International Journal of Information & Communication Technology Research

IJICTR

ITRC

Volume 9 - Number 1- Winter 2017(9-16)

QoS-aware web service composition using Gray Wolf Optimizer

Meysam Karimi
Department of computer science
Kashan University
Kashan, Iran
Meysam.Karimi84@gmail.com

Seyed Morteza Babamir
Department of computer science
Kashan University
Kashan, Iran
Babamir@kashanu.ac.ir

Received: November 4, 2016- Accepted: March 5, 2017

Abstract— **In a service-oriented application, an integrated model of web services is composed of multiple abstract tasks. Each abstract task denotes a certain functionality that could be executed by a number of candidate web services with different qualities. The selection of a web service among candidates for execution of each task that is led to an optimal composition of selected web services is a NP-hard problem. In this paper, we adapt the Gray Wolf Optimizer (GWO) algorithm for selection of candidate web services whose composition is optimal. To evaluate the effectiveness of the proposed method, four quality parameters, response time, reliability, availability, and cost of web services are considered and the derived results are compared with several Particle Swarm Optimization (PSO) methods. The proposed method was executed in from 100 to 1000 times and the results showed that a better optimal rate (between 0.2 and 0.4) compared with PSO.**

Keywords- Optimal web service composition; Gray Wolf Optimizer algorithm; Particle Swarm Optimization; Service oriented; Quality of service.

I. INTRODUCTION

In recent years, web services as computational models were developed quickly and played significant roles in e-commerce and web-based services. Therefore, the use of convenient and fast web service with atomic functionality has increased. However, for an application consisting of tasks, a combination of web services is used to execute the tasks where each task (called abstract task) is meant for a specific function. For each task, there are a number of candidate web services with the same functionality but with different quality characteristics. An *optimal solution* for execution of an application is a set of selected web services whose combination is the most suitable combination for the application. Since optimal values of quality parameters are not included just in a candidate web service and are found in different candidates, the selection of a candidate web service for execution of a task of the application is difficult. Furthermore, there may be conflict between some quality parameters. Lower cost and faster response time are always desired; however, they are in conflict with each other because a web service with faster (more optimal) response time demands more (less optimal) cost. Hence, it is clear that the web service composition is a combinatorial optimization problem. It is worth noting that the quality parameters play an important role in identifying the best combination of services at runtime. Finding the optimal solutions for web services composition with conflicting quality parameters is a complex problem that cannot be solved in a polynomial time (NP-Hard).

Generally speaking, the QoS-aware (Quality of Service-aware) web services composition is resolved using intelligent computational methods [1-3]. Methods used the PSO algorithm exhibit better results compared with genetic-based methods [4]. In this study, we adapted the Gray Wolf Optimizer (GWO) algorithm to resolve the QoS-aware web services composition. The proposed method was compared with standard PSO (Particle Swarm Optimization) algorithm [5], IDPSO (Improved Discrete PSO) [6] and QIPSO (Quantum-Inspired PSO) [7]. The results showed that the proposed method is more effective. We've already had experience using the adapted GWO for optimization where the GWO results were compared with those of other optimization algorithms [8].

This paper is organized as follows. Section 2 addresses the related literature. The GWO algorithm is described in Section 3. In Section 4, we explain the proposed method and in Section 5, present the results. Finally, Section 6 deals with concludes.

II. RELATED WORKS

A number of studies have been carried out for the QoS-aware web service composition problem. Most solutions are based on the PSO algorithm and exhibit better coverage compared with the genetic optimization algorithm. However, there is still room for optimization of the QoS-aware web service composition [6]. PSO is a population-based evolutionary algorithm in which each particle has a position and velocity and the population of particles saves its best local and global position. Each particle improves its position based on the value of: (1) its position, (2) the best local position (*pbest*) and (3) the best global position (*gbest*). Each particle has a D-dimensional vector in which 'D' represents dimension of the space that the particle wants to search.

Kang et al [5] used PSO to solve the problem of QoS-aware web service composition composed of the following stages. They noted that the results in terms of coverage and execution speed are superior to the genetic algorithm.

