Research Journal of Applied Sciences, Engineering and Technology 11(2): 229-234, 2015
DOI: 10.19026/rjaset.11.1711
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: May 20, 2015 Accepted: June 19, 2015 Published: September 15, 2015

Research Article

Service Composition Optimization Using Differential Evolution and Opposition-based Learning

M.A. Remli, S. Deris, M. Jamous, M.S. Mohamad and A. Abdullah
Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia

Abstract: The numbers of web services are increasing rapidly over the last decades. One of the most interesting challenges in using web services is the usage of service composition that allows users to select and invoke composite services. In addition, the characteristic of each service is distinguished based on the quality of service (QoS). QoS is utilized in optimizing decisive factors such as cost or response time that is required by the user in the runtime system. Thus, QoS and service composition problem can be modeled as an optimization problem. In this study, differential evolution and opposition-based learning optimization methods have been proposed to obtain the optimal solution from candidate services. The results show that the proposed method converges faster than others. Therefore, the method is capable to select better composite services in short time.

Keywords: Differential evolution, opposition-based learning, quality of services, service optimization, web service composition

INTRODUCTION

Web services requester faces challenging tasks when they are looking for a desired web service and at the same time there is no single service that satisfies their request. This challenge can be overcome by service composition, which involves the combination of a number of services to produce complex business processes. Service composition can give benefits and added values in software landscape for numerous areas such as travel agents, government services and biological researches. However, with the increasing number of services, it is obviously different from each service with regard to quality factors such as response time, execution time, cost, reliability and security (Alrifai and Risse, 2010). Therefore, web service composition problem has led to the determination of optimal solutions because these services should be chosen based on Quality of Service (QoS) (Jula *et al*., 2014; Mardukhi *et al*., 2013).

QoS attributes in the web services, including cost, time and quality, are non-functional properties attached in the service description. The properties can be employed to drive the selection of candidate services and later can be invoked when providers are exposing equivalent web services through compatible interfaces (Siadat *et al*., 2013). For example, the user would determine the criteria that they will use to invoke services; the cheaper services, services with fast response time, or services with the highest reliability. Fast composition process is obviously important for QoS services composition. For instance, a user does not want to wait long when they want to use travel booking services. The user will search for hotel and ticket services in order to find cheaper services and long waiting time in the list of candidate services is not acceptable. The situation is similar for biological research, where searching for a huge list of bio-related services can take very long time due to the large amount of bio-related web services that are currently published. Subsequently, this problem is described as "given a set of composite services that include service orchestration's features (referred to as abstract services) and concrete services, how do we obtain an optimal combination of composite services that satisfy QoS constraints in short time".

Figure 1 (www.taverna.org.uk) depicts an example of service composition in biological research, specifically for a case study in gene-pathway information retrieval. This process is also known as workflow composition. A composite service can be described as a process that involves the execution of several activities. For each activity that is assigned as abstract services (*S1-S6*), several concrete services exist. Each concrete service has different QoS properties; response time (t), reliability (r), availability (a) and cost (c). The QoS of the overall composite services is obtained by aggregating the QoS of the component services as shown in Fig. 2. Given m abstract services and n concrete services, there are n^m

Corresponding Author: M.A. Remli, Department of Software Engineering, Universiti Tekonologi Malaysia, 81310 Johor Bahru, Johor, Malaysia

