QoS aware Web Service Selection using Modified Gray Wolf Optimizer

A DISSERTATION

Submitted in fulfillment of the
requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Ashutosh Agrawal

(14535007)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
MAY, 2016

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the dissertation entitled **''QoS aware Web Service Selection using Modified Gray Wolf Optimizer''** towards the partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Computer Science and Engineering** submitted in the Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand (India) is an authentic record of my own work carried out during the period from July 2015 to May 2016, under the guidance of **Dr. Rajdeep Niyogi, Associate Professor,** Department of Computer Science and Engineering, IIT Roorkee.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

Date:

Place: Roorkee **(Ashutosh Agrawal)**

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date:

Place: Roorkee **(Dr. Rajdeep Niyogi)**

Associate Professor

Department of Computer Science and Engineering

IIT Roorkee

ACKNOWLEDGEMENT

I feel honored in presenting this dissertation report in such an authenticable form of sheer endurance and continual efforts of inspiring excellence from various coordinating factor of cooperation and sincere efforts drawn from all sources of knowledge.

I express deep gratitude for enthusiastic and valuable suggestions that I got from my guide Dr. Rajdeep Niyogi (Associate Professor), Department of Computer Science & Engineering, Indian Institute of Technology Roorkee, for successful completion of this project. His integrity and commitment has inspire me.

I heartily thank professors and technical staff of Computer Science & Engineering, IIT Roorkee, for helping me in exploring the possibility of carrying out the experiments in their laboratory.

Finally, I would like to acknowledge with gratitude, the support and love of my family and friends. They all kept me going.

ABSTRACT

Web service composition has become an important tool to implement complex business processes demanding multiple functionalities to be served. Selecting best set of services for composition becomes crucial when, there are large number of services available serving the same functionality but differing in various QoS attributes. When global constraints on non-functional characteristics like total cost or availability of the composite service, are given the problem converts itself into an optimization problem.

Some solutions using meta-heuristic techniques has been developed for this problem in the past years. We propose applicability of Modified Gray Wolf Optimizer for this problem. Furthermore we compare the results of Genetic Algorithm, GWO and MGWO algorithms and we found that the result obtained from MGWO algorithm is better than GA and GWO, in terms of solution efficiency.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Service-Oriented Architecture is being adapted by most of the organizations to serve user requirements which are composed of multiple functionalities. Composition of web services is done to find the workflow which satisfies the complete user requirement. Continuous development of web services leaves us with multiple web services for same functionality. These web services differ in various nonfunctional characteristics like cost, availability, reliability etc., also referred as QoS parameters of the web service.

Web services selected for the composition are developed independently and are executed in distributed manner. Service interface publication, service discovery and service invocations are performed using XML based standard protocols WSDL, UDDI, SOAP respectively [17]. Complex business processes or user requests which needs composition of web services, are modeled using web service orchestration languages like BPEL4WS, WSCI, BPML. Here each functionality required by the user is represented by an abstract web service for which multiple concrete web services may be available. These abstract services are also referred as service classes and the services available for abstract services are called candidate services.

User of the composite web service can put some constraints on the overall values of the QoS parameters of web services. For example, the overall cost of the composition should not be greater than some specific value or the overall availability of the composition should be at least some given value. These constraints on the composite web services are called global constraints. The problem of selecting an efficient set of services at each service class level, with multiple global constraints so that all the global constraints are satisfied (if possible) and the overall cost of the composition remains minimum, is known as QoS aware web service selection with global constraints. This problem has analogy with the combinatorial problem, the multi-dimensional multiple choice knapsack problem (MMKP), which is a well-known NP-hard problem. Finding solution to such a problem takes exponential time, which can be out of the runtime requirements.

Fig 1 shows the process of web service composition by taking example of an abstract web service, which can be expressed by the languages like BPEL. Web services for any functionality are searched in a common repository i.e. UDDI, where most of the services are registered. Multiple services, serving the same functionality can be present in this repository, which are listed as the result of searching. One service is to be selected from this list for the composition. The combination of the selected services should be such that the QoS requirement of the composition is fulfilled.

The decision of the selection of web service is taken at run time, hence the complexity of the selection algorithm play a crucial role, in areas where quick composition is required. QoS requirement or the global constraints on the composition can be available at run time only for some cases, hence the algorithm should be able to cope with these kind of situations.

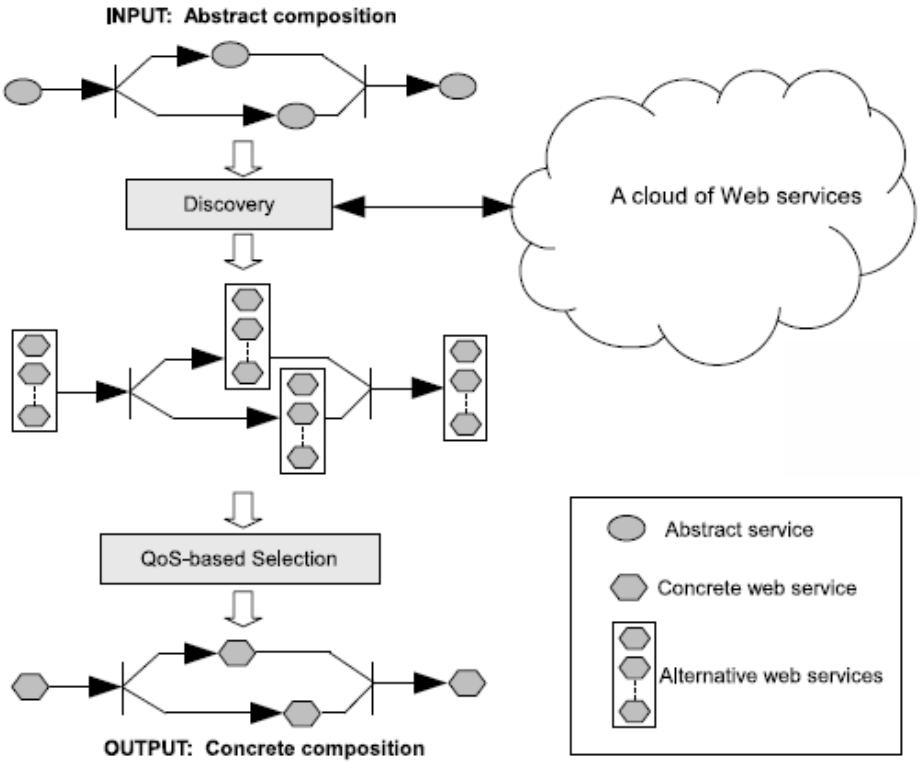


Fig 1. Conceptual Overview

Recently, population based meta-heuristic algorithms, due to their wide-spread applicability in the real-world problems, e.g., economics, biology, engineering design, and information science [1]–[3], have attracted the attention of researchers. Some of the most popular algorithms are: Genetic Algorithm (GA)[4], Differential Evolution (DE) [5], Particle Swarm Optimization (PSO) [6], and Artificial Bee Colony (ABC) algorithm [7]. The Grey Wolf Optimizer (GWO) [8] is latterly put forward population based optimization algorithm, which imitate the hunting and democratic behavior of grey wolves. In this thesis, optimal web services for each service class is selected using modified gray wolf optimizer (MGWO) [9]. Further, to judge the efficacy of MGWO, it is compared against two state-of-the-art algorithms: GA and GWO on the well known

web service selection problem from the QWS data set1. The statistical results obtained from experimental study clearly demonstrate that the MGWO outperforms GA and GWO.

1.1 Local Selection vs. Global Optimization

Web service composition can be done with 2 different strategies: first is Local Selection and second is Global Selection. In local selection strategies given a set of services for any functionality, a web service is selected independently from the other set of services for other functionalities. One utility function is defined prior to the selection which maps the QoS values of different parameters to a single value, which can be used to compare its cost with other web services. And the service which maximizes this utility value is finally selected for the web service composition. The main advantage of this method is that it take O(n), time complexity for each web service selection, where n is the number of different services available for one functionality. But this strategy can't be used for the composition where global (end to end) constraints are applied by the user, because at the time of the selection the end to end criteria of the composite service is not considered and the total value of the composite service for that parameter may exceed the maximum limit declared by the user.

