ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

An Improved Genetic Algorithm for Web Service Composition Optimization

Mengmeng Du[1, a, *]

[1]School of Computer Science & Technology, Tiangong University, Tianjin, 300387, China.

[a]17097491@qq.com

Abstract

For solving a problem that the user's demand for web service in quality of service (QoS) is increase. Combining the elite genetic algorithm and the lexicographic optimization method, an improved genetic algorithm (IGA)for web service composition optimization is proposed. First, using the lexicographic optimization method (LOP) and threshold constraint relaxation technique (TCR) delete the candidate web service with lower QoS. Then, the web service composition problem is transformed into a single-objective optimization problem, and the elitist genetic algorithm is adopted search global optimal solution satisfying the demand of user. The simulation results show that IGA can efficiently and quickly select a web service with higher QoS.

Keywords

Web service composition, Quality of service, Lexicographic optimization, Elitist genetic algorithm.

1. INTRODUCTION

With the popularization of the Internet, there are Web services in various areas of life. Therefore, there are many Web services with the same function and different QoS [1][2]in the internet. In addition, as the needs of users have the characteristics of integration and complexity, a single Web service can no longer meet the demand [3]. For solving this problem, it is necessary that using Web service composition technology to combine loosely coupled web services in the Internet meeting the complex requirements of users [4].

In this article, in order to combine the web service distributed in the Internet to a comprehensive service satisfying user's demands for QoS, we propose an improved genetic algorithm (IGA) for web service composition optimization. The IGA mainly consists of two parts: First, for increasing efficiency, using the lexicographic optimization method (LOP) [5] and threshold constraint relaxation technology (TCR) [6] delete candidate services with lower QoS. Finally, the simple linear weighting method (SLW) [7] is used to convert the Web service composition problem into a single-objective optimization problem, and the elitist genetic algorithm (EGA) [8] is employed to search the Web composite service with the greatest fitness from the candidate services.

The remaining parts are organized as follows in this paper. Section 2 summarizes related work for Web service composition optimization. Section 3 introduces the model of Web service and QoS. Section 4 details the IGA. Section 5 presents the simulation results. Section 6 gives conclusions in this work.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

2. RELATED WORK

In past, many scholars have conducted research on the optimization of Web service composition.

In [9-10], the author adopts the genetic algorithm (GA) to select web composite service, and expresses Qos parameter by gene. The GA does not search for the global optimal solution, and has the disadvantage of weak local search ability. In [11], the author uses particle swarm optimization algorithm (PSO) to solve the problem of Web service optimization, but this algorithm has the trouble of premature convergence. In [12], the author proposed an improved artificial bee colony algorithm (IABC) to solve the problem of web service optimization, but as the scale of the problem increases, the efficiency of the algorithm will decrease. In [13], the author proposed an estimation of distribution algorithm (EOD) to solve the problem of web service optimization. Similarly, this method is not suitable for large-scale web service composition problems. In [14], the author regards the web service composition as a multi-objective optimization problem and uses ant colony algorithm (AC) to find the global optimal solution.

Different from previous studies, in this article, we combined the lexicographic optimization method [5] and threshold relaxation technology [6] to delete candidate services with lower QoS, and then use the elitist genetic algorithm to search for the final Web composite service from the candidate service according to fitness value.

3. MODELS DESCRIPTION AND OPTIMIZATION GOAL

In the process of Web service composition, there are mainly two types of services: the concrete services Cs and the abstract service As. The abstract service is defined as a service function and includes multiple concrete services with the same function and different QoS [15]. The concrete services are provided by the service provider [16]. The process of Web service composition is shown in Figure 1.