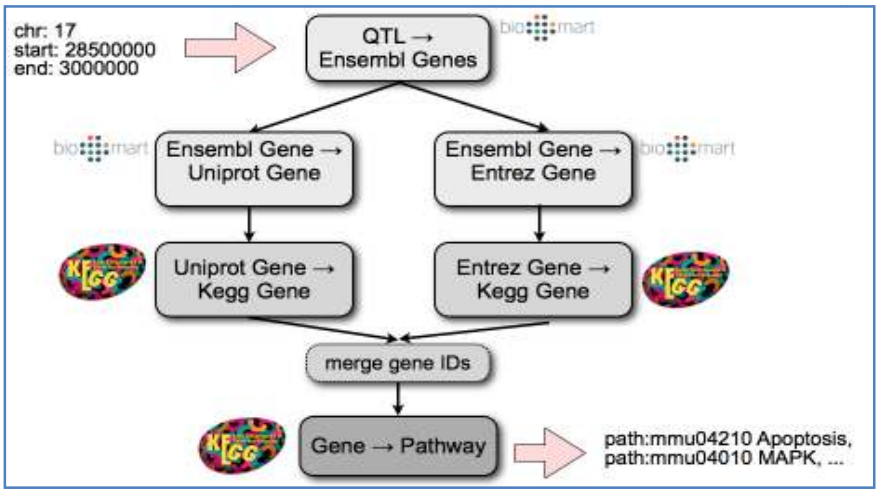

Fig. 1: A gene-pathway abstract retrieval process

possibilities. The search space is discrete since each abstract service needs to choose one concrete service and any combination is possible (Ngoko *et al*., 2013). The solution should be found at runtime. However, finding the solution from this problem is NP-hard. Nevertheless, the problem can be tackled using various optimization algorithms and methods, which are discussed briefly in the next section. Therefore, fast selection is needed in the runtime process to invoke web service composition.

LITERATURE REVIEW

Considering the large amount of web services that exist in the Universal Description Discovery and Integration (UDDI) registry, efficient tools are needed to search services accurately and fast. Furthermore, several interfaces and tools are available for web service discovery process. However, there is still lack of method for searching optimal candidates for service composition. The structure of composite services is described by abstract and concrete services and the method must be capable of finding the best combination of concrete services at runtime. In QoS web service composition problem, determining the number of abstract and concrete services is fundamental. QoS properties attached in the optimization method are crucial when there is a large number of possible candidates' services. Thus, many optimization algorithms have been developed using various strategies to optimize QoS service composition, such as Genetic Algorithm (GA), Differential Evolution (DE) and Particle Swarm Optimization (PSO).

Several works have reported that using evolutionary algorithm such as GA produces promising results. A prominent work from Canfora applied GA to optimize service composition based on QoS constraints. The author stated that GA outperformed Integer Programming (IP) in terms of computational time when the number of concrete services is large. Lecue proposed a GA-basedQoS-aware service composition that overcomes both non-functional and functional levels using semantic web and Description Logic reasoning (DL). The author stated that the work converged faster than the work by Canfora and has better fitness values. Besides GA, other works have used DE, which have similar strategy to GA, except for mutation and crossover operator. For instance, Florin reported that his proposed approach, *LongDE* is capable to converge faster than GA and other DE variants. The author proposed an approach that used DE and integer genome encoding to map abstract and concrete services intogenome. The strategy used in his work is *DE/best/1/bin*, with scaling factor, $F = 0.75$ and crossover constant, $Cr = 0.9$. For swarm intelligence method, Liou proposed an improved PSO that has better accuracy in finding the optimal solution compared to the original PSO. However, his work could not be compared with GA, DE or other types of evolutionary algorithms due to the different environment and platform.

In fact, GA is the most preferred approach to solve this problem based on previous works. GA is the best approach for large search spaces (complex composite services with numerous abstract and concrete services). However, QOS optimization is usually performed at runtime, where a fast algorithm is needed and GA may be slow for this optimization. Previous studies have been conducted and showed that for some general optimization problems, the algorithm based on DE performed significantly better than GA. The following section discusses briefly on the applicability of DE in service composition optimization. Later, the Opposition-Based Learning (OBL) method is presented, as well as how OBL can be combined with DE to improve the convergence speed of the original DE in web service composition problem.