On the other hand in Global Selection strategy used in [10], all possible combination of selections are traversed to reach to the solution which satisfies maximum possible global constraints. The problem of web service selection with global constraints can be modeled as Multi-Choice Multidimensional Knapsack Problem (MMKP), which is an NP hard problem. This solution strategy is practical for the compositions where small set of services are there for web service selection but for the large set of services for each functionality the overall complexity of the solution becomes unreasonable. Hence we need some algorithm which can give us near optimal solution in polynomial time.

1.2 Challenges in Web Service Selection

Below are the main challenges that we may face while composing web services.

1. For any particular operation of the required composition of web services there may be multiple and large number of web services available over internet hence web service selector must be able to search from this huge web service repository.

2. Web services are being developed and updated continuously hence the composition should be able to adopt the changes at run time.
3. Different web service providers use different specification languages and communication protocols hence the composition should be able to deal with this heterogeneity.
4. Handling Compositional structure like loop and parallel etc.
5. QoS value of a service may be different from what is claimed by the provider.(SOA ensures this doesn't happen).

The rest of the report is organized as follows. Section 2 outlines the related work. Section 3 describes the problem and its modeling. Section 4 introduces the proposed methodology. Experimental settings and the simulation results are described in Section 5. Finally, Section 6 concludes and presents a future scope of the work.

2. LITERATURE SURVEY

A very simple brute force solution to the problem of QoS aware web service selection with global constraints, searches for each possible combination of web services for different service class, which takes exponential time. As available services for each functionality are growing day by day, this solution becomes impractical for the situations where quick response is needed. Many researchers tried finding a fast solution to the problem. In [10] linear programming (LP) solution is proposed, which provides efficient solution of the problem. This solution takes less time than brute force solution. In this solution Qbroker architecture is used. In this system a broker is assigned to each service class which is responsible for selection and invocation of the services. Mainly this solution models the problem of web service selection to a linear programming problem, for which objective, constraints and variable needs to be defined. Objective and constraints in a linear programming problem should be a linear equation. In this solution linear objective and constraints are defined for 5 QoS parameters.

Sometimes finding a fast and near optimal solution is better than a slow and optimal solution. [11] gives near optimal solution to the problem in polynomial time. It encodes this problem in two different ways: first, the combinatorial model, in which the problem is defined as MMKP problem. And second is the graph model, which defines the problem as multiconstrained optimal path problem (MCOP).Furthermore two different heuristic algorithms are given for these models. Polynomial time complexity is achieved for combinatorial model in WS_HEU heuristic algorithm. Whereas for graph model (MCSP-K), the complexity achieved in heuristic algorithm is exponential. WS_HEU algorithm starts with finding a feasible solution to the problem, followed by 2 more steps. In the first step it searches for any feasible upgrade. In the last step of the algorithm alternatives are searched for any non-feasible downgrade followed by feasible upgrade. Though this algorithm finds the solution in polynomial time, it incurs a very high communication cost as the algorithm works in multiple iterations. [12] uses mixed integer programming to find the solution. The solution here works in two phases: In first phase mixed integer programming is used to decompose the global constraints into local constraints and in second phase local selection strategies are applied to find the solution. Here continuous quality values are divided into discrete quality levels. The number of quality levels are found by iterative method. This solution has exponential time complexity but it takes lesser time than [10] as the number of binary decision variables are less. This solution also provides near to optimal solution but it is better than [11]. Here the workflow of the composite web

service is assumed to be sequential. Qbroker is used for service selection and service invocation in this solution also. In [13], the work in [12] is extended for non-sequential workflows also. Non sequential workflow is converted into sequential workflow by a step by step process in which at each step a non-sequential construct is replaced by a virtual service class and the process is repeated until the workflow becomes sequential and then constraints on virtual service classes are decomposed until each service class of original composition is assigned with a set of local constraints.

Some Optimization methods are also used to solve this problem. [14] used simple genetic algorithm, in which a genome is represented as an integer array where each index represents a service class and its value represents index of the services available for that service class. For genome in which global constraints are violated, additional penalty is added to the fitness value so that the evolution can be directed towards the solution which satisfies most of the constraints. This solution is faster than integer programming techniques for large scale problems. In [18] algorithm CoDiGA is given, which is an improvement in simple genetic algorithm used in [14]. In this algorithm a relation matrix coding scheme is used so that all paths of composite web service can be expressed at the same time. Initial population and evolution policies are enhanced to quickly converge the algorithm. [15] used simulated annealing meta-heuristic technique in combination with genetic algorithm and developed the algorithm named QQDSGS. Result shown in this paper shows that this algorithm is better than simple GA and CoDiGA. Some other meta-heuristic techniques and its hybridizations are also used to solve this problem e.g. [16] used hybridization of GRASP and PR meta-heuristic techniques to solve the problem etc. Figure2 shows the comparison of some research works in this field.

	Main Idea	Complexity	Assumptions	Research Gaps	Quality of Sol.
1	Algorithm: (1) Find an initial feasible solution. (2) Improve the solution by feasible upgrades. (3) Improve the solution by infeasible upgrades followed by downgrades	$O(N^2 (l-1)^2 m)$	Any feasible solution is considered as the initial solution.	Very high communication cost as the algorithm works in multiple iterations.	Near Optimal
2	Qbroker is used. Linear Programming is used.	Exponential	Only 5 quality parameters are considered	Exponential time complexity.	Optimal
3	Qbroker is used. Global constraints are decomposed into local constraints. Mixed integer programming is used for decompsition. Quality values are divided into quality levels.	Exponential but better than 2. No. of binary decision variables : n*m*d note: d is no. of quality levels.	Workflow is sequential.	Exponential time complexity. Iterative method is used to find the optimal value of d(number of quality levels)	Near Optimal better than 1
4	Work in 3 is extended for hybrid workflows also. Non sequential workflow is converted into sequential workflow by a step by step process in which at each step a non sequential construct is replaced gy a virtual service class and the process is repeated untill the workflow becomes sequential and than constraints on virtual service classes are decomposed untill each service class of original composition is assigned with a set of local constraints.	Exponential but better than 2 No. of binary decision variables : n*m*d	N/A	Exponential time complexity. Iterative method is used to find the optimal value of d(number of quality levels)	Near Optimal better than 1

Figure 2: Comparison of different research works

3. SYSTEM MODEL

3.1 Problem Example and General Notations

Fig 3 is an abstract example of web service composition, which describes the composite service as the set of abstract services. Fig 4 gives the general parameters and the notations generally used in research papers.

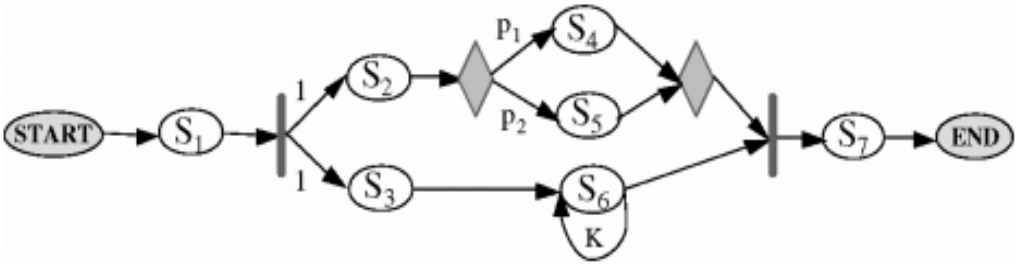


Fig 3. Composite Service abstract Example

S_i	Service class i
s_{ij}	Atomic service j for service class i
$\mathcal{F}$	Utility function
$\mathbf{q_{ij}}$	QoS vector for atomic service $s_{\mathbf{ij}}$, $\mathbf{q}_{ij} = [q_{ij}^1, . . , q_{ij}^n]$
$\mathbf{Q_c}$	QoS requirements vector of a composite service, $\mathbf{Q_c} = [Q_c^1, . . , Q_c^m]$

Fig 4. General Problem Parameters and Notations

Service class is a functionality for which multiple concrete services can be available

Utility Function denotes one single value (or cost) for a web service across all QoS parameters. This function is used to compare two web services.

