An Efficient Discrete Invasive Weed Optimization Algorithm for Web Services Selection

Kai Su, Liangli Ma, Xiaoming Guo, Yufei Sun
Department of Computer Engineering, Naval University of Engineering, Wuhan 430033, China
Email: keppelsue@163.com

***Abstract*—Efficient algorithm is required to select component services with end-to-end QoS constraints in dynamic composition of Web Services while optimizing the QoS of the composite service. In this paper, the services selection probelm is modeled as a nonlinear optimization with constraints, then a novel discrete invasive weed optimization web services selection algorithm is proposed. The proposed solution consists of two steps：first, a set of randomly generated feasible solutions are transformed into decimal code. Second, We utilize Gaussian diffusion to guide the population to spread in the solution space. The mutation probability and mutation step size of individual is dynamically adjusted by changing the stantard deviation of Gaussian distribution. Accordingly, the population diversity is ensured in the early stage to expand the search space, while the local search nearby excellent individuals is focused in the latter stage to ensure the global convergence. Theoretical analysis and experiment results indicate the efficiency, robustness and feasibility of our approach.**

***Index Terms*—Web Services, Service Composition, Service Selection, End-to-end QoS, Discrete Invasive Weed Optimization**

I. INTRODUCTION

The emergence of service-oriented computing paradigm and web services technologies provide a promising solution for the integration of business application within and across organization boundaries. Component services can be dynamically composed to form complex and value-added services so that the requirement of users be satisfied. With the growing number of Web Services that provide similar functionality but differ in QoS level, the dynamic service composition becomes a decision problem on which component services should be selected so that the end-to-end QoS constraints are satisfied.

Local selection strategy in which the candidate service who has the maximum QoS utility value is selected for each service class is very efficient but can not handle end-to-end QoS constraints(e.g., the constraints of overall execution time)[1]. In [2,3], the authors use integer linear programming techniques to find the optimal selection of component services for the composition. However, deterministic algorithms like linear programming are only applicable to small-scale problems due to their exponential time complexity.

In recent years, approaches based on genetic algorithm are proposed to solve the service selection problem[4,5,6,7,8]. These approaches are very efficient but too sensitive to the parameters setting and they may easily fall into premature convergence. Mehrabian and Lucas proposes the invasive weed optimization (IWO)[9], a novel heuristic intelligent algorithm, which is inspired from weed colonization. IWO has found successful applications in many pratical problems like the design and configuration of smart antennas[10,11], base station location planning for mobile communication system[12], design of encoding sequences for DNA computing[13], and the training of feed-forward artificial neural networks[14]. IWO mimics the colonizing behaviors of weeds like seed generating, reproduction, spatial dispersal and competitive exclusion. IWO showed strong robustness, adaptability and global convergence. To our best knowledge, the IWO algorithm has not been applied to web services selection problem yet.

Traditional IWO algorithm can only handle continuous space optimization problem. We proposes a discrete invasive weed optimization algorithm to solve the web services selection problem. In our approach, the problem is modeled as a nonlinear optimization problem under several constraints. First, a set of feasible solutions is randomly generated as the initial population before decimal encoding. Then, the fitness of the individuals in the population is evaluated to determine the reproductive number for each individual. Individuals with higher fitness would have stronger ability to reproduce. Finally, the individuals are randomly scattered with a Gaussian distribution in the solution space.

The rest of this paper is organized as follows. In section II, the basic concept of service composition, QoS model and its evaluation methods and QoS aware service selection model are introduced. Our approach for service selection is presented in section III. Section IV analyses the theoretical performance of our approach. Experiments evaluations are presented in section V. Finally, section 6 gives the conclusions and an outlook on future work.

II. SYSTEM MODEL

A. Concepts Definition

doi:10.4304/jsw.9.3.709-715

Service Class: Service Class $S_j = \{s_{1j}, s_{2j}, ..., s_{mj}\}$ is used to describe a set of functionality equivalent web services, where $s_{ij}(1 \le i \le m)$ represents the i th component service in service class S_j.

Abstract Composite Service: An abstract composite service $CS_{abstract} = \{S_1, S_2, \ldots, S_n\}$ can be defined as an abstract representation of a composition request. Each task of the abstract composite service refers to a service class without referring to any concrete service.

Concrete composite service: A concrete composite service CS is a instantiation of an abstract composite service. It can be gained by binding each service class S_j in $CS_{abstract}$ to a concrete service s_{ij}.