I) Reduction of the multi-objective optimization problem to a single-objective one,

II) Initialization of the particles and adjust the parameters of the algorithm,

for each particle{

III) Computation of the fitness value of the particle position as a candidate for the composition,

IV) Comparison of the fitness value of current position of the particle with pbest of the particle and replacement of the *pbest* by the fitness value if the value is more,

V) Comparison of the *pbest* value of the particle with the *gbest* and replacement of the gbest by the pbest if the pbest value is more,

VI) Calculation of velocity and position of the particle using the PSO formulas,

}

Zhao et al. [6] used IDPSO to address discrete QoS-aware web service composition. They modified the PSO position and velocity formulas to resolve the QoS-aware web services composition and showed that the quality of the service based on the composition obtained by IDPSO is higher than PSO. We compared our results with IDPOS.

QIPSO [9] was created by the integration of quantum display of problem space and PSO trying to improve the ability of the PSO algorithm. Jatush and Gangazaran [7] first reduced the QoS-aware web service composition problem to the single-objective optimization and then resolved it through QIPSO. QIPSO contains three basic parts: (1) quantum measurement, (2) quantum interference, and (3) quantum flight.

Quantum measurement is a function to extract binary particles from quantum particles. Consequently, the quantum particles can be transformed to binary vectors in the problem space. *Quantum interference* is a function increasing the composition optimization and decreasing the probability of suboptimal composition. The main purpose of the quantum interference is that the state of each *qubit* tends towards the optimum composition (solution). A *qubit* in quantum computing or quantum bit is a basic unit like a bit in the classical computing. *Quantum flight* is a function allowing a quantum moves from its current position to its next one to enhance the capacity of the search space. A new solution uses standard phrases forming the next position of the particle in the PSO algorithm. It was shown that QIPSO is more effective than PSO and IDPSO.

III. GRAY WOLF OPTIMIZER

Gray Wolf Optimizer (GWO) [10] is a population-based meta-heuristic algorithm that simulates leadership structure and hunting mechanism of gray wolves in nature. Gray wolves prefer to live in a grouping of five to twelve in form of a hierarchical society consisting of four levels: Alpha, Beta, Delta, and Omega.

The Alpha wolves (male or female) are leaders and responsible for deciding on time of hunting, sleeping, waking, and so on. The rest of the wolves in the group are forced to obey the order of Alphas. Alphas prevail over other levels and all their orders must be followed by members of the group.

The Beta wolves (male or female) are subalterns of Alphas and help Alphas in decision-making. They are the best alternatives to the Alphas at the time of death or aging.

The Delta wolves obey Alphas and Betas, but are superior to the Omegas. Omegas are considered as devotees and obey all wolves of their higher levels. They are the last ones allowed to eat.

GWO simulates hunting of gray wolves where the hunting process is divided into three phases: (1) to chase and surround a prey, (2) harass the prey, and (3) attack the prey. For mathematical modeling of the problem, the best solution is considered as Alpha (α). Similarly, the second and third best solutions are considered as Beta (β) and Delta (δ). Remaining candidate solutions are considered as Omega (ω). Hunting (optimization) in GWO is guided by Alphas, Betas, and Deltas while Omegas follow them. Eqs. 1 and 2 are used to surround a prey [10]:

$$\vec{D} = |\vec{C}.\vec{X}_P(t) - \vec{X}(t)| \quad (1)$$

$$\vec{X}(t+1) = \vec{X}_P(t) - \vec{A}.\vec{D} \quad (2)$$

As stated above, wolves should surround the prey first. To this end, the distance of each wolf from the prey is calculated according to Eq. 1 and the next position of the wolf is calculated according to Eq. 2 where '*t*' represents the current run and vectors $\vec{A}$ and $\vec{C}$ are coefficient vectors for distance and prey respectively. $\overrightarrow{X}_p$ and $\overrightarrow{X}$ are the prey position vector and position of the gray wolf, respectively. Vectors $\vec{A}$ and $\vec{C}$ are calculated according to Eqs. 3 and 4 [10]:

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \quad (3)$$

$$\vec{C} = 2.\vec{r}_2 \quad (4)$$

Elements of the vector $\vec{\alpha}$ linearly decrease from 2 to zero during the execution of algorithm. Vectors r_1 and r_2 contain random values in interval [0, 1].