Atomic Services are the set of services which can be used for any service class. A j^{th} web service for i^{th} service class is represented by s_{ij} .

Qc is a m size vector, where m is the number of QoS parameters, which stores the global constraints for all the QoS attributes.

3.2 Composition Model

The structure of composition for composite web services can be different for different user requirements. For example consider a personalized multimedia news delivery scenario [12]. Figure5 shows the composition structure for this scenario. In this example multimedia news is delivered to smartphone users. The news contains multimedia content (news videos in MPEG 2 only) and news tickers. A transcoding service is needed to convert the news videos in user required format. Also one translation service is required for translating the news ticker. A merging web service is required to merge the results of transcoding and translation services. Finally a compression web service is required to send the data through wireless medium.

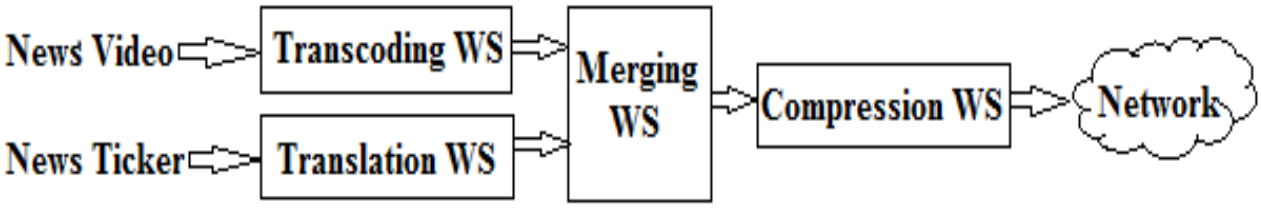


Figure 5: Real Life Composition Example

Figure 6 shows few composition structures by which different web service can be connected in a composite web service. Figure 6(a) shows a sequential construct where web service S2 will always execute after web service S1. Figure 6(b) shows split composition construct in which web services S2 and S3 can be executed in parallel but they are dependent on service S1. Figure 6(c) shows join construct where services S1 and S2 can be executed in parallel but S3 is dependent on S1 and S2 both. Moreover split structure can be of 2 types: first is AND split and second is OR split. For OR split a probability is assigned to each outgoing path by the workflow designer so that the overall cost of the composition can be calculated for different QoS attributes.

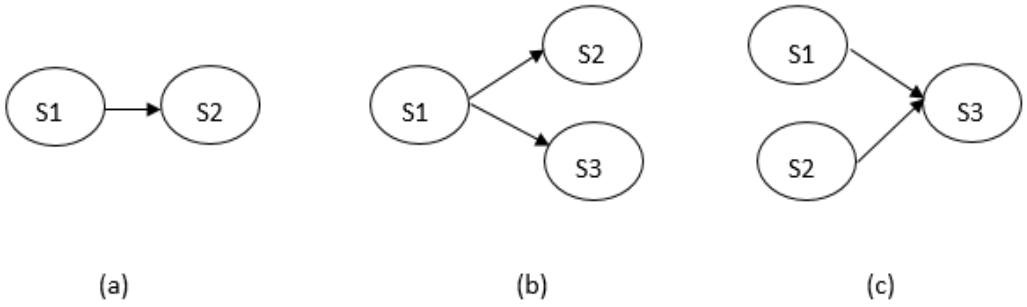


Figure 6: Composition Structures

3.3 *QoS Model*

QoS of web services is given by some non-functional characteristics like cost, efficiency, reliability, availability etc. These attributes are used to compute the QoS value of the composite web service. A quality vector $Q_s = \{q_1(s), q_2(s), \ldots q_n(s)\}$, is used to represent the QoS of a web service s. Here q_i is i^{th} quality attribute and $q_i(s)$ is value of i^{th} quality attribute for web service s. The quality attributes can be of 2 types: positive attributes and negative attributes. The value of the positive QoS attribute should be high for a web service e.g. availability, whereas the value of the negative attributes should be low e.g. cost.

Different aggregate functions are used to find the aggregated QoS value of the composite web service. Aggregate functions for different QoS attributes are shown in Figure 7.

Criteria	Aggregation function
Price	$Q_{price}(p) = \sum_{i=1}^{N} q_{price}(s_i, op_i)$
Duration	$Q_{du}(p) = CPA(q_{du}(s_1, op_1), ..., q_{du}(s_N, op_N))$
Reputation	$Q_{rep}(p) = \frac{1}{N} \sum_{i=1}^{N} q_{rep}(s_i)$
Reliability	$Q_{rel}(p) = \Pi_{i=1}^{N} (e^{q_{rel}(s_i) * z_i})$
Availability	$Q_{av}(p) = \Pi_{i=1}^{N} (e^{q_{av}(s_i) * z_i})$

Figure 7: Aggregation Functions

3.4 *Utility Function*

To evaluate the single quality value of a web service across all quality parameters, a utility function is used. This single quality value is called the utility value of the web service. It is used to compare quality of two web services. Different attribute values can be present in different units like cost in rupees, response time in sec or msec. To bring all the values in uniform platform, scaling is done. In scaling process all the attribute values are scaled between 0 and 1. Different normalization schemes can be used for this purpose. We are using MinMax normalization in this work. The formula for MinMax normalization is given by the following formula.

$$x_{norm}=(x\text{-}min)/(max\text{-}min)$$

Where min is the minimum value and max is the maximum value of the attribute in which normalization is being applied. x is the original value and x_{norm} is the normalized value.

Aggregated QoS value for the composite web service is then calculated using aggregate functions shown in Table I. For the sake of simplicity we have used summation as the aggregate function for all the attributes. As user can give more importance to one attribute than other, weighting process is followed by scaling process. In weighting process all the aggregated values of different attributes are multiplied by the user assigned weights for that attributes.

Utility function can also be of two types: One that should be minimized and the other that should be maximized. In first case all the positive attributes should decrease the utility value and all the negative attributes should increase the utility value. And in second case all the positive attributes should increase the utility value and all the negative attributes should decrease the utility value. One utility function for first case can be as follows:

$$U(CS) = \frac{w1 * q1(CS) + w2 * q2(CS)}{w3 * q3(CS) + w4 * q4(CS)}$$

Here wi is the weight for i^{th} QoS attribute, q1 and q2 are negative attributes, whereas q3 and q4 are positive attributes. The above function represents the utility function for 4 QoS attributes. The utility function for more QoS attribute can be defined similarly by adding the negative attributes in numerator and adding the negative attributes in denominator.

3.5 *Constraint Model*

The user of the composite web service can put some constraints on some attributes of the composed service i.e. the overall value of some QoS parameter can be restricted by the user. For example the total cost of the composition cannot be greater than some given value or the overall availability should be less than some specific value. Constraint for a positive attributes is given by the minimum threshold value for that attribute, and constraint for a negative attribute is given by the maximum threshold value for that attribute. These constraints are given by a vector of integers representing a threshold value for each attribute. A constraint vector C in our approach is represented as follows:

$$C=(c_1,c_2,\ldots c_n)$$

Where c_i is threshold value for i^{th} QoS parameter. The above constraint vector C represents constraints for n QoS attributes.

4. RESEARCH GAPS

- As mentioned in the previous section [7] solves the problem of local selection and gives the best computation cost, but does not consider the global constraints, hence it does not address this problem.
- [2] provides the solution for dynamic as well as quality driven web service selection and considers the global constraints also. This solution uses Linear Programming methods which gives the optimal selection of services which satisfies the global constraints also, but this solution works when the set of services for each service class is small. This solution has poor scalability because of the exponential time complexity of the solution and as the number of services increase the solution becomes unreasonable to use practically.
- [5] and [6] extends the solution given by [2] to include the local constraints also but it still uses Linear Programming which takes exponential time complexity.
- [4] also provides algorithm for QoS aware web service composition and but this paper considers only reliability as the QoS criteria. It gives a polynomial time algorithm to find the best reliable composition, but as it considers only one QoS criteria the solution doesn't address our problem.
- [1] gives heuristic algorithm for web service selection with global constraints and provides the near optimal solution to the problem. The complexity of the algorithm is $O(N^2(l-1)^2m)$, which is polynomial, but the results and complexity of this algorithm can still be improved as it solves the problem in many iterations.