B. Qos Criteria

QoS criteria, which can be used to describe the quality of web services, is a set of non-functional properties. QoS criteria can be devided into several aspects such as runtime-related QoS, transaction support related QoS, configuration management QoS, cost related QoS and security related QoS[15]. More generally, QoS criteria can include generic QoS like response time, price, reliability, availability etc, as well as domain specific QoS[16]. In[16], the authors presented a extensible QoS computation model includes both the generic and domain specific criteria, where new criteria can be added and used to evaluate the QoS without changing the underlying model. The QoS criteria of a concrete service s can be defined as a vector $Q(s) = [q_1(s), q_2(s), \ldots, q_r(s)]$, where $q_k(s)(1 \le k \le r)$ is the k th QoS criterion of s. The QoS criteria of a concrete composite service CS can be defined as a vector $Q(CS) = [q'_1(CS), q'_2(CS), \ldots, q'_r(CS)]$, $q_k'(CS)(1 \le k \le r)$ is the k th QoS criterion of CS.

The QoS of a concrete composite service is decided by the QoS of its component services and the composition model used(e.g., sequential, parallel, condition, and loops). In this paper, we focus on the sequential composition model, due to the other models can be transformed to the sequential model by using some technologies like Critical Path Algorithm, Hot Path, and Loop Unfolding[2]. Table I shows the computation expression of the QoS of composite services. In this paper, only response time, reliability, availability and throughput are considered.

TABLE I.
COMPUTATION EXPRESSION OF THE QOS OF COMPOSITE SERVICES

QoS Criteria	Aggregation Function
Response Time	$q_{time}'(CS) = \sum_{j=1}^{n} q_{time}(s_j)$
Reliability	$q_{rel}'(CS) = \prod_{j=1}^{n} q_{rel}(s_j)$
Availability	$q_{av}'(CS) = \prod_{j=1}^{n} q_{av}(s_j)$
Throughput	$q_{thr}'(CS) = \min_{1 \le j \le n}\{q_{thr}(s_j)\}$

Criteria like reliability, availability and throughput are positive that means the higher the value is, the higher the quality is. On the contrary, response time is a negative criterion that means the higher the value is, the lower the quality is. We utilize the simple addtive weighting(SAW)[2] technique to map the QoS vector into a single real value for the purpose of evaluating the quality of composite services. The utility of composite service CS is computed as

$$U(CS) = \sum_{k=1}^{1} w_k \bullet \frac{Q_{\max}(k) - q_k'(CS)}{Q_{\max}(k) - Q_{\min}(k)} + \sum_{k=2}^{4} w_k \bullet \frac{q_k'(CS) - Q_{\min}(k)}{Q_{\max}(k) - Q_{\min}(k)} \quad (1)$$

$Q_{\min}(k)$ and $Q_{\max}(k)$ represent the minimum and maximum value of the k th QoS criterion of all possible instantiations of the given abstract composite service. $Q_{\min}(k)$ and $Q_{\max}(k)$ can be computed as follows:

$$Q_{\min}(k) = \sum_{j=1}^{n} \underset{s_{ij} \in S_j}{\text{Min}}\, q_k(s_{ij}) \quad (2)$$

$$Q_{\max}(k) = \sum_{j=1}^{n} \underset{s_{ij} \in S_j}{\text{Max}}\, q_k(s_{ij}) \quad (3)$$

and w_k $(1 \le k \le 4)$ is the weight assigned to each QoS criterion, which represents user's priorities. we have the following constraints on w_k.

$$\sum_{k=1}^{4} w_k = 1, 0 \le w_k \le 1, w_k \in R \quad (4)$$

C. Qos-aware Service Composition

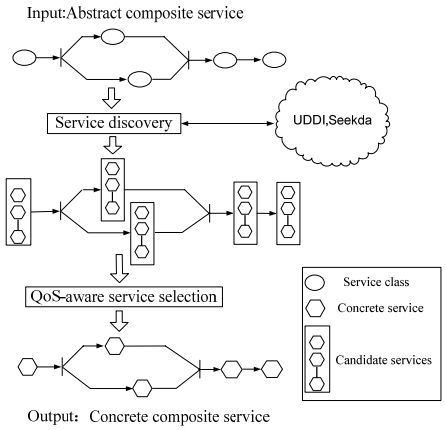

Figure 1. QoS aware Web Service Composition

The process of QoS-aware service composition can be defined as follows[17]. First, an abstract composite service can be stated in a workflow-like language like BPEL4WS. Second, the discovery engine utilizes existing infrastructure like UDDI or Seekda to discover several concrete services for each task in the workflow by means of syntactic or semantic matching. Finally, the service selection middleware selects one component service for each task. The component services selected should maximize the utility of composite service without violating the end-to-end QoS constraints. Figure 1 depicts the process of QoS-aware service

composition.