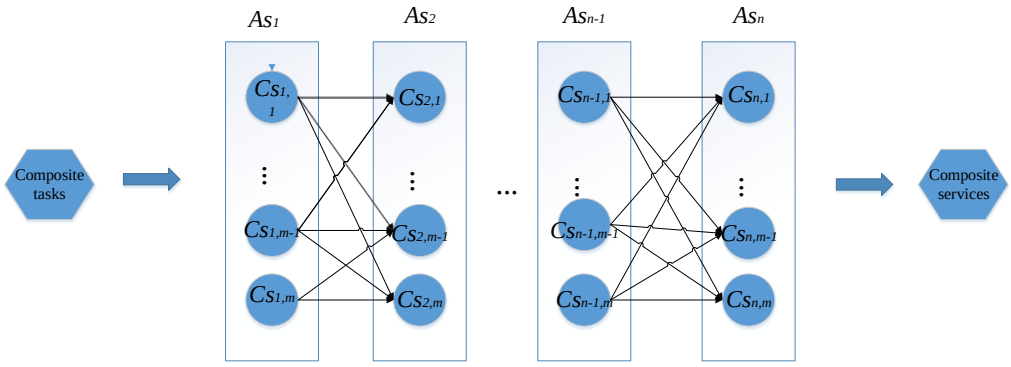


Figure 1. The model of Web service composition

3.1. The Web Service Models

In this article, the concrete service Cs_{ij}, also known as invocable web service, is provided by the service providers and can be called through the Internet As follows.

$$Cs_{ij} = \{QosA(Cs_{ij}), Action(Cs_{ij})\} \quad (1)$$

Where $QosA(Cs_{ij})$ is the QoS attribute of the invocable Web service and $Action(Cs_{ij})$ represents the function of the invocable Web service.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

The abstract service As_i consists of m invocable Web services, which have the same functions but different QoS levels. As follows:

$$As_i = \{Cs_{i1}, Cs_{i2}, ..., Cs_{ij}\} \text{ and } i \in [1,n], j \in [1,m] \tag{2}$$

3.2. The QoS model

3.2.1 The invocable Web service QoS model

The QoS attributes of invocable Web service usually have two types: the positive attribute and the negative attribute. The positive attribute means that, as the value of the QoS attribute increasing, the QoS of Web service increases. The negative attribute means that, with the value of the QoS attribute growing, the QoS of Web service decreases.

The QoS attributes of the concrete service Cs_{ij} are as follows:

$$QosA(Cs_{ij}) = \{qos_1^{ij}, qos_2^{ij}, ..., qos_t^{ij}\} \tag{3}$$

Where qos_t^{ij} represents the $t\text{-}th$ QoS attribute in the concrete service Cs_{ij}.

The QoS model of the concrete service Cs_{ij} is as follows:

$$\begin{cases} QaggregationP(Cs_{ij}) = \sum_{l=1}^{t_1} w_l qos_l^{ij}, if \ \ qos_t^{ij} \ \ is \ postive \\ QaggregationN(Cs_{ij}) = \sum_{l=1}^{t_2} w_l qos_l^{ij}, otherwise \\ \sum_{l=1}^{t} w_l = 1, and \ \ t_1 + t_2 = t \end{cases} \tag{4}$$

Where $QaggregationP(Cs_{ij})$ and $QaggregationN(Cs_{ij})$ are the QoS aggregation values for the positive and negative attributes in the concrete service Cs_{ij}, respectively, and w_l represents the weight of the user's preference for each QoS attributes.

$$Qaggregation(Cs_{ij}) = 1 \Big/ QaggregationN(Cs_{ij}) + QaggregationP(Cs_{ij}) \tag{5}$$

Where $Qaggregation(Cs_{ij})$ is the overall QoS aggregation value for the concrete service Cs_{ij}. The Web service with larger the overall QoS aggregation value shows that this Web service Cs_{ij} with the best QoS in abstract service As_i.

3.2.2 The QoS Model of Web Composite Service

The QoS of the composite service is affected by its own structure. Due to any complex structure can be transformed into a sequential structure through a certain technology [6], so the Web composite service in this article adopts a sequential structure.