5. PROPOSED APPROACH

5.1 Modified Gray Wolf Optimizer

The Gray Wolf Optimizer (GWO) is a population based meta-heuristic algorithm that is proposed by Mirjalili et al. in 2005 [8] for optimizing numerical problems. The algorithm is specifically based on the hunting and democratic behavior of gray wolves. Generally, the grey wolves live in the pack of 5-12 members on average and all the group members are compelled to follow a strict dominant hierarchy in the group. In the group, the highest position is assigned to alpha wolf (α), followed by beta (β) wolf which holds the second position and works as aide to alpha wolf. The beta wolf deliver the instructions of alpha wolf to all group members and acknowledges their response to alpha. The delta wolf (δ) holding third position in the hierarchy, act like a subordinate to alpha and beta. Finally, omega wolves (Ω) hold the last position in the hierarchy. Another fascinating characteristic of wolves is group hunting which is accomplished in three stages; Chasing, encircling, and attacking.

The algorithm starts with predetermined number of wolves where their initial positions are randomly decided. The best, the second best, and the third best positions are assigned to alpha, beta, and delta, respectively. And the remaining positions are allotted to omega wolves. In order to find the position of the prey, wolves adopt an intelligent strategy. After knowing the position of the pray, they cleverly make an effort to encircle it. Suppose in any generation t of the algorithm, the position of the prey and the wolf is denoted by X(t) and $X^p(t)$, respectively. The mathematical modeling for the encircling process is as follows: [8]

$$D = | C * X^p(t) - X(t) | \qquad (1)$$

$$X(t+1) = X^p(t) - A * D \qquad (2)$$

Where $C = 2r_2$ and $A = 2a*r_1-a$, r_1 and r2 are random numbers selected from interval (0,1), and a = 2-2*t/t_Max [8], which decreases linearly from 2 to 0. Here t_Max denotes the maximum number of generations in the algorithm. In hunting process, alpha act as leader, while beta and delta play the role of subordinate. The location of the prey (optimal solution) is not known previously in most of the cases. However, it is presumed that alpha, beta and delta are superior and have some clue about location of prey and the rest wolves are entailed to reposition themselves according to alpha, beta, and delta using equations (1)-(2).

$$D^{\alpha} = | C_1 * X^{\alpha} - X | ; D^{\beta} = | C_2 * X^{\beta} - X | ; D^{\delta} = | C_3 * X^{\delta} - X | \quad (3)$$

$$X_1 = X^{\alpha} - A_1 * D^{\alpha}; X_2 = X^{\beta} - A_2 * D^{\beta}; X_3 = X^{\delta} - A_3 * D^{\delta} \quad (4)$$

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \quad (5)$$

However, to investigate the advantages and weakness of the GWO, Kishor et al. [9] tested it on numerical bench mark functions. The empirical study divulge that there is problem of premature convergence in the GWO due to lack of diversity in the search space. Further, they have pointed out that only three best solutions share their information with each other and convey their opinion to all members. In other words, the GWO loses diversity as it reliance on three solutions only. Thereby, in case of multimodal problems (where many local optima are present), it can be trapped into a sub optimal point, deteriorates it diversity, and premature convergence occurred. On knowing the drawbacks of the GWO, Kishor et al. [9] proposed a modified GWO (MGWO) to alleviate the local optimal stagnation and resolve the diversity problem. In this modification, the crossover operator of GA is integrated with the GWO. The incorporation of crossover into GWO, ameliorates the search capability as every member can share its information with other pack mates. In this way, MGWO makes proper balance between exploration and exploitation. The pseudo code for MGWO is depicted in Figure 8.

Algorithm 1 Pseudo code of the MGWO algorithm

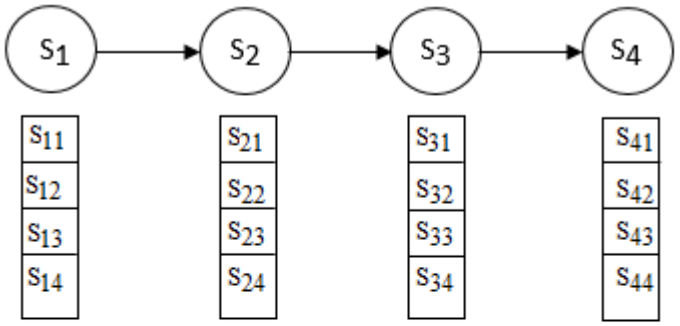
Initialize the position wolves (potential solutions) $X^i (i = 1, 2, ..n)$, *a, A, and C*
Evaluate the objective $f(X^i)$ for all agents and assign top three values to alpha, beta, and delta respectively.
for t = 1 to *MaxGen* **do**
    for all agents **do**
        update X_i by (5)
    end for
    /* Crossover */
    for $i = 1$ to *number of search agents* **do**
        Pick two solutions X^j and X^i randomly where $(X^j \neq X^i)$
        perform two point crossover between X^j and X^i to reproduce X^i_{new}
        compare the fitness of X^i_{new} and X^i and push fitter in the population
    end for
    update *a, A, C,* X^α,X^β, *and* X^δ
end for
return X^α

Figure 8: MGWO Algorithm pseudo code

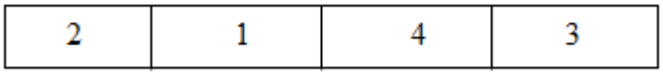
5.2 MGWO for web service selection

5.2.1 Solution Representation

To apply MGWO algorithm, a suitable genome needs to be defined. For this problem a genome can be an array of integers, which represents indices of the concrete web services available for each service class. The size of this integer array is equal to the number of service classes in composite web service. Figure 9(b) shows a genome for an abstract sequential composite web service shown in Figure 9(a). The value 2 at first index of this genome represents second concrete web service S12 for first service class S1. As a whole this genome represents the composition of S12, S21, S34 and S43 concrete web services.



(a) Abstract Composite Web Service



(b) Genome

Figure 9: Solution Representation

5.2.2 Parameters and Methods

To apply Gray Wolf Optimizer to the problem of web service selection with global constraints. We have used population size of 100. And for crossover, standard 2 point crossover is used with crossover probability of 1. As the values in a chromosome represents the indices of the concrete web services available for any particular service class, the values from the algorithm are brought into the range by using the method of clipping. In clipping the value goes beyond the upper bound the value is set to the maximum value and if the value comes out to be lesser than the lower bound, it is set to be the minimum value. And to normalize the values of the penalty values, the difference from the constraint satisfaction i.e. difference of the total value and the threshold value (explained in more detail in the following sub section), is divided by the maximum of threshold and the total value of the attribute. This is done is order to bring the penalty values in the range, which is same as the utility value. The weights for the different QoS attributes are taken to be 1.

5.2.3 Fitness Function

A fitness function is defined for the problem, which signifies the overall value (or cost) of the composition. The solution approaches to minimize (or maximize) the fitness value. In our case utility function can be used as the fitness function, but some penalty parameter needs to be added in utility value to include the constraints satisfaction. Adding this penalty directs the algorithm to reach the solution, which satisfies most of the global constraints. Penalty to be added, can be calculated by finding the distance from constraint satisfaction for each QoS attribute.