We use binary decision variables x_{ij} to represent whether the candidate service s_{ij} is selected or not ($x_{ij}=1$ means s_{ij} is selected, $x_{ij}=0$ means s_{ij} is not selected). By re-writing (1) to include the decision variables and the QoS computation expression of composite services defined in table (1), the problem of QoS-aware service selection can be formulated as a maximization problem of the utility value given by

$$F=\mathrm{Max}\left(w_1\bullet\frac{Q_{\max}(1)-\sum_{j=1}^{n}\sum_{i=1}^{m}q_1(s_{ij})\bullet x_{ij}}{Q_{\max}(1)-Q_{\min}(1)}+\sum_{k=2}^{3}w_k\bullet\frac{\prod_{j=1}^{n}\sum_{i=1}^{m}q_k(s_{ij})\bullet x_{ij}-Q_{\min}(k)}{Q_{\max}(k)-Q_{\min}(k)}\right.$$

$$\left.+w_4\bullet\frac{\mathrm{Min}_j(\sum_{i=1}^{m}q_4(s_{ij})\bullet x_{ij})-Q_{\min}(4)}{Q_{\max}(4)-Q_{\min}(4)}\right) \quad (5)$$

Subject to the following constraints:

$$\sum_{i=1}^{m}x_{ij}=1, 1\le j\le n \quad (6)$$

$$\sum_{j=1}^{n}\sum_{i=1}^{m}q_1(s_{ij})\bullet x_{ij}\le C_1 \quad (7)$$

$$\prod_{j=1}^{n}\sum_{i=1}^{m}q_k(s_{ij})\bullet x_{ij}\le C_k, k=2,3 \quad (8)$$

$$\mathrm{Min}_j\left(\sum_{i=1}^{m}q_4(s_{ij})\bullet x_{ij}\right)\le C_4 \quad (9)$$

Where C_1, C_2, C_3, C_4 represent the end-to-end constraints on response time, reliability, availability and throughput respectively. (6) means that only one concrete service should be selected for each task. From the above discussion, we can find that QoS-aware service selection problem is a nonlinear optimization problem under several constraints. Our approach for this problem is presented in the next section.

III. DIWOWS ALGORITHM

QoS-aware service selection problem is proved to be a NP-Complete problem. Deterministic algorithms like linear programming are only applicable to small-scale problems due to their exponential time complexity. Genetic algorithm(GA) is considered as an effcient strategy to solve the service selection problem. However, GA may easily fall into premature convergence. In this section we propose a discrete invasive weed optimization algorithm for web service selection(DIWOWS). DIWOWS can obtain a approximate optimal solution within an acceptable period.

Weeds indicates the vigorous and invasive plants in nature which can threat the growth of other plants. Weeds can maintain the vitality through improving their environmental fitness. The main idea of DIWOWS can be described as follows: first, a set of feasible solutions is generated, these solutions will be encoded into decimal code groups which represent the weed individuals. Second, the fitness of the individuals in the population are evaluated to determine the seeds number for each individual. The seeds will be randomly distributed over the search space in normal manner. The seeds who satisfy the constraints and their parents weeds will together constitute the next generation population. When the maximum number of population $P_{\max}$ in a colony is reached, new seeds with their parents are ranked together with respect to fitness, weeds with lower fitness are eliminated to attain the allowable population size. The whole process continues until the maximum number of iterations is reached and hopefully the weed with the best fitness is the closest to the optimal solution. The DIWOWS algorithm has the following steps:

Algorithm 1. DIWOWS.

Input: maximum population size $P_{\max}$, maximum number of iterations $iter_{\max}$.

Output: solutions set P.