3.3. The Optimization Goal

In this work, the Web service composition is finally transformed into a single-objective optimization problem, and the elitist genetic algorithm is used to find a Web composite service meeting the complex requirements. In the composition process, we will choose the final Web composite service based on the fitness. The optimization goal in this paper is as follows.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

$$Maximizing \rightarrow ComFitness(x) = \sum_{i=1}^{n} Qaggregation(x) \tag{6}$$

Where $ComFitness(x)$ is the fitness value of web composite service, n is the task numbers of the Web composite service, and x represents the concrete service Cs_{ij} contained in the Web composite service.

4. IGA: AN IMPROVED GENETIC ALGORITHM FOR WEB COMPOSITE SERVICE OPTIMIZATION

The IGA proposed in this paper consists of two steps. First, lexicographic optimization method [5] and threshold relaxation technology [6] are used to delete candidate services with lower QoS. Then, employing elitist genetic algorithm to search for the Web composite service within the global QoS constrains and the preferences of user.

4.1. The Deleting Web Service Concerning QoS and User'S Needs

In this article, we model the deleting Web service operation concerning QoS and user's needs as a multi-attribute decision-making problem, and use lexicographic optimization technology [5], and threshold relaxation technique [6] to delete candidates with lower QoS.

In this article, we innovatively regard the QoS aggregation value of the Web service as the sub-optimization problem wich be solved in the lexicographic optimization technology. the service deleting steps are as follows:

(1) Ranking QoS aggregation value according to the sum of user's preference weights for QoS.

(2) For each Web service in abstract service As_i, each QoS aggregation value is processed, negative or positive, according to the previous order.

A) Discover the optimal QoS aggregation value $Best_Q$ in abstract service As_i.

B) Using threshold relaxation technology [6], calculate the threshold of each QoS aggregation based on the $Best_Q$ and tolerance factor δ_{QoS}. As follows:

$$QoS_{Aggeration}^{Threshold} = \begin{cases} BestQ\text{-}QoS_{Aggeration}^{Decrease}, \ if\ QoS_{Aggregate}^{Value}\ is\ positive \\ BestQ + QoS_{Aggeration}^{Decrease}, \end{cases} \tag{7}$$

$$with\ \ QoS_{Aggeration}^{Decrease} = BestQ * \delta_{QoS} \tag{8}$$

(3) The Web service that is not within the escape of the threshold $QoS_{Aggeration}^{Threshold}$ will be deleted from the candidate service sets of the abstract service As_i.

The pseudo-code of the deleting Web service operation concerning QoS and user's needs based on lexicographic optimization and threshold relaxation technology is shown in Algorithm 1.

4.2. The Final Web Composite Service Selection Based on Elitist Genetic Algorithm

In this paper, the invocable web service with relatively low QoS are deleted for each abstract service As_i, which reduces the search domain to a certain extent. The nature of the Web service composition is an NP-hard problem. The genetic algorithm (GA) is a common method to solve this problem. In order to speed up efficiency, empolying elitist genetic algorithm [8] to find the web composite service. The flow chart of the elitist genetic algorithm is shown in Figure 2.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

For the elitist genetic algorithm, the current population is generated according to the number of the invocable web service contained in the candidate service set CAS. In order to ensure the complete evolution of the population, the evolution iterations changes adaptively according to the size of the web composite service, as shown below.

$$Gertaions = \frac{n*m}{10} \tag{9}$$

Where n is the total number of abstract service, and m is the number of concrete services for abstract service.