Suppose the global constraints for the composition are represented by the vector C=(c1,c2….cn), where ci is the threshold value for i^{th} QoS attribute. The distance from constraint satisfaction for i^{th} attribute can be calculated by finding the difference between ci and the aggregated value of i^{th} attribute. Distance from constraint satisfaction for any attribute, is added to the total penalty only if the constraint is violated. The total penalty P can be given by the following equation.

$$\mathcal{P}(CS) = \sum_{i=1}^{n} d_i(CS) * x_i \qquad (6)$$

Where,

$$x_i = \begin{cases} 1 & ; if\ i^{th}\ constraint\ is\ violated \\ 0 & ; otherwise \end{cases}$$

In above equation d_i(CS) is the distance from constraint satisfaction for i^{th} attribute. And x_i ensures that the distance is added to the total penalty only if the constraint is getting violated. The value of xi is decided differently for positive and negative attributes. If the aggregated value of some i^{th} QoS attribute which is a positive attribute, is less than c_i (threshold value for i^{th} attribute), then the constraint is considered to be violated and the value for x_i becomes 1. On the other hand if the aggregated value of that attribute counts to be more than c_i then, the constraint is not violated and the value of x_i becomes 0. Similarly for negative attributes if the aggregated value of some i^{th} QoS attribute which is a negative attribute, is more than c_i (threshold value for i^{th} attribute), then the constraint is considered to be violated and the value for x_i becomes 1. On the other hand if the aggregated value of that attribute counts to be less than c_i then, the constraint is not violated and the value of x_i becomes 0.

Now the total fitness function for the composite service can be given by the following equation.

$$f(X) = \left(\frac{w_1 * q_1(X) + w_2 * q_2 * (X)}{w_3 * q_3(X) + w_4 * q_4 * (X)}\right) + \mathcal{P}(X) \qquad (7)$$

Where, X is the genome for which the fitness value is being calculated. P(X) is the total penalty, calculated by equation 6. w_i the weight for i^{th} QoS attributed defined by the user. $q_i(X)$, is the aggregated attribute value for i^{th} QoS attributed. By adding the penalty value in the utility function the function becomes balanced to move towards the constraint satisfaction over the course of evolution.

5.2.4 Algorithm Explanation

First of all initial population of size 100 is generated by selecting random values in the range 1 to no of concrete web services available for each service class. Each solution in this population represents a genome as shown in Figure 9. After that cost for all the solutions is calculated and top 3 solutions are assigned to α (alpha), β (beta) and δ (delta) respectively. Now in each iteration of the 100 iterations following happens:

- The value of **a** is calculated by the formula **a = 2-2*t/t_Max** [8], which reduces from 2 to 0. In our approach we have taken t_Max=100.
- Now the values of **A** and **C** are calculated by the formula **C = 2*r2** and **A = 2a*r1-a**, where r_1 and r2 are random numbers in the range 0 to 1.
- Now equation 1 and 2 is applied on each solution of the initial population to calculate the values of D and X for all 3 solutions to calculate the value of X_1, X_2 and X_3 as shown in equation 3 and 4.
- Initial population is updated by equation 5 which takes average of X_1, X_2 and X_3, calculated in previous step.
- The crossover is performed on the updated population on each two successive solutions, with crossover probability equal to 1.
- After crossover, the cost of the new obtained population is calculated.

- Top 3 solutions are found out to replace the values of alpha (α), beta (β) and delta (δ).

The process repeats for all 100 iterations. The above algorithm except the crossover function represents the GWO algorithm. Adding crossover to the GWO algorithm, adds diversity of the solutions, by creating a good mix of the solutions. In this way the MGWO algorithm picks the positive sides of GWO and GA algorithm and given better performance.

6. EXPERIMENTAL RESULTS AND DISCUSSION

In order to perform optimal web service selection with global constraints, we use MGWO algorithm. Further, to judge the relative performance of the MGWO, it is compared against two well-known state-of-the-art meta-heuristic techniques: GA and GWO. It is noteworthy that GA is already applied for web service selection [14].

6.1 Parameter Setting

We used MATLAB 2015(b) to implement all the experiments in the window environment on a 64 bit 1.70 GHz Intel(R) Core(TM) i3-4005U PC with 4 GB RAM. In order to make a fair comparison, following parameter setting is adopted for the algorithms: We select maximum number of generations as a stopping criterion, which is set to 100 for all three algorithms. And total 20 runs are carried for each experiment. However, the parameter for GA is kept as in [14].

6.2 Data set

We used the data set from repository[2] in this work. A brief description of the data set is given as follows: The dataset provides values of 9 QoS attributes for 2507 concrete web services. We have used data for 2500 web services, in order to make the problem symmetric. We have taken 10 service classes in our composite web service example and assigned 250 web services from this dataset to each service class. All 9 QoS attributes are considered in our experimental study. Two out of the 9 QoS attributes are negative QoS attributes, whereas rest of the 7 QoS attributes are positive QoS attributes. The values given in this dataset are in different units, for example availability and reliability is given in percentage, and response time and reliability is given in seconds.

6.3 Discussion on the result

Table I shows the average value of the results obtained in 20 runs of GA, GWO and MGWO algorithm. The results obtained clearly express that the MGWO give comparatively better than other two algorithms. Figure 10 demonstrates the convergence rate of the MGWO, the GWO,

and the GA. The successive generations are represented in x-axes and the average of best fitness values obtained in 20 different runs is represented in y-axes. It is clear from Figure 10 that convergence speed of the MGWO is far better than GA and comparable to the GWO. Further, the average values of all nine service attributes achieved by three algorithms in successive generation are depicted in Figure 11. It is interesting to see that the MGWO gives better performance in almost all the cases. For some attributes GWO is better, because it is not necessary for values of all the attributes to be best in the best combination. The ultimate result of the MGWO algorithm is best, with best results for most of the attributes, if we want some specific attribute to get minimum value, we can increase it's weight.

	GA	**GWO**	**MGWO**
Response Time	1935.21	1435.55	1438.46
Availability	856.421	865.4	886.35
Throughput	123.895	129.505	135.965
Successability	885.316	899	919.4
Reliability	722.684	718.3	725.9
Compliance	905.053	912	907.6
Best Practices	813.211	809.3	818.8
Latency	176.768	110.95	112.485
Documentation	390.421	399	394.85
Total Fitness	0.0077	0.00515	0.00507