Begin

1 $iter \leftarrow 0, P \leftarrow \varnothing$
2 Initialize
3 **While**($iter \le iter_{\max}$) **do**
4 **For** each individual $P_i \in P$
5 $f_i \leftarrow U(P_i)$
6 $W_i \leftarrow SeedNumCal(P_i, f_i)$
7 $j \leftarrow 0, Temp \leftarrow \varnothing, \hat{P}_i \leftarrow 0$
8 **While**($j < W_i$) **do**
9 $\hat{P}_i \leftarrow$ SeedGenerator(P_i)
10 **If**($Constr(\hat{P}_i)$)**then**
11 $Temp \leftarrow Temp \cup \hat{P}_i$
12 **Else** Goto(9)
13 **End If**
14 $j \leftarrow j+1$
15 $P \leftarrow P \cup Temp$
16 **End For**
17 **If (** $|P| > P_{\max}$ **) then**
18 $P \leftarrow Select(P, P_{\max})$
19 **End If**
20 $iter \leftarrow iter+1$
21 **Output** P

End

Where P_i represents a weed individual; f_i represents the fitness of P_i, which can be obtained by (1); $\hat{P}_i$ represents the seeds generated by P_i; $Constr(\bullet)$ determines whether an individual satisfy the QoS constraints; $Temp$ is a temp set used to store the new generated seeds; Step 2 generates the initial weeds population, which is described in subsection A; Step 6 evaluates the number of seeds of each individual according to the fitness, subsection *B* gives the details; Step 9 generates new seeds for the weed individual, subsection *C* gives the explicit procedure; Step 18 selects $P_{\max}$ individuals with better fitness in population

P .

A. Initialize

1) Initialize population

The initial population is generated by random means shown in algorithm 2.

Algorithm 2. InitWeedPopulation.

Input: population size L ,object function F , constraints set C

Output: initial population P .

Begin

1 $P \leftarrow \varnothing, i \leftarrow 0$

2 **While** ($i < L$)

3 $rand(P_i)$

4 **If** ($Constr(P_i)$)**then**

5 $P \leftarrow P \cup P_i$

6 **Else** Goto(3)

7 **End If**

8 $i \leftarrow i+1$

9 **Output** P

End

Where step 3 represents generating one individual randomly and step 4 determines whether the individual satisfy the constraints.

2) Chromosome coding

We assume that the given abstract composite service $CS_{abstract} = \{S_1, S_2, \ldots, S_N\}$ has N tasks and each task has M available candidate services. Decimal variables x_j can be used to represent the participation of service $s_{x_{j,j}}$ (i.e., $\{ x_1 = 2, x_2 = 5, x_3 = 1, x_4 = 4, x_5 = 8 \}$ indicates that service $\{ s_{2,1}, s_{5,2}, s_{1,3}, s_{4,4}, s_{8,5} \}$ are selected to participate in the composite service). Consequently, the solutions of service selection problem can be represented by chromosomes $x_1 x_2 x_3 \ldots x_N$ ($1 \le x_j \le M$, where $1 \le j \le N$). The chromosomes that satisfy the constraints defined by formula (6)(7)(8)(9) are valid chromosomes, others are invalid chromosomes.

B . Reproduction

Each individual will produce new seeds depending on their fitness. Individuals with higher fitness produces more seeds[9]. The number of seeds is computed as

$$W_i = \left\lfloor \frac{f_i - f_{\min}}{f_{\max} - f_{\min}} \bullet (W_{\max} - W_{\min}) + W_{\min} \right\rfloor$$

$$\left(\text{if } f_{\max} = f_{\min}, \text{then } W_i = W_{\max}\right) \qquad (10)$$

W_i represents the number of seeds produced by the i th individual in current population; f_i represents the fitness of the i th individual; $f_{\max}$ and $f_{\min}$ are the maximum and minimum fitness of current population; $W_{\max}$ and $W_{\min}$ are the maximum and minimum number of seeds that can be produced by a single individual, both of them are predefined parameters of the algorithm and can be adjusted according to the practical problems. Figure 2 illustrates the calculation of seeds number of individuals. The figure shows that the seeds number of individual increases linearly with the fitness.

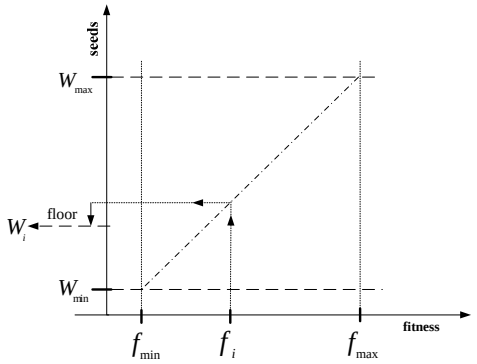

Figure 2. Calculation of Seeds Number

C. Spatial Dispersal

The produced seeds are being randomly dispread over the N-dimensional search space by normally distributed random numbers with a mean equal to zero and a varying standard deviation which decreases from $\sigma_{initial}$ to σ_{final} with the iteration. The standard deviation is computed as[9]

$$\sigma_{iter} = \left(\frac{iter_{\max} - iter}{iter_{\max}} \right)^n \bullet (\sigma_{initial} - \sigma_{final}) + \sigma_{final} \qquad (11)$$