The wolves chase the prey based on positions of Alphas, Betas, and Deltas. Wolves get away from each other for searching (called divergence) but get close to each other to attack (called convergence). To model the divergence, values of the $\vec{A}$ vector are greater than 1 or less than -1. It forces the search agent to diverge and perhaps find a better prey. This practice focuses on exploration and allows GWO to perform a complete search. Another part of GWO, which facilitates exploration, is the $\vec{C}$ vector whose values are random values in [0,1] denoting weight of prey in Eq. 1. It causes the behavior that is more random during hunting leading to find a proper prey; this avoids the local optimization. Note that the $\vec{C}$ vector decreases non-linearly. Moreover, the $\vec{C}$ vector can be interpreted as natural obstacles in the path of the wolves in hunting and preventing them from rapidly reaching the prey. The wolves are able to recognize and surround a prey position. The hunting of prey usually is guided by Alphas and Betas and Deltas participate in hunting in some cases. However, for the optimization problem, there is no information about the exact position of the prey. To model the behavior of hunting, Eqs. 5-7 are used [10].

$$\vec{D}_\alpha = \left|\vec{C}_1.\vec{X}_\alpha - \vec{X}\right|, \vec{D}_\beta = \left|\vec{C}_2.\vec{X}_\beta - \vec{X}\right|, \vec{D}_\delta = \left|\vec{C}_3.\vec{X}_\delta - \vec{X}\right| \quad (5)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.(\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.(\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.(\vec{D}_\delta) \quad (6)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (7)$$

Since there is not the precise estimate of the actual location of the prey, the distance of each wolf from the *best* positions of Alpha, Beta, and Delta is calculated using Eq. 5. The next position of Alpha, Beta, and Delta is calculated using Eq. 6. Using Eq. 7, next wolf position is calculated regarding the average position of Alpha, Beta, and Delta.

IV. PROPOSED METHOD

In this section, an accurate description of the problem is defined and then quality parameters of the QoS-aware web service composition are described. Afterwards, three steps are taken to solve the optimization problem using GWO.

A. *Problem Description*

An abstract description of a workflow is defined as a composition of abstract services indicated by *A* = (*A*$_1$, *A*$_2$, .., *A*$_n$) where *A*$_i$ is an abstract service. Suppose for each abstract service, there are some candidate concrete services that are able to perform the abstract service with different qualities. Concrete candidate services for abstract service *A*$_i$ is shown as $C_i=\{C_{i1},C_{i2},\ldots,C_{im}\}$ where C_{ij} is the *j*th concrete candidate service for abstract service *A*$_i$.

If quality attributes are response time, cost, availability, reliability, quality of the concrete service S is defined as:

QoS(S)=(Time(S),Cost(S),Availability(S), Reliability(S))

The goal is to obtain an optimal composition of concrete candidate services for services of a workflow so that the composed web services have the best QoS. Therefore for each abstract service, say *A*$_i$, the goal is to find solution $S_k=C_{ij}$. For a workflow consisting of abstract services *A*$_1$, *A*$_2$, .., *A*$_n$, an optimal composition consisting of candidate services C_{1j_1}, C_{2j_2} ,…, C_{nj_k} should be obtained with respect to minimizing response time and cost and maximizing reliability and availability. Notation C_{ij_i} indicates the selected concrete service for abstract service *A*$_i$ is j_i where j_i denotes the jth concrete service of the services are candidate for *A*$_i$.

Depending upon the execution of concrete services in serial or in parallel, response time, cost, reliability, and availability are calculated according to Table1 [11].

B. *QoS-aware web service composition using GWO*

To optimize QoS-aware web service composition using GWO, we should first determine: (1) the representation of wolfs, (2) the initial population of wolfs, (3) the fitness function to evaluate wolfs and (4) the mechanism of updating wolfs' positions at the end of each algorithm iteration.

Table 1. Calculation of quality parameters in execution of serial or parallel services [11]

Quality parameter	Serial	Parallel
Response Time	$\Sigma(T_i)$	$Max(T_i)$
Cost	$\Sigma(C_i)$	ΣC_i
Availability	$\Pi(A_i)$	$\Pi(A_i)$
Reliability	$\Pi(R_i)$	$Min(R_i)$

C-1 Wolf representation

One of the most important steps in the GWO design is the representation of a solution (wolf). In a QoS-aware web service composition, a proper solution is shown by a vector with *D* dimensions (called *D*-dimensional vector) in which *D* is the number of abstract tasks of the workflow. Each element of the vector has a value (see Eqs. 5 to 7) indicating index of the concrete service selected from the candidate services.