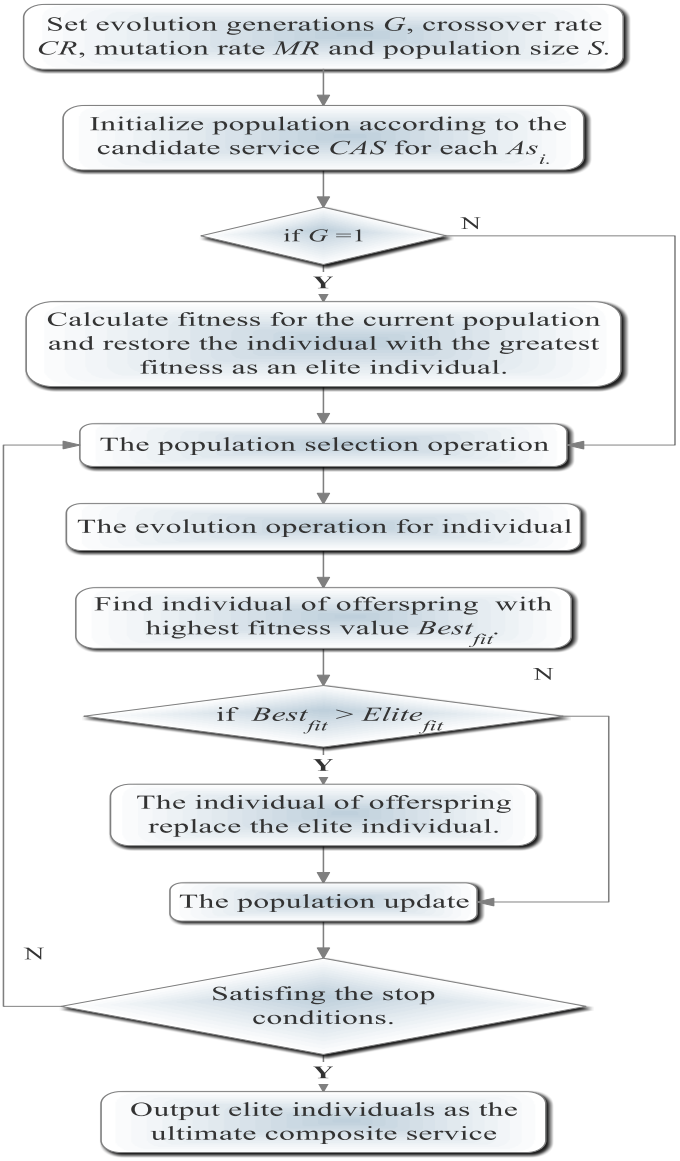


Figure 2. The process of elitist genetic algorithm

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

Algorithm 1— service preselection concerning QoS levels and user's QoS needs

Input": abstract service As_i containing m invocable web service.

δ_{QoS} : the tolerance factor for QoS aggregation value.

Step 1: the QoS aggregation values are ranked according to the user's preference weight for QoS Attribute. The rank is 1 or 2. The rand of Qos aggregation value with 1 have the highest priority.

Step 2: $CAS=As_i$

Step 3: for rank=1 to 2 do

Step 4: discover $Best_Q$ from CAS

Step 5: compute $QoS_{Aggeration}^{Decrease}$ according to (8)

Step 6: compute $QoS_{Aggeration}^{Threshold}$ according to (7)

Step 7: update CAS as follows:

Step 8: if current QoS aggregation value is positive then

Step 9: $CAS=\left\{Cs_{ij} \in CAS \cap QaggregationP(Cs_{ij}) \geq QoS_{Aggeration}^{Threshold}\right\}$

Step 10: else

Step 11: $CAS=\left\{Cs_{ij} \in CAS \cap QaggregationN(Cs_{ij}) \leq QoS_{Aggeration}^{Threshold}\right\}$

Step 12: end if

Step 13: end for

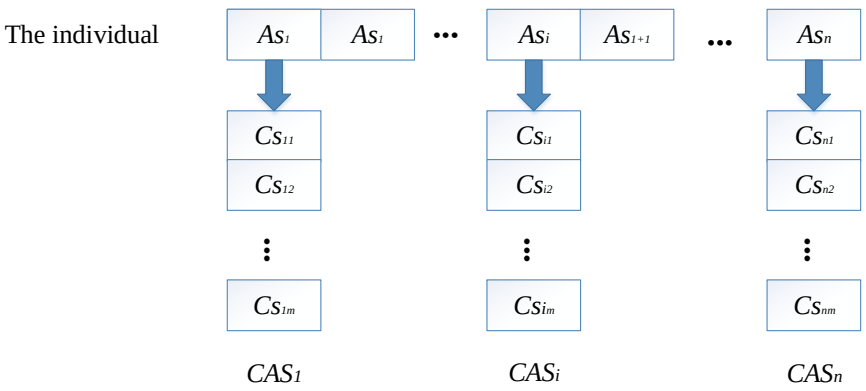
Outputs: the candidate service sets CAS for each abstract service class As_i.

