S. Zhiqing, "Concept use or concept refinement: an important distinction in building generic specifications," in *Proceedings of the 4th International Conference on Formal Engineering Methods (ICFEM '02)*, pp. 132–143, Shanghai, China, 2002.

[4] Z. Jie and S. Zhiqing, "The proof system based on tecton—violet," *Journal of East China University of Science and Technology*, vol. 31, no. 2, pp. 198–202, 2005.

[5] L. Z. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[6] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "An approach for QoS-aware service composition on algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pp. 1069–1075, Washington, DC, USA, June 2005.

[7] S. He, Q. H. Wu, and J. R. Saunders, "Group search optimizer: an optimization algorithm inspired by animal searching behavior," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 973–990, 2009.