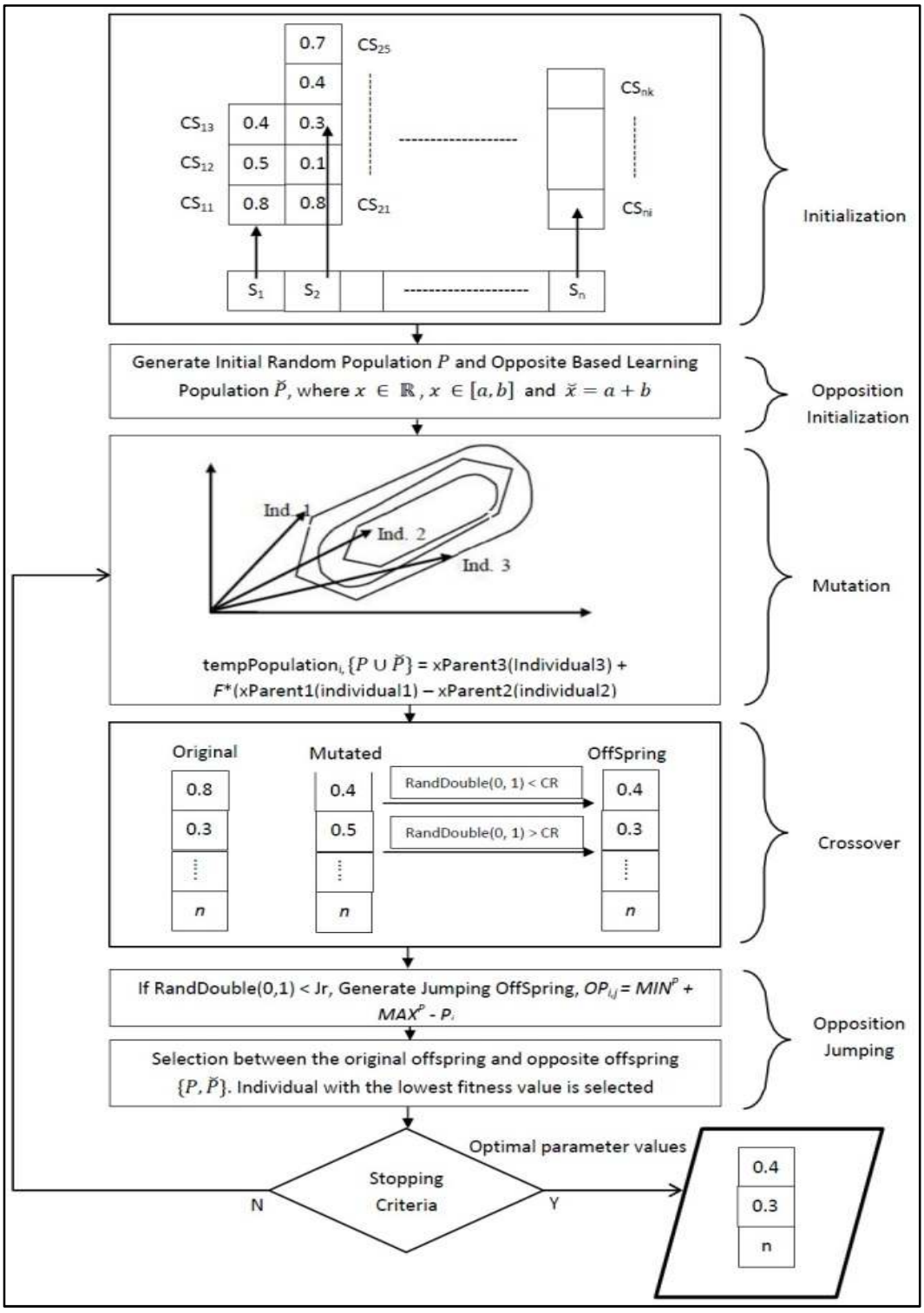

Fig. 2: Overview of the proposed method

DIFFERENTIAL EVOLUTION FOR SERVICES COMPOSITION

Differential Evolution (DE) is a family of evolutionary algorithm, which is similar to GA (Storn and Price, 1997). DE is well applied in optimization problems because of its robustness, effectiveness and simple to implement. In previous works, many authors have reported that DE outperformed many optimization methods in terms of convergence speed and robustness