4.2.1 The Population Selection Operation

Commonly used methods of population selection are roulette wheel selection [17], stochastic universal sampling [18], and tournament selection method [19]. In order to ensure the population diversity, this paper adopts the tournament selection method for population selection operations.

4.2.2 The Individual Coding in Evolutionary Operation

In this article, the coding of population individuals adopts integer coding. As shown in Figure 3, the individual length is the total number n of abstract service participating in service composition. The $i\text{-}th$ locus of an individual is the serial number of the invocable web service included in the candidate service sets CAS.



The pre-selecting candidate service *CAS* for each abstract service.

Figure 3. the individual coding method

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

4.2.3 The Individual Crossover Method in Evolutionary Operation

In order to speed up the evolution, this paper uses the single-point crossover [20] to perform the individual crossover operation.

4.2.4 The Individual Mutation Method in Evolutionary Operation

For ensuring the integrity of the individual, this paper uses the small disturbance mutation method to perform mutation operations for the individual as shown below:

Step 1: select a mutation point M_p form individual, randomly.

Step 2: chose a serial number for the candidate service sets for abstract service As_i to replace the serial number in M_p.

4.2.5 The Population Update Strategy

In order to ensure that population evolution does not undergo evolutionary recession. In this article, we use the coverage method to update the population. As follows:

Step1: arrange the individuals of the current population in descending order according to their fitness.

Step2: according to this order, the top Re_N individuals in the current population replaced the elitist individual.

$$Re_N = R_{replace} * Size_{pop} \tag{10}$$

Where $R_{replace}$ is the population replacement rate and $Size_{pop}$ is the current population size.

5. THE SIMULATION AND RESULT ANALYSIS

In this works, for verifying the performance of IGA, we use matlab2014a to conduct simulation experiments. In the simulation, IGA will be compared with traditional genetic algorithms (TGA) [21] in term of the maximum fitness of individual.

5.1. The Simulation Settings

In this paper, for truly simulate the process of web services composition in the internet, we consider five common QoS attributes for the web service, and all QoS attributes is uniform distribution in a certain interval as shown in Table 1. Meanwhile, the user's preference weight for reliability, reputation, security, cost, and response time are 0.3, 0.25,0.2, 0.15, and 0.1, respectively.

In this work, we set multiple abstract service numbers n and concrete service numbers m to evaluate the performance of IGA, and the detailed setting is shown in Table 2.

Table 1. QoS attributes considered for the simulation

The positive QoS attributes			The negative QoS attributes	
Reliability	Reputation	Security	Cost	Response time
[0.3,07]	[0.2,0.8]	[0.1,0.9]	[100,500]	[0.2,0.4]

Table 2. The parameters Setting for simulation

The condition number	The abstract service number n	The concrete service number m
1	10	Changing in [100,500]
2	Changing in [10,50]	100

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

5.2. The Results Analysis

In the simulation, by different parameter setting in table 2, we shows the performance of the IGA algorithm in terms of maximum fitness of individual.

5.2.1 Maximum Fitness of Individual Versus Evolutionary Iteration

In this simulation, the number of abstract service and the number of invocable web services are set to 10 and 500, the number of evolutionary iterations is 500 for the IGA and TGA algorithms, and the tolerance factor is set to 0.4, 0.5 and 0.6, respectively. From Figure.4, it is not difficult to see that compared to TGA, the IGA converges to the global optimal solution before 50 generations on average. Therefore, the convergence speed of IGA is higher than that of TGA.