Consider a workflow consisting of (n=5) abstract services, for instance, where *n* solutions (indicated by vectors $\vec{x}_1$ to $\vec{x}_n$ in Fig. 1) were proposed. Each solution, indicated by $\vec{x}_i$, shows a composition of 5 candidate services. Vector $\vec{x}_1$, for instance, indicates that concrete services 3, 1, 4, 11, and 6 are selected for abstract services 1 to 5.

$$\vec{x}_1 = \{3,1,4,11,6\},\ \vec{x}_2 = \{110,68,63,400,100\}\ \ldots$$

$$\vec{x}_n = \{40,36,92,57,102\}$$

Fig. 1

C-2 initializing population

After representing solutions, a population of solutions should be initialized. Initially, *n* wolves (solutions) are randomly chosen for each abstract task from candidate services in a dataset. Each wolf consists of *d* values (for instance 5 values for the example stated above).

C-3 Fitness Function

A fitness function should be determined to measure wolf's accuracy. For QoS-aware web services composition, wolf's accuracy is measured by its services' quality values and considering the importance (weight) of each service quality. To compute quality of web services, we use the relations stated in Table 1. Table 2 shows typical weights for quality services.

Availability and reliability are *positive* qualities while cost and response time are *negative* ones. While higher values are more desirable for positives, fewer values are sought for negatives. Because qualities values have different scales, they should be normalized. Eqs. 8 and 9 show normalization of positive and negative qualities, respectively [6].

Table 2. Typical weight (importance) for each quality

Parameter	Cost	Availability	Response time	Reliability
Weight	0.1	0.2	0.4	0.3

$$\begin{cases} \dfrac{Q_i^{\max} - Q_i}{Q_i^{\max} - Q_i^{\min}} & Q_i^{\max} - Q_i^{\min} \neq 0 \\ 1 & otherwise \end{cases} \tag{8}$$

$$\begin{cases} \dfrac{Q_i - Q_i^{\min}}{Q_i^{\max} - Q_i^{\min}} & Q_i^{\max} - Q_i^{\min} \neq 0 \\ 1 & otherwise \end{cases} \tag{9}$$

According to Eq. 8, higher (more desirable) value for the positive quality Q_i leads to less value for the fraction; similarly, based on Eq. 9, less (more desirable) value for negative the quality Q_i does to less value for the fraction. Therefore, our optimization problem is a minimization problem.

Given that the Max. and Min. values of reliability and cost are according to Table 3 and reliability and cost values of a service are according to Table 4, normalized values are shown in Table 5.

As stated above, our optimization is a minimization problem; therefore, according to Eqs. 10 and 12, Service1 is more reliable than Service 2 but Service2 is less costly than Service1.

Fitness value of each dimension (indicating a concrete service) in a candidate solution (wolf) is calculated according to Eq. 14. For instance, fitness value of concrete Service 3 consisting of values of 100% for response time, 2.2% for availability, 90% for cost, and 89% for reliability are calculated as follows (for weights, see Table 2).
Table 6 shows Fitness values of services in vector $\vec{x}_1 = \{3,1,4,11,6\}$ (see Fig. 1) as a solution for 5 abstract services of a workflow.

Eq. 15 shows the fitness value of solution $\vec{x}_1$; similarly, the fitness values of $\vec{x}_2$ to $\vec{x}_n$ are calculated and the solution with the smallest value is selected as Alfa wolf and the smallest values greater than Alfa are selected as Beta and Delta wolves, respectively.

Table 3. Min. and Max. values of reliability and cost

Reliability(%)	Maximum	100
	Minimum	20
Cost($)	Maximum	80
	Minimum	20

Table 4. An example of quality values of 2 services

Service 1	Reliability	80
	Cost	60
Service 2	Reliability	70
	Cost	40

Table 5. Normalizing quality values

Concrete service	Reliability(%)	Cost(%)
Service 1	$\begin{cases}\frac{100-80}{100-20}=\frac{20}{80}=0.25 & ` \\ \end{cases}$ (10)	$\begin{cases}\frac{60-20}{80-20}=\frac{40}{60}=0.67 & ` \\ \end{cases}$ (11)
Service 2	$\begin{cases}\frac{100-70}{100-20}=\frac{30}{80}=0.37 & ` \\ \end{cases}$ (12)	$\begin{cases}\frac{40-20}{80-20}=\frac{20}{60}=0.33 & ` \\ \end{cases}$ (13)

$$\text{fitness(cs)}=\sum_{i=1}^{d} weight(quality(i)) * quality(i) \qquad (14)$$