Where σ_{iter} is the standard deviation at the present step; $iter_{\max}$ is the maximum number of iterations; $\sigma_{initial}$ and σ_{final} are the predefined initial and final value of standard deviation, they are set to $M/2$ and 1 respectively in DIWOWS. That is to say, the higher M is, the larger the initial value of standard deviation is. n is a nonlinear modulation index which can be set to 3 experientially.

In traditional IWO algorithm, the seeds are produced by adding *N*-dimensional normally distributed random diffusion value to the parent individual. In DIWOWS, the N-dimensional random diffusion value which satisfy $N(0, \sigma_{iter}^2)$ are mapped to the mutation probability of corresponding symbol. The seeds are produced by mutating some symbols of the individual according to the mutation probablity. The mutation step size reduces with the decreasing of the standard deviation. We suppose that the diffusion value of the j th symbol is equal to d_j^{iter} in the $iter$ th iteration. Then, the diffusion value can be mapped to the mutation probability of this symbol by formula (12). Figure 3 demonstrates the mapping from symbol diffusion value to mutation probability.

$$pr_j = \frac{1}{1 + e^{-d_j^{iter} + \frac{M}{2}}} + \frac{1}{1 + e^{d_j^{iter} + \frac{M}{2}}} \qquad (12)$$

Where pr_j represents the mutation probability of the j th symbol of weed individual; d_j^{iter} , which is generated by the normal distribution $N(0, \sigma_{iter}^2)$, represents the diffusion value of the j th symbol; M represents the maximum value of the symbol.

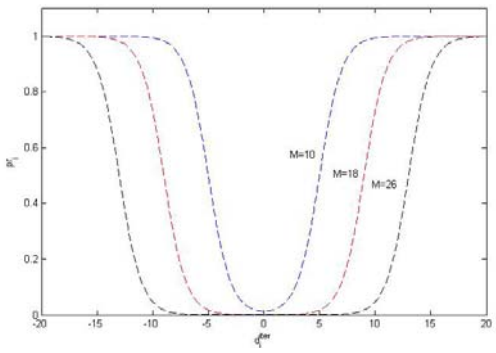

Figure3. Mapping of Symbol Diffusion Value to Mutation Probability

In the initial phase, the standard deviation σ_{iter} should take a larger number, so that the random diffusion value d_j^{iter} will be widely distributed. In this case, both of the symbol mutation probability pr_j and the mutation step size would be larger to strengthen the global search for the solution space. In the anaphase, the standard deviation σ_{iter} should take a smaller number, so that the random diffusion value d_j^{iter} will be densely distributed. In this case, both of the symbol mutation probability and the mutation step size would be smaller to strengthen the local search nearby excellent individuals. Figure 3 also demonstrates that the smaller M is, the narrower the curve will be. This is because when M is small, the range of diffusion value will be small, then the curve should correspondingly be compressed so that more data can be mapped to the high mutation probability domain thereby preventing local convergence. On the contrary, when M is large, the range of diffusion value will be large, then the curve should be correspondingly broadened so that the data which is mapped to the high mutation probability domain will be reduced, thereby preventing the search process be too random.

After getting the mutation probability of each symbol, a random number satisfies [0,1] uniform distribution will be generated to compare to the mutation probability. The symbol whose mutation probability is larger than the random number will be mutated. Each mutated symbol of the individual can be added to a random number which satisfies $N(0,\sigma_{iter}^{\ 2})$ to generate a new seed. The symbols in the new seed that are smaller than 0 and larger than M should be set to 0 and M, and the symbols that are not interger should be rounded to zero. Algorithm 3 shows how the new seeds be produced.