Table I: Average attributes and fitness value for 20 runs

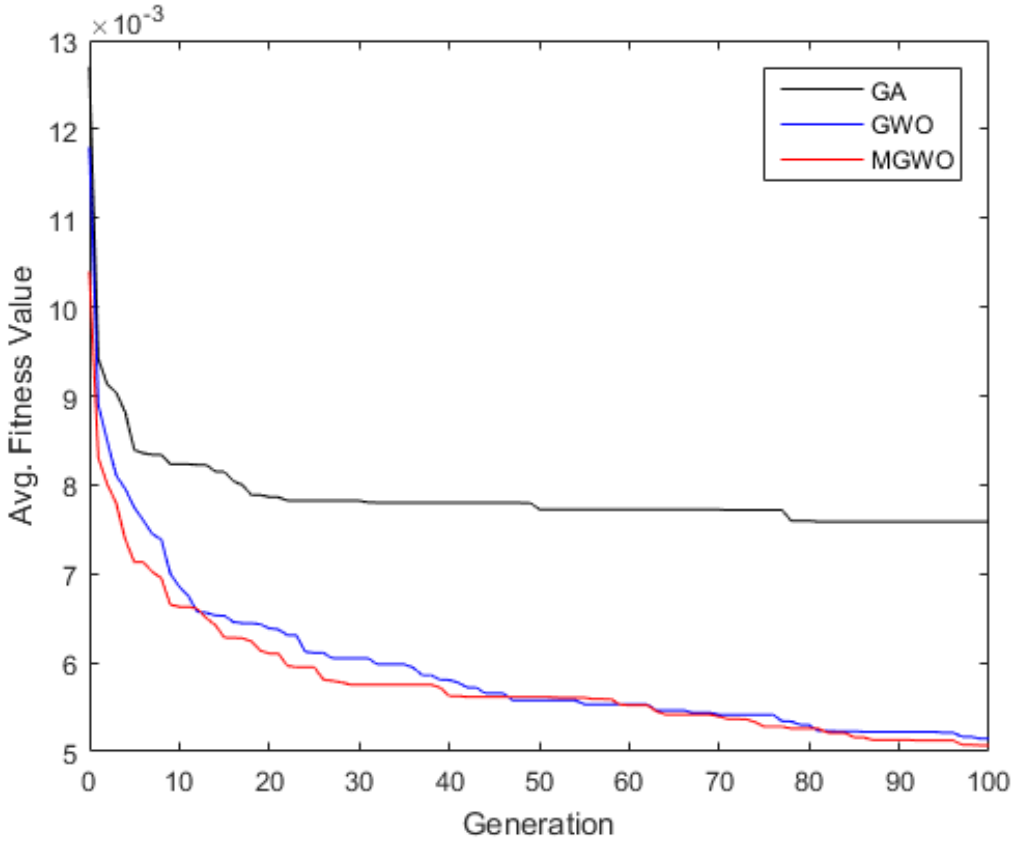
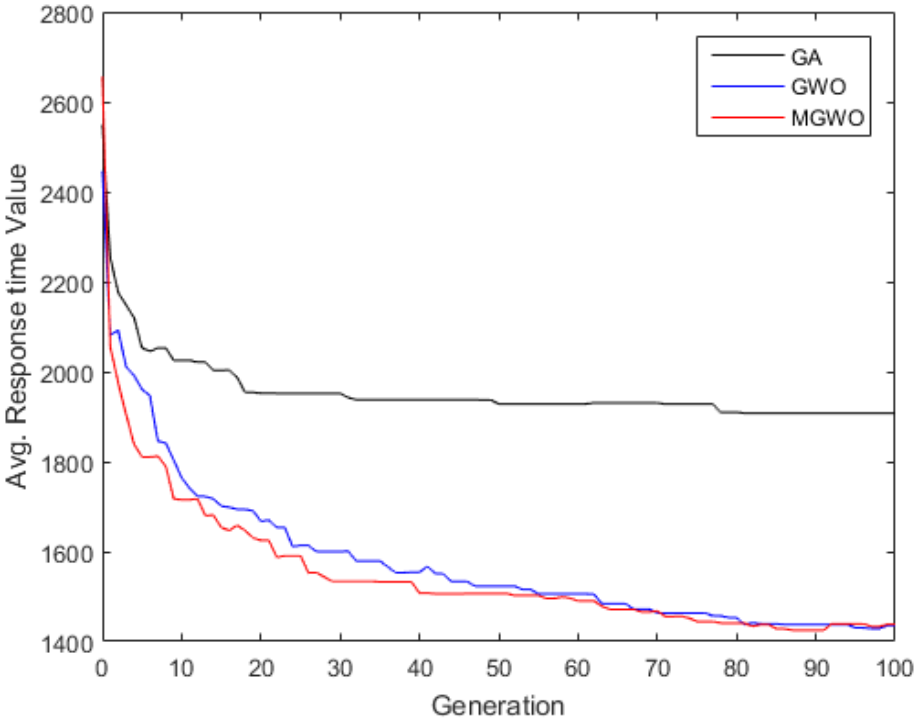
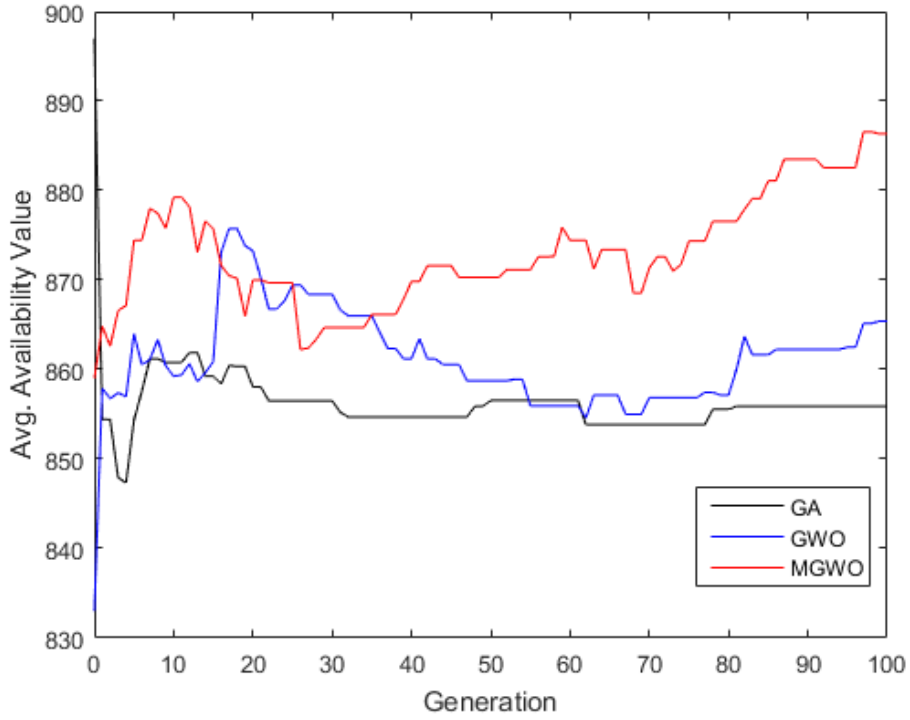


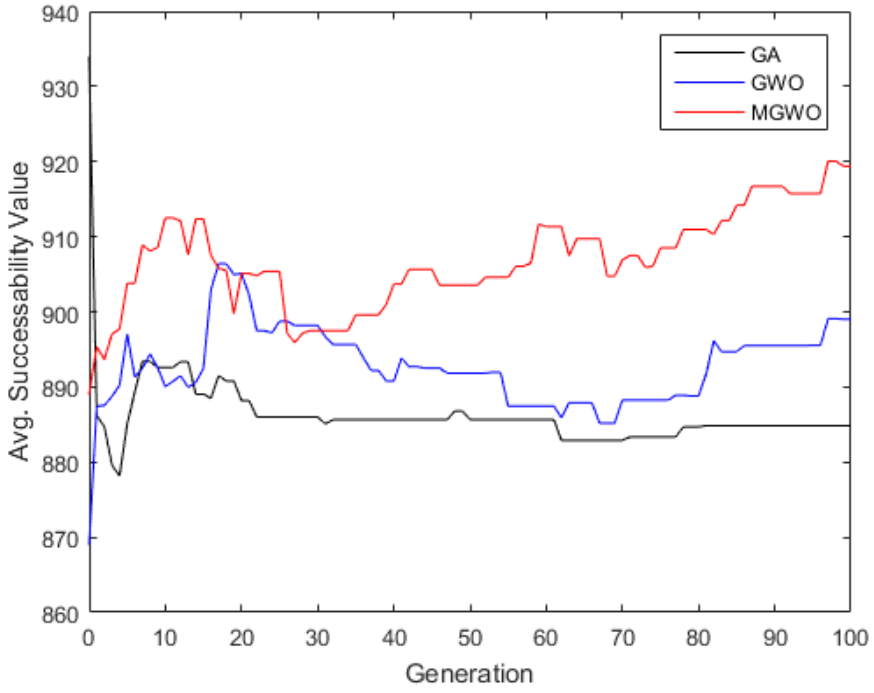
Figure 10 : Comparison of Avg. fitness values from GA,GWO and MGWO


GA
GWO
MGWO
Avg. Response time Value
Generation
2800
2600
2400
2200
2000
1800
1600
1400
0
10
20
30
40
50
60
70
80
90
100