=(0.4*1)+(0.022*0.2)+(0.9*0.1)+(0.89*0.3)=76%

Table 6. Fitness values of the services in vector x_1

Service#	Response time (%)	Availability (%)	Cost (%)	Reliability (%)	Fitness value%
3	100	2.2	90	89	76
1	85	1	33	100	67.5
4	83	23	6	89	65.18
11	73	3.7	12	78	54.54
6	46	10	39	89	89

$$FitnessValue=(\sum_{i=1}^{D} F(i))/D \qquad (15)$$

=(76+67.5+65.18+54.54+89)/5=70.44

C-4 Update wolf position

In GWO, wolves need to update their position at the end of each algorithm iteration according to Alpha, Beta, and Delta (which are the wolves with the best fitness values in the population), to get closer to the prey. The classic GWO is not appropriate for solving discrete problem and since the web service composition has a discrete space (each dimension of web services composition is a representation of one dimension of a concrete service and cannot accept continuous values), we should justify the basic GWO to a discrete problem. Kennedy and Eberhart [4] used a *sigmoid* method to convert continues problems into discrete ones. Mirjalili et al. [12] described different ways of transforming continuous problems to discrete ones. Leading solutions were considered in this study and the results showed that the *hyperbolic tangent function* is more suitable for our problem. Therefore, after using GWO equations (Eqs. 5-7) and calculation of the approximate position of wolves in continuous space, update formulas for wolf position are applied.

The next position of a wolf (calculated using Eq. 7) is used as argument of the hyperbolic tangent function in Eq. 16 and the output of the function is compared with a random number between zero and one. If it is lower than the random value, it means that we do not need to change the concrete service; otherwise, we must replace the concrete service with a new one.

For example, suppose Alpha, Beta, and Delta are defined as follows:

$\vec{X}_{\alpha}=\{500,12,139,41,7\}$, $\vec{X}_{\beta}=\{17,29,219,76,900\}$, $\vec{X}_{\delta}=\{2028,1101,739,606,84\}$

To update solution (wolf) $\vec{X}=\{3,1,4,11,6\}$, distance of each dimension of the concrete service from the corresponding dimensions of Alpha, Beta, and Delta is calculated and then a new approximate position is calculated according to Alpha, Beta, and Delta separately.

$$X_{id}^{new}=\begin{cases}1, & if\ (\left|Tanh((X(t+1)_{id}\right| > U(0,1)) \\ 0, & otherwise\end{cases} \qquad (16)$$

Moreover, the mean of these positions is used in the transition function. Afterwards, output of this function determines whether this service should be replaced or not. The following example shows the calculations stated above.

$$\vec{D}\alpha\_1=|\vec{C}1.\vec{X}\alpha-\vec{X}|\Rightarrow 0.58\times 0.92-0.83=29.10$$

$$\vec{D}\beta 1\_=|\vec{C}2.\vec{X}\beta-\vec{X}|\Rightarrow 0.58\times 0.88-0.83=31.45$$

$$\vec{D}\delta 1\_=|\vec{C}3.\vec{X}\delta-\vec{X}|\Rightarrow 0.58\times 0.95-0.83=27.34$$

$$\vec{X}1\_1=|\vec{X}\alpha-\vec{A}1.(\vec{D}\alpha)|\Rightarrow 0.92-(-1.82)*2.10=144.95$$

$$\vec{X}2\_1=|\vec{X}\beta-\vec{A}2.(\vec{D}\beta)|\Rightarrow 0.88-(-0.89)*31.45=116.22$$

$$\vec{X}3\_1=|\vec{X}\delta-\vec{A}3.(\vec{D}\delta)|\Rightarrow 0.95-(0.78)*27.34=73.77$$

$$\vec{X}1(t+1)=\frac{\vec{X}1+\vec{X}2+\vec{X}3}{3}$$

$$\Rightarrow\frac{144.95+116.22+73.77}{3}=111.64$$

$$TransformValue=\begin{cases}1 & if\ (\left|Tanh(\vec{X}(t+1)\right| > \cup(0,1)) \\ 0 & otherwise\end{cases}$$