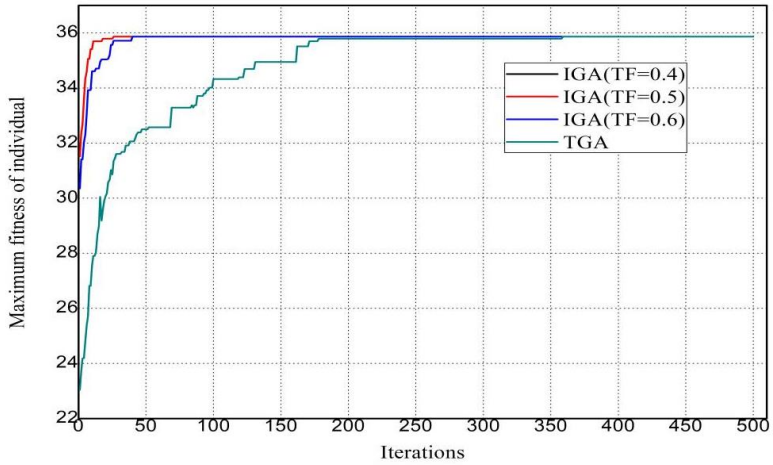


Figure 4. maximum fitness of individual for n=10 and m=500

For the second simulation, the number of abstract service and the number of invocable web services are set to 50 and 100, the number of evolutionary iterations is 500 for the IGA and TGA algorithms, and the tolerance factor is set to 0.4, 0.5 and 0.6 respectively. From Figure. 5, it is not difficult to see a phenomenon that compared to TGA, the IGA can converges to the global optimal solution before 300 generations on average. Therefore, the convergence speed of IGA is higher than that of TGA.

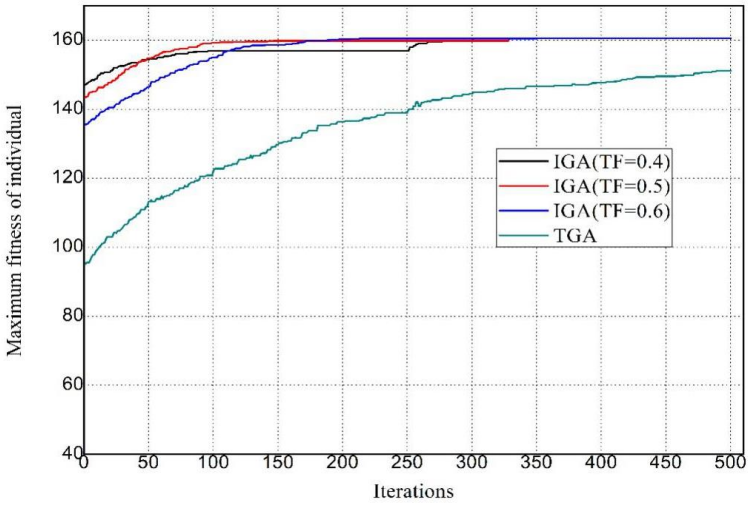


Figure 5. Maximum fitness of individual for n=50 and m=100

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

5.2.2 Maximum Fitness of Individual Versus Number of Abstract Service

In this simulation, the setting of parameters are consistent with condition 2 in table 2, and the tolerant factor is 0.5. From Figure. 6, it is very obvious to find that as the number of abstract service increases, the maximum individual fitness also increases, which is due to the fact that as the number of abstract service increases, the size of web composite services also increases.