Algorithm 3. SeedGenerator.
Input: P_i, $iter$, M.
Output: $\hat{P}_i$.
Begin
1 $j \leftarrow 0,\ d \leftarrow 0$
2 $\sigma_{iter} \leftarrow SdCal(iter)$
3 **While**($j < N$)
4 $d_j \leftarrow normrnd(0,\sigma_{iter})$
5 $pr_j \leftarrow Map(d_j, M)$
6 **If**($pr_j \le rand[0,1]$)**then**
7 $d_j \leftarrow 0$
8 **Else** $d_j \leftarrow fix(normrnd(0,\sigma_{iter}))$
9 **End If**
10 $\hat{P}_i^j \leftarrow P_i^j + d_j$
11 **If**($\hat{P}_i^j$ <0)**then**
12 $\hat{P}_i^j \leftarrow 0$
13 **Else If** ($\hat{P}_i^j > M$)**then**
14 $\hat{P}_i^j \leftarrow M$
15 **End If**
16 $j \leftarrow j+1$
17 Output $\hat{P}_i$
End

Where P_i is a weed individual; $\hat{P}_i$ is a seed of P_i; d is the diffusion value of the weed individual; d_j represents the j th element of d; P_i^j represents the j th symbol of P_i; $\hat{P}_i^j$ represents the j th symbol of $\hat{P}_i$; $SdCal(iter)$ represents calculating the standard deviation in the $iter$ th iteration by formula (11); $Map(d_j, M)$ represents calculating the symbol mutation probability mapped to d_j by formula (12); $rand[0,1]$ indicates generating a random number which satisfies [0,1] uniform distribution; $normrnd(0,\sigma_{iter})$ indicates generating a random number which satisfies $N(0,\sigma_{iter}^{\ 2})$; $fix(\bullet)$ indicates rounding operation to zero.

IV. Performance Analysis

There are three factors that determine the size of the QoS-aware service selection problem: the number of tasks in composite service N, the number of candidate services per class M and the number of QoS constraints k. The time complexity of exhaustive approach is $O(M^N)$. The time complexity of interger linear programming is $O(2^{M\cdot N})$ in the worst case[17]. Obviously, both of them are very costly for large scale problems. The Monte Carlo approach, in which M^q solutions(where q is an integer smaller than N) are sampled randomly from the search space, may obtain close-to-optimal results with high probability while reducing the time consumed. However, the computation cost of Monte Carlo is also unacceptable when the size of problem is large. The time complexity of DIWOWS includes three parts: generating new seeds, verifying the validity of seeds and selecting individuals based on fitness sorting. We propose that the maximum number of iterations is $iter_{\max}$, the maximum population size is $P_{\max}$, the maximum number of seeds is $W_{\max}$, the number of constraints is k. Then the time complexity of generating new seeds is equal to $O(iter_{\max}\bullet P_{\max}\bullet W_{\max}\bullet N)$, the time complexity of verifying the validity of seeds is equal to $O(iter_{\max}\bullet P_{\max}\bullet W_{\max}\bullet N\bullet k)$, the time complexity of selecting individuals based on fitness sorting is equal to

$O(iter_{\max} \bullet P_{\max} \bullet W_{\max} \bullet \log(P_{\max} \bullet W_{\max}))$ .Therefore, the total time complexity of DIWOWS is equal to $O(iter_{\max} \bullet P_{\max} \bullet W_{\max} \bullet (N + N \bullet k + \log(P_{\max} \bullet W_{\max})))$. In this paper, only the QoS criteria of response time, reliability, availability and throughput are considered, which means that $k = 4$. Thus, the total time complexity of DIWOWS can be simplified as $O(iter_{\max} \bullet P_{\max} \bullet W_{\max} \bullet (N + \log(P_{\max} \bullet W_{\max})))$ which is dominated by the maximum number of iterations $iter_{\max}$, the maximum population size $P_{\max}$, the maximum number of seeds $W_{\max}$ and the number of tasks in composite service N . Consequently, DIWOWS outperforms the approaches referred above in term of computation time, due to the time complexity of DIWOWS is independent on the number of candidate services M .

V. Experimental Evaluation

We conducted experiments using the QWS real dataset from [16], which includes measurements of 9 QoS attributes for 2500 real web services on the web. The experiments were conducted on a Dell inspiron 545 machine with Intel Core2 Quad 2.50GHz processors and 4 GB RAM. The machine is running under Windows XP and Matlab7.1.