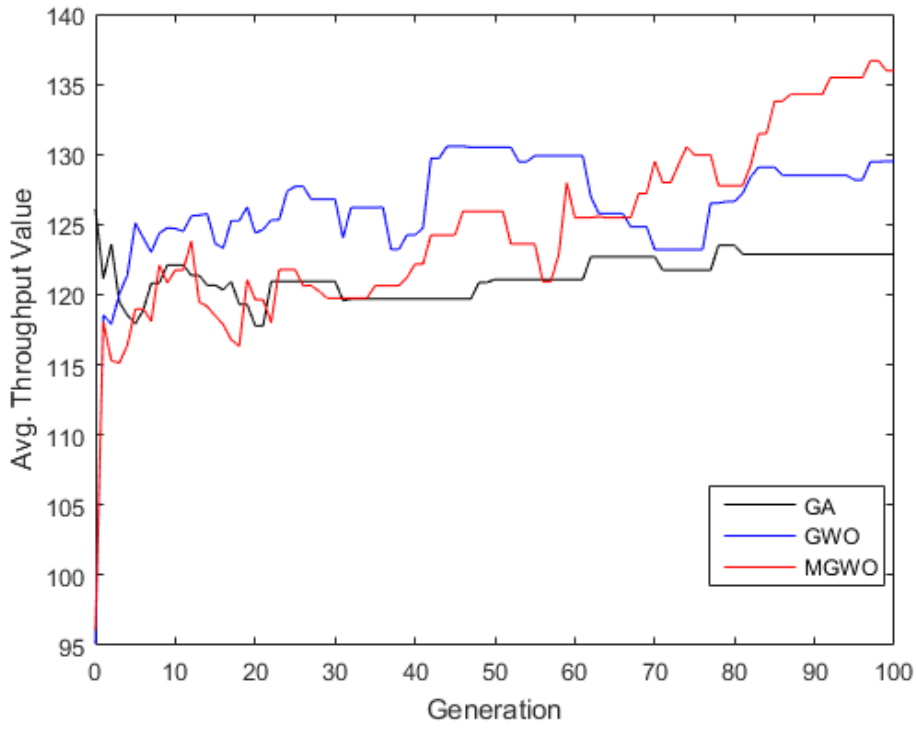
(a)


GA
GWO
MGWO
Avg. Availability Value
Generation
900
890
880
870
860
850
840
830
0
10
20
30
40
50
60
70
80
90
100

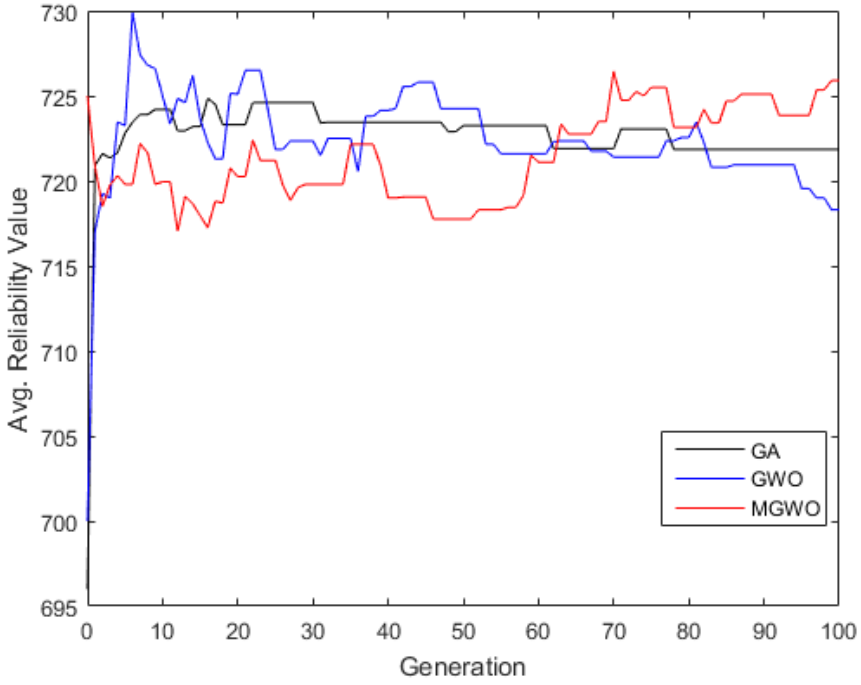
(b)


GA
GWO
MGWO
Avg. Successability Value
Generation
940
930
920
910
900
890
880
870
860
0
10
20
30
40
50
60
70
80
90
100

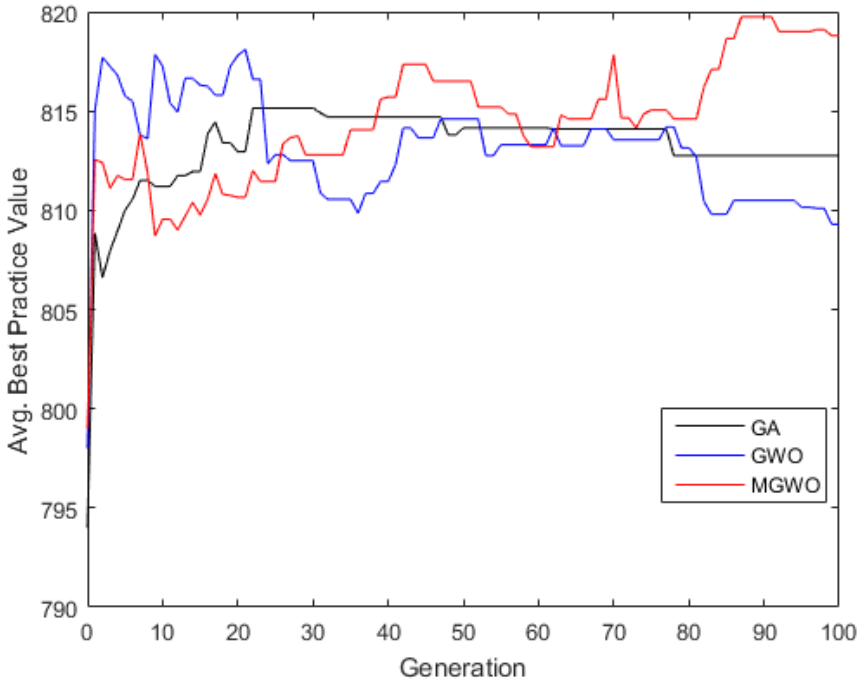
(c)


GA
GWO
MGWO
Avg. Throughput Value
Generation
140
135
130
125
120
115
110
105
100
95
0
10
20
30
40
50
60
70
80
90
100

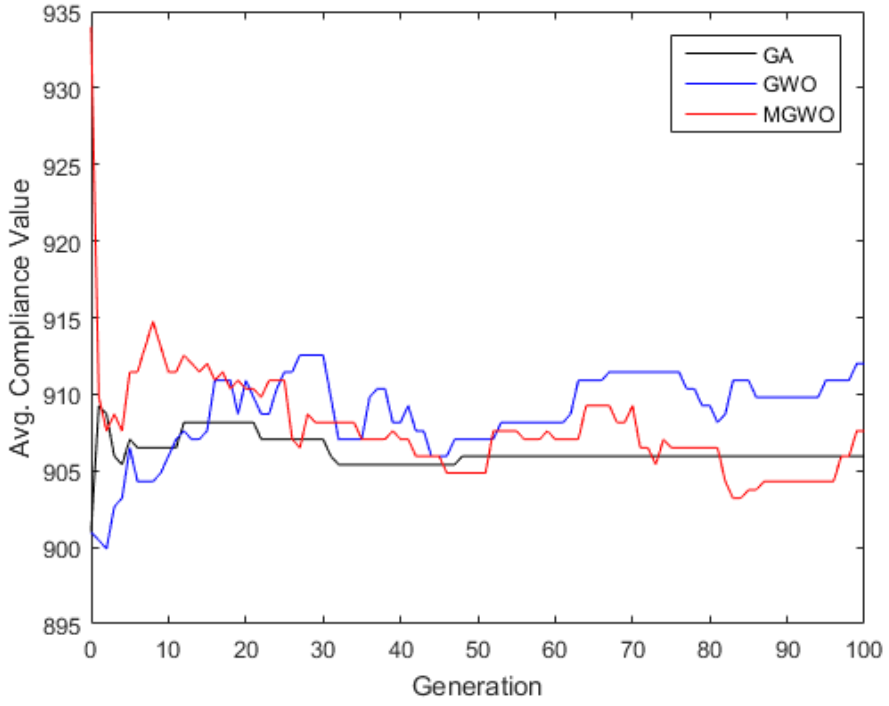
(d)


730
725
720
715
710
705
700
695
Avg. Reliability Value
0
10
20
30
40
50
60
70
80
90
100
Generation
GA
GWO
MGWO