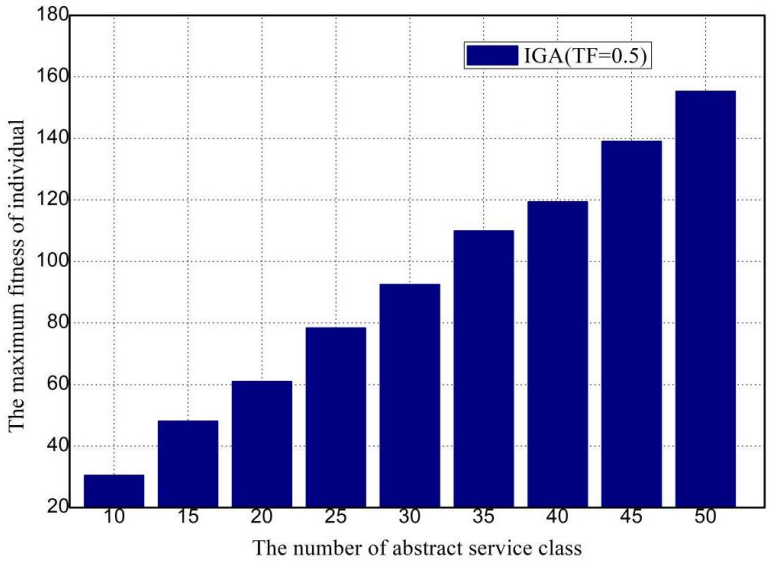


Figure 6. Maximum fitness of individual in different number of abstract service

5.2.3 Maximum Fitness of Individual Versus Number of Concrete Service

In this simulation, the setting of parameter are consistent with condition 1 in table 2, and the tolerant factor is 0.5. From Figure.7, it is not difficult to find that as the number of concrete services increases, the maximum individual fitness also increases, which is due to the fact that as the number of concrete services increases, the probability of choosing the service with the highest fitness also increases.

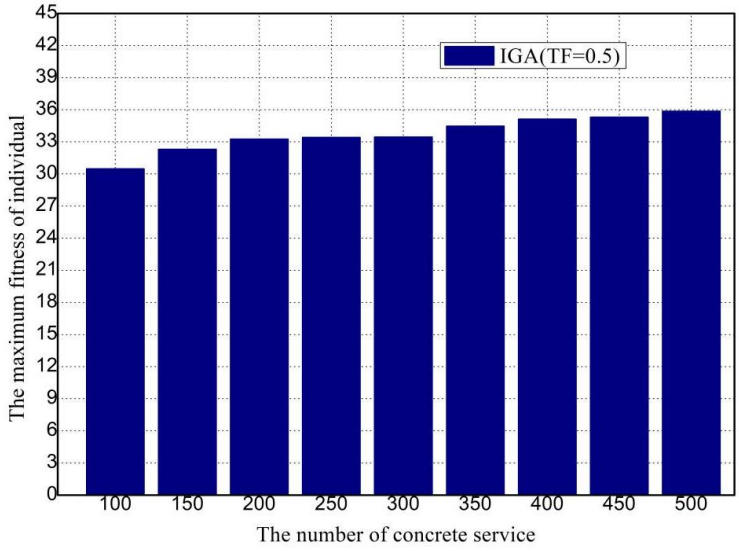


Figure 7. Maximum fitness of individual in different number of concrete service

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

6. CONCLUSIONS

In this paper, for dealing with the web service composition problem within QoS constraints and user's needs. We adopting lexicographic optimization technology [5], and threshold relaxation technique [6] to propose the IGA algorithm. The simulation results indicate that the global search capability and convergence speed of IGA are better than the TGA algorithm. Meanwhile, IGA can efficiently and accurately find the best Web composite service.

REFERENCES

[1] Zeng L Z,Benatallah B,Dumas M,et al.:QoS-aware Middleware for Web Services Composition,IEEE Transactions on Software Engineering, Vol.30(2004) No. 5, p.311-327.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash: Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, IEEE Communications Surveys & Tutorials, vol. 17(2015) No. 4, p. 2347-2376.

[3] Yangchao Ou,Bozhi Chen, Guodong Chen, Improved Genetic Algorithm for Web Service composition QoS optimization, Computer Engineering,Vol.43(2017) No.08, p.231-235+242. (In Chinese)

[4] Wenan Tan, Jiakai Wu, Web service composition optimization based on improved flowe-r pollination algorithm, Computer Engineering, Online: https://doi.org/10.19678/j.issn.1000-3428.0056206. (In Chinese,2020)

[5] P. C. Fishburn: Exceptional paper-lexicographic orders, utilities an decision rules: A survey,Manag. Sci., vol. 20(1974) No. 11, p. 1442–1471.