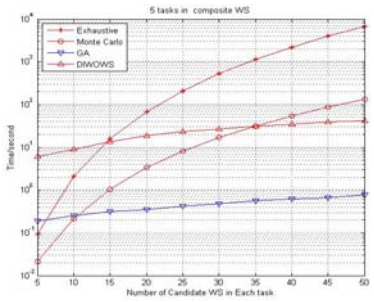

Figure 4. Performance Comparison

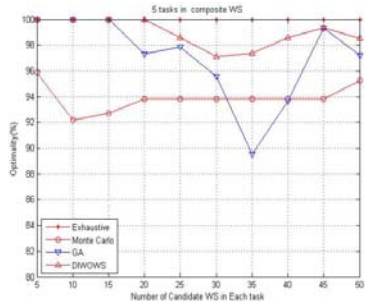

Figure 5. Optimality Comparison

In Figure 4 we compare the performance of our approach and other approaches like Exhaustive search, Monte Carlo and Genetic Algorithm. In this experiment the weights of QoS criteria W_k ($1 \le k \le 4$) are equally set to 0.25, the number of service classes N is fixed to 5, the number of candidate services per class M varies from 5 to 50. The parameter q of Monte Carlo approach is set to 4. The crossover probability, mutation probability and population size of GA are set to 0.7, 0.15 and 40 respectively. The parameters of DIWOWS like L , $P_{\max}$, $W_{\max}$, $W_{\min}$ are set to 40, 50, 5 and 1 respectively. The maximum number of iterations of GA and DIWOWS varies from 100 to 600 according to the increasing of M . Figure 4 indicates that he computation time of Exhaustive search and Monte Carlo increases exponentially with the increase of M , while DIWOWS and GA increases linearly. Figure 4 also indicates that the computation time of DIWOWS exceeds that of GA. We will discuss the trade-off between computation time and the quality of solution.

In Figure 5 we compare the optimality of DIWOWS and other approaches with different number of candidate services. Optimality can be defined as a ratio between the optimal QoS utility value obtained by the given approach and the optimal QoS utility value obtained by exhaustive approach. In this experiment, the values of the parameters are same as above. The results show that DIWOWS is able to achieve above 97% optimality, which outperforms other approaches. Therefore, DIWOWS is able to obtain a close-to-optimal solution with very low cost.

Figure 6 and Figure 7 depicts the convergence results of DIWOWS compared to GA. The parameters like crossover probability, mutation probability, population size of GA and L , $P_{\max}$, $W_{\max}$, $W_{\min}$ of DIWOWS are same as above. In Figure 6, the number of service classes N is fixed to 5, the number of candidate services per class M is fixed to 50,

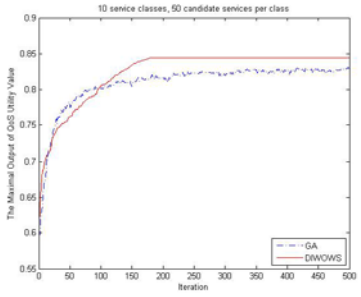

Figure 6. Convergence Comparison 1 Between DIWOWS and GA

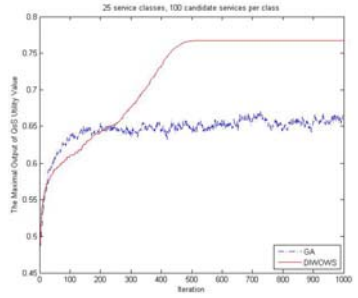

Figure 7. Convergence Comparison 2 Between DIWOWS and GA

the maximum number of iterations of GA and DIWOWS are set to 500. Figure 6 shows that both of them converge after about 200 iterations, the result of our approach slightly outperforms GA. In Figure 7, the number of service classes N is fixed to 25, the number of candidate services per class M is fixed to 100, the maximum number of iterations of both algorithms are set to 1000. Figure 7 shows that GA converges after about 200 iterations and DIWOWS converges after about 500 iterations. Although the convergence speed of the latter is lower, while the quality of the convergence result significantly outperforms the former.

The superiority could be more obvious when the scale of composition service is larger. As we know before, the computation time of DIWOWS is slightly higher than GA. Therefore, there is a trade-off between computation time and the quality of results in pratical application. When N is set to 25 and M is set to 100, the global approximate optimal solution can be obtained within about 20 seconds, which meets the demand of most tasks. Therefore, our approach can achieve an approximate optimal solution within acceptable time. In the experiment we also find the results of GA are very sensitive to the parameters setting, while the results of DIWOWS is not sensitive to the parameters setting, which indicates DIWOWS is more robust. Generally, $P_{\max}$ and $W_{\max}$ are set to 50 and 5 to meet the demands of most composition tasks.

VI. Conclusion And Future Work

QoS-aware web services selection is an important issue in service composition. In this paper, the problem is modeled as a nonlinear optimization problem under the end-to-end QoS constraints. Then we presented an efficient discrete invasive weed optimization algorithm to solve this problem. Theoretical analysis and experiment results indicate the efficiency, robustness and feasibility of our approach. In the current approach the QoS dataset is assumed to be static. In the future work, We aim at developing algorithms under dynamic QoS dataset.