(Das and Suganthan, 2010). Basically, DE starts by initially generating random populations (candidate solutions) if no prior knowledge about the solution space is available. Let assume that $X_{ri,G}(i = 1,2,..N_p)$ are the vector of solution in generation G, in which $N_p$represents the population size. The next generation of population, called mutation vector, is calculated by adding weighted difference of two randomly selected vectors to a third randomly selected vector as follows:

$$V_{i,G} = X_{r1,G} + F(X_{r2,G} - X_{r3,G}) \quad (1)$$

where, $i = \{1,2, \ldots N_{p\}}$ and $r1, r2$ and $r3$ are different integers randomly selected from {i}. Then, the crossover operator is applied to produce a trial vector from the mutant vector and the target vector. This process is iterated until the best solution, either from the trial or target vector, is chosen. The most commonly used strategy of DE is *DE/best/1/bin*. However, similar to other population-based algorithm, the major drawback of DE occurs in long computation time because of its nature in terms of evolution and stochastic. Some methods that are capable of increasing convergence speed by adding extra rules in DE for service composition problems are explained in the next section.

Opposition-based learning: The original idea of Opposition-Based Learning (OBL) is to estimate the consideration of solution and the opposite solution simultaneously (Rahnamayan *et al*., 2008). The results of both estimate and opposite estimation solution are compared to obtain a better candidate solution. By using the opposite candidate solution, the global optimum can be reached in shorter time than the random candidate solution. In other words, the utilization of OBL during population initialization and for a new population during the evolutionary process can speed up convergence time of DE algorithms. In the application of a higher dimension problem, this definition can be extended as follows. Let $P = (x_1, x_2, .., x_D)$ be a point in dimensional space D, where $x_1, x_2, .., x_D \in R$ and $x_i \in [a_i, b_i] \forall_i \in \{1,2, \ldots, D\}$. The opposite point $\breve{P} = (\breve{x}_1, \breve{x}_2, .., \breve{x}_D)$ is defined by:

$$\breve{x}_i = a_i + b_i - x_i \quad (2)$$

Opposition-based optimization: Let $P = (x_1, x_2, .., x_D)$ is a candidate solution and $f(.)$ is a fitness function to measure the candidate's fitness. Therefore, if the opposite candidate solution is better than the current candidate solution $f(\breve{P}) \geq f(P)$ with $\breve{P} = (\breve{x}_1, \breve{x}_2, .., \breve{x}_D)$, then point P is replaced with $\breve{P}$; otherwise, the process is continued with P (Xu *et al*., 2014).

Table 1: QoS aggregation

QoS Properties	Sequence	Switch	Loop	Flow
Time (T)	$\sum_{i=1}^{m} T(t_i)$	$\sum_{i=1}^{n} p_{ai} * T(t_i)$	$Max\{T(t_i)_{i\in\{1\ldots p\}}\}$	$k * T(t)$
Cost (C)	$\sum_{i=1}^{m} C(t_i)$	$\sum_{i=1}^{n} p_{ai} * C(t_i)$	$k * C(t)$	$\sum_{i=1}^{p} C(t_i)$
Availability (A)	$\prod_{i=1}^{m} A(t_i)$	$\sum_{i=1}^{n} p_{ai} * A(t_i)$	$A(t)^k$	$\prod_{i=1}^{p} A(t_i)$
Reliability (R)	$\prod_{i=1}^{m} R(t_i)$	$\sum_{i=1}^{n} p_{ai} * R(t_i)$	$R(t)^k$	$\prod_{i=1}^{p} R(t_i)$

PROPOSED METHOD

In DE, a suitable genome must be encoded based on service composition problem to enable the evolution strategy to search for an optimal solution. In this study, a genome is encoded as an integer array. Each integer array contains a number of items that represent abstract services. For each abstract service, it contains an index to the array of the concrete services. Each gene encoded in concrete services in CS_i realizes the abstract service S_i. The real value stored in the gene represents the preference for selecting concrete service, which is generated randomly in the interval [0, 1] (Canfora *et al*., 2005). For example, if the service composition workflow consists of 2 abstract services, S_1 and S_2 and for S_1, there are 3 alternative concrete services, where S_2 has 2 alternative services, then the concrete services will be chosen based on the greater value of the corresponding concrete services. These preference values are updated during the evolutionary processes in DE. The fitness is assigned to a composite service function of its QoS attributes by using QoS aggregation as shown in Table 1. The aggregation depends on the composite service architecture, including each of the control construct.