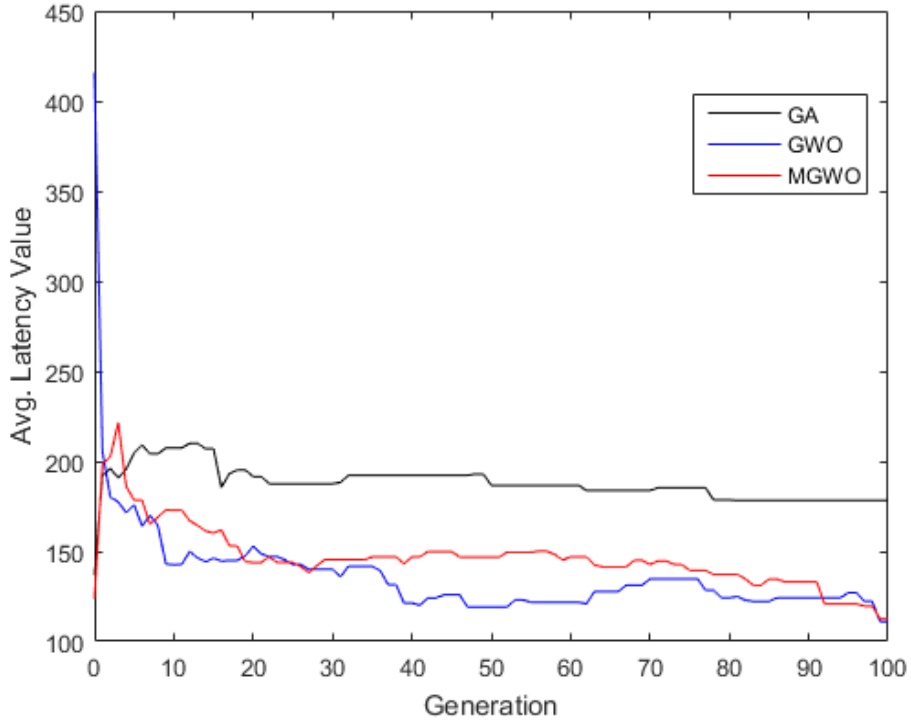
(e)


820
815
810
805
800
795
790
Avg. Best Practice Value
0
10
20
30
40
50
60
70
80
90
100
Generation
GA
GWO
MGWO

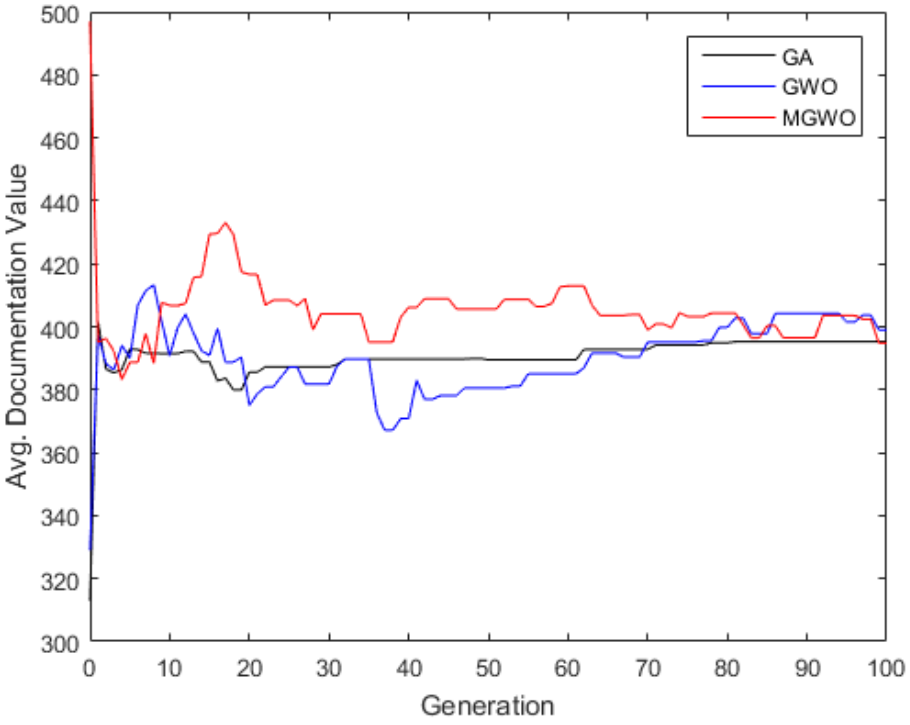
(f)


GA
GWO
MGWO
Avg. Compliance Value
Generation
935
930
925
920
915
910
905
900
895
0
10
20
30
40
50
60
70
80
90
100

(g)


GA
GWO
MGWO
Avg. Latency Value
Generation
450
400
350
300
250
200
150
100
0
10
20
30
40
50
60
70
80
90
100

(h)



(i)

Figure 11 : Evolution of QoS attributes over the course of generations

7. CONCLUSION

In this work, we applied Modified Gray Wolf Optimizer algorithm for the problem of web service selection with global constraints. The MGWO is able to balance exploration and exploitation in the search space. Our proposed approach determines a set of services that satisfy the constraints and optimize the QoS attributes. To validate the performance of the MGWO, in terms of optimality and convergence rate, it is compared against two state-of-the-art algorithms: GA and GWO on benchmark test data set from public repository. Finally, the results reveal that the MGWO performs better than its two contestants, achieving lower fitness value and showing faster convergence rate.

Furthermore, in this work we assumed the sequential workflow of composition. In future this work can be extended for web service selection in non-sequential workflows.

REFERENCES

[1] Dervis Karaboga and Celal Ozturk. A novel clustering approach: Artificial bee colony (abc) algorithm. Applied soft computing, 11(1):652–657, 2011.

[2] Mohd Herwan Sulaiman, Zuriani Mustaffa, Mohd Rusllim Mohamed, and Omar Aliman. Using the gray wolf optimizer for solving optimal reactive power dispatch problem. Applied Soft Computing, 32:286–292, 2015.

[3] Kusum Kumari Bharti and Pramod Kumar Singh. Opposition chaotic fitness mutation based adaptive inertia based bpso for feature selection in text clustering. Applied Soft Computing, 43:20–34, 2016.

[4] David E Goldberg et al. Genetic algorithms in search optimization and machine learning, volume 412. Addison-wesley Reading Menlo Park, 1989.

[5] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4):341–359, 1997.

[6] Russ C Eberhart, James Kennedy, et al. A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science, volume 1, pages 39–43. New York, NY, 1995.

[7] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[8] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. Advances in Engineering Software, 69:46–61, 2014.

[9] Avadh Kishor and Pramod Kumar Singh. Empirical study of grey wolf optimizer. In Proceedings of Fifth International Conference on Soft Computing for Problem Solving, pages 1037–1049. Springer, 2016.

[10] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z Sheng. Quality driven web services composition. In Proceedings of the 12th international conference on World Wide Web, pages 411–421. ACM, 2003.

[11] Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. ACM Transactions on the Web (TWEB), 1(1):6, 2007.

[12] Mohammad Alrifai and Thomas Risse. Combining global optimization with local selection for efficient qos-aware service composition. In Proceedings of the 18th international conference on World wide web, pages 881–890. ACM, 2009.

[13] Mohammad Alrifai, Thomas Risse, and Wolfgang Nejdl. A hybrid approach for efficient web service composition with end-to-end qos constraints. ACM Transactions on the Web (TWEB), 6(2):7, 2012.

[14] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for qos-aware service composition based on genetic algorithms. In Proceedings of the 7th annual conference on Genetic and evolutionary computation, pages 1069–1075. ACM, 2005.

[15] Zhi-peng Gao, Chen Jian, Xue-song Qiu, and Luo-ming Meng. Qoe/qos driven simulated annealing-based genetic algorithm for web services selection. The Journal of China Universities of Posts and Telecommunications, 16:102–107, 2009.

[16] Jos´e Antonio Parejo, Sergio Segura, Pablo Fernandez, and Antonio Ruiz- Cort´es. Qos-aware web services composition using grasp with path relinking. Expert Systems with Applications, 41(9):4211–4223, 2014.

[17] W3C Working Group. Web services architecture. *http://www.w3.org/*.

[18] Ma, Yue, and Chengwen Zhang. "Quick convergence of genetic algorithm for QoS-driven web service selection." *Computer Networks* 52.5 (2008): 1093-1104.