Tanh(111.64)=1,U(0,1)=0.41,1≥0.41 ⇒TransformValue=1

Because the value of the transition function is 1, this concrete service is replaced by one of randomly selected candidate services of this abstract service. The presented calculations were done for the first dimension of a candidate solution (wolf). Similarly, this process should be done for all dimensions of a wolf. Pseudo code of proposed method is presented in Fig. 2.

```
InitializePopulation();
currentIteration=0;
While(currentIteration < maxIteration){
      GetQualityParameters(population);
      UpdatePopulation(population, Alpha, Beta, Delta)
      currentIteration++;
}

InitializePopulation(){
       Foreach dimension in Dimension
       {
            SelectRandomService(Repository);
       }
}

UpdatePopulation(population, Alpha, Beta, Delta){
      Foreach wolf in population
      {
            $\vec{D}\alpha = |\vec{C}1.\vec{X}\alpha - \vec{X}|$;
            $\vec{D}\beta = |\vec{C}2.\vec{X}\beta - \vec{X}|$;
            $\vec{D}\delta = |\vec{C}3.\vec{X}\delta - \vec{X}|$;
            $\vec{X}1 = \vec{X}\alpha - \vec{A}1.(\vec{D}\alpha)$;
            $\vec{X}2 = \vec{X}\beta - \vec{A}2.(\vec{D}\beta)$;
            $\vec{X}3 = \vec{X}\delta - \vec{A}3.(\vec{D}\delta)$;
            $\vec{X}(t+1) = \frac{\vec{X}1 + \vec{X}2 + \vec{X}3}{3}$;
            $If\ (|Tanh(\vec{X}(t+1))| > U(0,1))$
                  GetNewConcreteServiceFromRepository();
      }
}
```

Fig. 2. Pseudo code of the proposed method

V. EVALUATION OF THE PROPOSED METHOD

We implemented the QoS-aware web service composition using GWO in C# programming language and tested using PC with Intel® Core (TM) i5, 2.6 GHz and RAM of 8 GB. The optimal rate of the proposed method was compared with PSO [5], IDPSO [6] and QIPSO [7] using the QWS dataset [13] containing 2507 real web services. The optimality rate was calculated using Eq. 17 [14].

$$\text{OptimalityRate}=\frac{OptimalSolution}{InitialSolution} \qquad (17)$$

The “Initial Solution” is the best solution at the end of the first iteration of the algorithm and the “Optimal solution” is the best solution after convergence of the algorithm. Table 7, for instance, shows initial and optimal solutions of algorithms A and B. According to Eq. 17, the optimal rate of algorithms A and B are 0.33 and 0.5, respectively. Since the minimum value of the optimal rate is desired, algorithm A outperforms algorithm B. To compare our proposed method with other methods, the optimal rate is used (Fig. 3). As Fig. 3 shows, the optimal rates of QIPSO and IDPSO are better than PSO and QIPSO outperforms IDPSO when the number of iterations increases. However, the optimal rate of the proposed method shows it outperforms other methods.

A suitable algorithm is the one that produces effective results independently from the number of the algorithm iterations; GWO enjoys such feature. In this study, we run the proposed algorithm 40 times with an arbitrary number of iterations in each execution. Table 8 shows convergence of the algorithm for 10 services.

VI. CONCLUSION AND FUTURE WORKS

In this study, the effective use of GWO for QoS-aware web service composition was investigated. To found an optimal composition of solutions in a discrete space, we modified the basic GWO.

To evaluate the effectiveness of the proposed method, we thought of quality parameters: response time, reliability, availability, and cost for each web service. By comparing results of the proposed method with several variations of PSO, it was shown the proposed method outperforms the various PSO-based methods.

We showed that the GWO was a suitable algorithm to produce effective results independently from the number of the algorithm iterations. In this study to produce effective results and not using non-optimal results, we run the proposed algorithm 40 times with an arbitrary number of iterations in each run.

The disadvantage of the proposed method, however, is that if a web service has the best fitness value it will be selected as the suggested solution; while there may be several similar solutions with lower fitness values but with more user-friendly candidates. In this case, these solutions would stay away from users. Therefore, as future work, we plan to solve QoS-aware web-service composition problem using the *Pareto front* concept without transforming it to the single-objective optimization problem. This would produce good non-dominated results and user would be free to decide between several suggested solutions.

As a future work, we may use Analytical Hierarchical Process (AHP) when we reach to a set of optimal solutions instead of single one. AHP is used when we should choose a solution from a set of alternatives. A solution is chosen from alternatives by considering some criteria influencing on the solutions.