To evaluate the quality of each potential solution, the aggregate objective function is considered in the following equation:

$$F(y) = w_1.R + w_2.A + \frac{w_3}{T} + \frac{w_4}{C} \quad (3)$$

where, w_i are the weights that correspond to the importance of each QoS property to the user and R, A, T, C are the aggregated QoS values for the services workflow (Pop *et al*., 2011a). The original DE was used as the parent algorithm and the OBL studied in this study was hybridized into two parts of DE; opposition initialization and opposition jumping. Firstly, OBL for initializing population was performed based on the preference of services, which produced opposite population, $\breve{P}$. Random population P and opposite population $\breve{P}$ were compared to select a better population that is close to fitness value. Then,

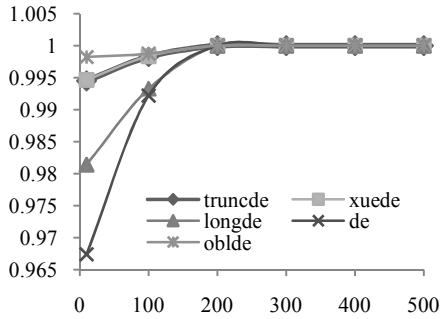

Fig. 3: Evolution of fitness with m = 10 and n = 20

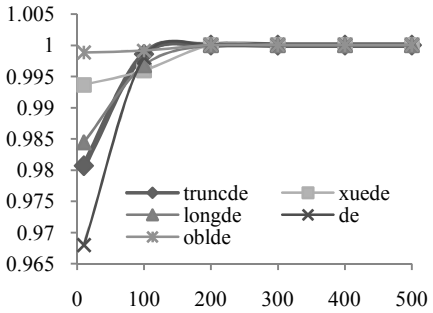

Fig. 4: Evolution of fitness with m = 40 and n = 40

tempPopulation was assigned and mutated by adding weighted difference factor *F* from 3 vectors as shown in formula (1). The mutation and crossover step are classical *DE/rand/1/bin*. Then, by applying the similar approach, the next evolutionary process could be forced to jump to a new candidate solution (which is fitter than the current one) by performing opposition jumping. Opposition jumping calculated the opposite of each variable based on the minimum (MIN_j^p) and maximum MAX_j^p) values of variable in the current population, $OP_{i,j} = MIN_j^p + MAX_j^p - P_{i,j}$ where $i = 1,2,..N$ and $j = 1,2,...D$. For the stopping criteria, a maximum of 500 generation was defined, in which no fitness improvement was observed. Figure 2 depicts the overall process of the proposed method.

NUMERICAL EXPERIMENTS AND EVALUATION

The proposed method has been implemented and compared with the existing method based on the DE that applies discrete optimization approach, including TruncDE, XueDE, LongDE and original DE (Pop *et al*., 2011b). In our proposed method, OBLDE was set with the following parameters: scaling factor, *F = 0.98* and crossover constant, *Cr = 0.9*. The strategy used for OBLDE was *DE/best/1/bin*. The population was set to 100, which involved a maximum of 500 generations. The experiments were run 10 times and the result was averaged. Figure 3 illustrates the result obtained with business workflow consists of 10 abstract services *m* and 20 concrete services *n*. Based on the result, all methods converged within 200 generations, with OBLDE was close to the global maximum. Figure 4 presents a more complex service workflow involving 40 abstract services and 40 concrete services. The result is similar with the previous result. Therefore, the proposed OBLDE converges faster than other methods.