Acknowledgment

This work was supported by the pre-research funds from PLA General Armament Department of China (No. 9140A27040413JB11407)

References

[1] L. Zeng, B. Benatallah, "QoS-Aware Middleware for Web Services Composition," IEEE Transactions on software engineering, vol. 30, no. 5, pp. 321–322, 2004.

[2] L. Zeng, B. Benatallah, "Quality Driven Web Services Composition," In Proceedings of the International World Wide Web Conference, 2003, pp. 411–421.

[3] D. Ardagna, B. Pernici, "Adaptive Service Composition in Flexible Processes," IEEE Transactions on software engineering, vol. 33, no. 6, pp. 373–376, 2007.

[4] Canfora G, Dipenta M, Esposito R, et al, "A lightweight approach for QoS-aware service composition," In Proc. of the 2nd International Conference on Service Oriented Computing. New York, 2004, pp. 232–239.

[5] Canfora G, Dipenta M, Esposito R, "An approach for QoS-aware service composition based on genetic algorithm," In Proc. of the 2005 Conf. On Genetic and Evolutionary Computation. Washington, 2005, pp. 1069–1075.

[6] Minghui Wu, Xianghui Xiong, Jing Ying, et al, "Qos-driven Global Optimization Approach for Large-scale Web Services Composition," Journal of Computers, vol. 6, no. 7, pp. 1452–1456, 2011.

[7] ZHANG Cheng-Wen, SU Sen, CHEN Jun-Liang, "Genetic Algorithm on Web Service Selection Supporting QoS," Chinese Journal of Computers, vol. 29, no. 7, pp. 1029–1037, 2006.

[8] LIU Shu-lei, LIU Yun-xiang, ZHANG Fan, et al, "A Dynamic Web Services Selection Algorithm with QoS Global Optimal in Web Services Composition," Journal of Software, vol. 18, no. 3, pp. 651–655, 2007.

[9] Mehrabian A R, Lucas C, "A novel numerical opimization algorithm inspired from weed colonization," Ecological Informatics, vol.1, no. 4, pp. 355-366, 2006.

[10] Roshanaei M M, Lucas C, Mehrabian A R, "Adaptive beamforming using a novel numerical optimisation algorithm," IET Microwaves, Antennas & Propagation, vol. 3, no. 5, pp. 765–773, 2009.

[11] Mallahzadeh A R, Oraizi H, Davoodi Z, "Application of the invasive weed optimization technique for antenna configuration," Progress In Electromagnetic Research, 2008, PIER 79, pp. 137–150.

[12] Rafal Zdunek, Tomasz Ignor, "UMTS Base Station Location Planning with Invasive Weed Optimization," ICAISC 2010, Part II, LNAI 6114, pp. 698–705.

[13] Zhang Xuncai, Wang Yanfeng, Cui Guangzhao, et al, "Application of a novel IWO to the design of encoding sequences for DNA computing," Computers&Mathematics with Applications, vol. 57, no. 11, pp. 2001–2008, 2009.

[14] Ritwik Giri, Aritra Chowdhury, Arnob Ghosh, et al, "A Modified Invasive Weed Optimization Algorithm for Training of Feed-Forward Neural Networks," IEEE International Conf. on System Man and Cybernetics, 2010, pp. 3166–3173.

[15] S Ran, "A Model for Web Services Discovery With QoS," ACM, 2003, pp. 6–9.

[16] Y. Liu, A. H. H. Ngu, and L. Zeng, "QoS Computation and Policing in Dynamic Web Service Selection," In Proceedings of the International World Wide Web Conference, 2004, pp. 66–73.

[17] M. Alrifai, T. Risse, "Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition," In Proc. of the 18th Int'1 Conf. On World Wide Web, 2009, pp. 881–882.

Kai Su received his B.S. degree in electronic information engineering from Nanchang University, Nanchang, China in June 2009 and his M.S. degree in communication Engineering from Naval University of Engineering, Wuhan, China in December 2011. He is a PhD candidate in computer science of Naval Univeristy of Engineering. He was born in 1987. His current research interests includes web services composition and service computing.

Liangli Ma received her PhD degree in Computer Science from HuaZhong University of Science and Techonology, Wuhan, China in 2006. She is a professor in Naval University of Engineering, Wuhan, China. She was born in 1969. Her research interests includes Business process technologies and Software testing.