Table 7. Initial and optimal solutions of 2 algorithms

Alg.	Initial Solution	Optimal Solution	Init. S./ Opt. S.
A	0.6	0.2	0.2/0.6=33
B	0.4	0.2	0.2/0.4=0.5

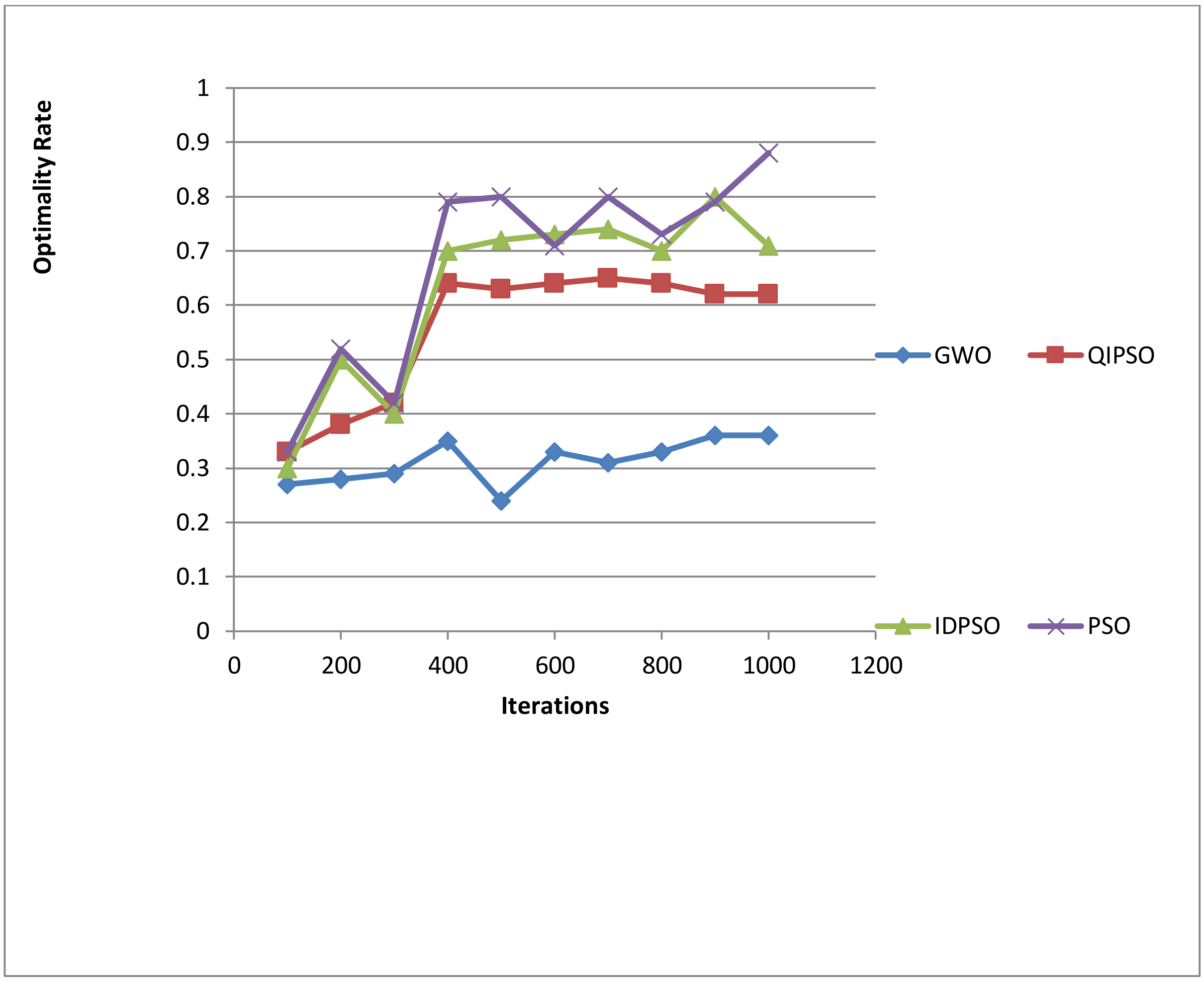

Fig. 3- The optimality rate of PSO, IDPSO, GWO, and QIPSO

Table 8. Convergence of the proposed algorithm
OG: #Optimal Generations, RI: #Run Iterations

	OG	RI		OG	RI
1	32	100	6	401	600
2	131	200	7	325	700
3	242	300	8	597	800
4	211	400	9	702	900
5	273	500	10	654	1000

REFERENCES

[1] A. Strunk, “QoS-Aware Service Composition: A Survey,” The 8th IEE European Conference on Web Services, pp. 67–74, 2010.

[2] M. Amiri and H. Serajzadeh, “Effective Web Service Composition Using Particle Swarm Optimization Algorithm,” The 6th IEE Symposium on Telecommunications, pp. 1190–1194, 2012.