[6] M. E. Khanouche, Y. Amirat, A. Chibani, M. Kerkar and A. Yachir: Energy-Centered and QoS-Aware Services Selection for Internet of Things, *IEEE Transactions on Automation Science and Engineering*, vol. 13(2016) No. 3, p. 1256-1269.

[7] A. Yachir, Y. Amirat, A. Chibani, and N. Badache: Event-Aware Framework for Dynamic Services Discovery and Selection in the Context of Ambient Intelligence and Internet of Things, IEEE Transactions on Automation Science and Engineering, 13(2016) No. 1, p. 85-102, 2016.

[8] Cheng W , Shi H , Yin X , et al.:An Elitism Strategy Based Genetic Algorithm for Streaming Pattern Discovery in Wireless Sensor Networks, IEEE Communications Letters, Vol.15(2011) No.4, p.419-421.

[9] Meiling Cai, Maogui Li,Jie zhou: Multi-choice Web services composition based on multi-object genetic algorithm,Computer Engineering and Applicattions, Vol.46(2010) No.13,p. 202-205. (In Chinese)

[10] Xiaoyong Gong,Qingsheng Zhu,Chunling Wu: Improved genetic algorithm based on QoS in Web services composition ,Application Research of Computers, Vol. 25(2008) No.10, p.2922-2924. (In Chinese)

[11] Po Hu, Yuansheng Lou: Application of Improved Particle Swarm Optimization Algorithm in Service Composition, Computer Engineering, Vol. 37(2011) No.17, p.130-132. (In Chinese)

[12] Jun He,Liang Chen,Yonggang Li,Xiaolong Wang: Optimizing Web Service Composition Based on Improved Artificial Bee Colony Algorithm,Jouranl of Academy of Equipment,Vol. 24(2013)No.5, p.87-92. (In Chinese)

[13] Heng Liu,Gongrang Zhang,Man Luo: Web Service Composition Optimization Based on Estimation of Distribution Algorithm, Computer Technology and Development, Vol.06(2014), p.10-14. (In Chinese)

ISSN: 2472-3703 DOI: 10.6911/WSRJ.202010_6(10).0045

[14] Ma H , Zhou D , Liu C , et al.: Recommender systems with social regularization, Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM (New York, USA: ACM Press, February 9-12, 2011.).

[15] Li Zhang: Research on QoS-driven Service Composition Optimization Algorithm in Internet of Things(M.D Xi'an Shiyou University, China, 2018), p.45.

[16] Lu Chenghua, Kou Jisong: Optimization of Large Scale QoS-Oriented Web Service Composition Based on Multi-Objective and Multi-Attribute Decision Making, Chinese Journal of Management,Vol. 15(2018) No.04, p.586-597. (In Chinese)

[17] Yu-Hong L , Xin Z: An improved method for roulette wheel selection of genetic algorithm. Information Technology, (2009).

[18] Dan Stefanoiu,Janetta Culita,Florin Ionescu: Vibration fault diagnosis through genetic matching pursuit optimization, Soft Computing, Vol. 23(2019) No. 17.

[19] Coello C A C , Efrén Mezura Montes: Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Advanced Engineering Informatics: Vol.16(2002) No.3, p.193-203.

[20] Gamze Güngör-Demirci,Ayşegül Aksoy: Evaluation of the genetic algorithm parameters on the optimization performance: a case study on pump-and-treat remediation design, TOP,Vol.18(2010) No. 2, p.303-320.

[21] Yi J, Xing L, Wang G, et al.: Behavior of crossover operators in NSGA-III for large-scale optimization Problems, Information Sciences, Vol. 509(2018),p.470-487.