Moreover, CPU user time was compared with similaralgorithms such as original DE, GA and IP. The number of concrete services was set to 25 and the experiments were executed 10 times and the average values were computed as shown in Fig. 5. The

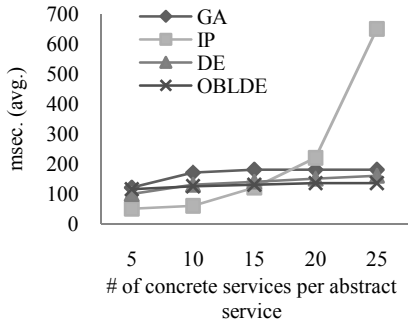

Fig. 5: Comparison of performance with GA, IP and original DE

experiments were performed on a 2.40 GHz Intel(R) Core 2 Duo computer with 1.5 GB, Windows 7 and JRE7. Other methods like IP performed well when the concrete service was small. Substantially, GA, DE and OBLDE demonstrated similar performance. OBLDE requires slightly less computation time due to better population of candidate services, which is utilized with OBL before and during the optimization process.

CONCLUSION AND RECOMMENDATIONS

This study proposes a new method for optimizing web service composition using Differential Evolution (DE) and Opposition-Based Learning (OBL) based on QoS properties. The method used DE as the parent algorithm and OBL to improve better candidate services during the initializing population and evolutionary processes. Prior to initialization, the genome is encoded using integer array that stores values of concrete and abstract services. The values are considered as service preference values that lead to the choice of better concrete services in the optimization algorithm. The opposite numbers of service preference value are obtained and compared with randomly generate values. Thus, better service value was selected and served as the initial population. The similar approach, called generation jumping, was used to force new and better candidates' services in the iterative

processes. The parameters for the proposed method were set to scaling factor, F = 0.98 and crossover constant, Cr = 0.9. The proposed method was compared with several existing optimization methods. The result shows that the proposed method is capable to find better service to be used for service composition in fast selection compared to other methods. For future work, the researchers intend to simulate more detailed real scenario of service composition using travel agent system and bioinformatics workflow application.

ACKNOWLEDGMENT

This study is partially supported by Universiti Teknologi Malaysia with Research University Grant (GUP), Vot Number 04H34 and 4S102. The authors also gratefully acknowledge helpful comments and suggestions of the reviewers, which have improved the presentation of this study.

REFERENCES

Alrifai, M. and T. Risse, 2010. Efficient QoS-aware service composition. Ws. So. Ag. Te., 3: 75-87.

Canfora, G., M.D. Penta, R. Esposito and M.L. Villani, 2005. An approach for QoS-aware service composition based on genetic algorithms. Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05), pp: 1069-1075.

Das, S. and P.N. Suganthan, 2010. Differential evolution: A survey of the state-of-the-art. IEEE T. Evolut. Comput., 15(1): 4-31.

Jula, A., E. Sundararajan and Z. Othman, 2014. Cloud computing service composition: A systematic literature review. Expert Syst. Appl., 41(8): 3809-3824.

Mardukhi, F., N. NematBakhsh, K. Zamanifar and A. Barati, 2013. QoS decomposition for service composition using genetic algorithm. Appl. Soft Comput., 13(7): 3409-3421.

Ngoko, Y., A. Goldman and D. Milojicic, 2013. Service selection in web service compositions optimizing energy consumption and service response time. J. Internet Serv. Appl., 4: 19.

Pop, F.C., D. Pallez, M. Cremene, A. Tettamanzi, M. Suciu and M. Vaida, 2011a. QoS-based service optimization using differential evolution. Proceeding of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO'11), pp: 1891-1898.

Pop, F.C., M. Cremene, M.F. Vaida and A. Serbanescu, 2011b. Medical services optimization using differential evolution. Proceeding of International Conference on Advancements of Medicine and Health Care through Technology, 36: 72-77.

Rahnamayan, S., H.R. Tizhoosh and M.M.A. Salama, 2008. Opposition-based differential evolution. IEEE T. Evolut. Comput., 12(1): 64-79.

Siadat, S.H., A.M. Ferreira and P. Milano, 2013. Performance analysis of QoS-based web service selection through integer programming, World Appl. Sci. J., 28(4): 463-472.

Storn, R. and K. Price, 1997. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim., 11: 341-359.

Xu, Q., L. Wang, N. Wang, X. Hei and L. Zhao, 2014. A review of opposition-based learning from 2005 to 2012. Eng. Appl. Artif. Intel., 29: 1-12.