[3] S. Ludwig, “Applying Particle Swarm Optimization to Quality-of-Service-Driven Web Service Composition,” The 26th IEEE International Conference on Advanced Information Networking and Applications, pp. 613–620, 2012.

[4] Q. Bai, “Analysis of Particle Swarm Optimization Algorithm,” Computer and Information Science, Canadian Center of Science and Education, vol. 3, no. 1, pp. 180-184, 2010.

[5] G. Kang, J. Liu, M. Tang and Y. Xu, “An Effective Dynamic Web Service Selection Strategy with Global Optimal QoS Based on Particle Swarm Optimization Algorithm,” The 26th IEEE International Symposium Workshops on Parallel and Distributed Processing, pp. 2280–2285, 2012.

[6] X. Zhao et al., “An Improved Discrete Immune Optimization Algorithm Based on PSO for QoS-driven Web Service Composition,” Applied Soft Computing, Elsevier, vol. 12, no. 8, pp. 2208–2216, 2012.

[7] C. Jatoth and G.R. Gangadharan, “QoS-aware Web Service Composition Using Quantum Inspired Particle Swarm Optimization,” Intelligent Decision Technologies, Springer, pp. 255-265, 2015.

[8] A.Khalil and S.M. Babamir, “A Pareto-based Optimizer for Workflow Scheduling in Cloud Computing Environment”, International Journal Information and Communication Technology Research, vol. 8, no. 1, pp. 51-59, 2016.

[9] A. Layesb, “A Quantum Inspired Particle Swarm Algorithm for Solving the Maximum Satisfiability Problem,” International Journal of Combinatorial Optimization Problems and Informatics, vol. 1, no. 1, pp.13–23, 2010.

[10] S.A. Mirjalili, S.M. Mirjalili and A. Lewis, “Grey Wolf Optimizer,” Advances in Engineering Software, Elsevier, vol. 69, no. 1, pp. 46-61, 2014.

[11] Y. Yao and H.Chen, “QoS-aware Service Composition Using NSGA-II,” The 2nd International Conference on Interaction Science: Information Technology, Culture and Human, ACM, 2009.

[12] S.A. Mirjalili, A. Lewis, “S-Shaped Versus V-Shaped Transfer Functions for Binary Particle Swarm Optimization,” Swarm and Evolutionary Computation, Elsevier, vol. 9, no.1, pp. 1-14, 2013.

[13] E. Al-Masri and Q.H. Mahmoud, "Discovering the Best Web Service," The 16th International Conference on World Wide Web, pp. 1257-1258, 2007.

[14] B. Boussalia and A. Chaoui, “Optimizing QoS-based Web Services Composition by Using Quantum Inspired Cuckoo Search Algorithm. In Proceedings of the Mobile Web Information Systems, vol. 8640, pp. 41–55. Springer, 2014.

Meysam Karimi received the B.E. degree in software engineering from the Tabarestan University, Chalus, Iran, in 2007, and M.Sc. degree in Software Engineering from University of Kashan, Kashan, Iran in 2016, respectively. In 2008, he joined the Lovin Information Technology Company; a software house [illegible] digital repository under .Net technology, as a full-time Developer, and soon was promoted to the project manager in 2009. Since that time, he has been with the Lovin, where he was a technical manager, Scrum Master and part-time developer. In 2010, he joined Ministry of Education as a teacher and he has been teaching computer courses for students eagering to join an associate's degree. Also, he has held lecturing position at Azad University of Islamshahr, Tehran, Iran. He was the recipient of the best teacher of the Conservatory of Mostafa Khomeini in 2014, 2016, and 2017. His current research interests include software engineering, service oriented, cloud computing, distributed systems, evolutionary algorithms, and big data.

Seyed Morteza Babamir received BS degree in Software Engineering from Ferdowsi University of Meshhad and MSc and PhD degrees in Software Engineering from Tarbiat Modares University in 2002 and 2007 respectively. He was a researcher at Iran Aircraft Industries, in Tehran, Iran, from 1987 to 1993, head of Computer Center in University of Kashan, Kashan, Iran, from 1997 to 1999 and haed of Computer Engineering Department at University of Kashan from 2002 to 2005. Since 2007, he has been an associate professor of Department of Computer Engineering at University of Kashan, Kashan, Iran. He authored one book in Software Testing, 4 book chapters, 20 journal papers and more than 50 international and national conference papers (http://ce.kashanu.ac.ir/babamir/ Publication.htm). He is managing director of Soft Computing Journal published by supporting University of Kashan, Kashan, Iran.