An Improved Genetic Algorithm for Web Services Selection*

Sen Su, Chengwen Zhang, and Junliang Chen

State Key Lab of Networking and Switching Technology
Beijing University of Posts & Telecommunications (BUPT), 187#
10 Xi Tu Cheng Rd., Beijing 100876, China
zwjcbj2007@gmail.com, {susen, chjl}@bupt.edu.cn

Abstract. An improved genetic algorithm is presented to select optimal web services composite plans from a lot of composite plans on the basis of global Quality-of-Service (QoS) constraints. The relation matrix coding scheme of genome is its basis. In this genetic algorithm, an especial fitness function and a mutation policy are proposed on the basis of the relation matrix coding scheme of genome. They enhance convergence of genetic algorithm and can get more excellent composite service plan because they accord with web services selection very well. The simulation results on QoS-aware web services selection have shown that the improved genetic algorithm can gain effectively the composite service plan that satisfies the global QoS requirements, and that the convergence of genetic algorithm was improved very well.

1 Introduction

Web service is a software application identified by an URL. The most-promising aspect of web service is the ability of engaging other web services in order to realize higher-order business transactions. Some interoperation mechanisms [1] are enabled in a service-oriented architecture. The framework of web services creates new possibilities to assemble distributed web services. How to create robust service compositions becomes the next step [15] and there are a lot of researches concentrated on it [8,9,16,17].

A composite service has specific functions that can be divided into some component functions. These component functions are accomplished by component services respectively. If the dependencies among component functions are represented through state charts that were used in [9], there are usually many available paths that can finish the same composite functions. So, web service composition has many scenarios [3], such as probabilistic invocation, parallel invocation, sequential activation and so on. If every component function is signified by a task, an execution path of a composite service can be constructed by a sequence of tasks including an

* The work presented in this paper was supported by the National Basic Research and Development Program (973 program) of China under Grant No. 2003CB314806; 863 program of China under Grant No.2006AA01Z164; the Program for New Century Excellent Talents in University of China under Grant No. NCET-05-0114.

J. Indulska and K. Raymond (Eds.): DAIS 2007, LNCS 4531, pp. 284–295, 2007.

initial task and a final task. In the phase of running time, some candidate services with same functions and different QoS attributes are discovered for every task. Thus, for each path, there are various composite plans corresponding to the specific function of composite service. Moreover, since component services with the same functions and different QoS are increasing with the proliferation of web services, the composite size should be larger and larger. For example, there are only one path that accords with the composite functions, 15 component functions in this composite path, and average 10 candidate web services for each component function. In this kind of composition scenario, the composite size should be about 10^{15}. Furthermore, since web services requesters always express both their functional requirements and their global QoS constraints set, it is needed to select which component services will be used in a given composite service in order to maximize user satisfaction, select the best composite plan from numerous plans and satisfy the consumers' global QoS constraints. Hence, web services selection with global QoS constraints plays an important role in web services composition [2, 3]. In the past years, the researches about web services selection have gained considerable momentums.

To figure out web services selection, some approaches are presented with the help of semantic web [4, 5, 6], and the others are based on QoS attributes computation [7, 8, 9, 10, 11, 23]. But the latter approaches are the more suitable solutions satisfying the global QoS requirements of web services selection. It is a combinatorial optimization issue that the best combination of web services is selected in order to accord with the global QoS constraints. Some traditional optimization techniques are proposed in [7, 8, 9, 23]. However, finding a plan for quality driven web services selection is NP-hard [11], so the effective strategies based on Genetic Algorithm (GA) are introduced in [10, 11].

Genetic Algorithm is a powerful tool to solve combinatorial optimizing problems [13]. It solves the formulated optimization problem using the idea of Darwinian evolution. It is an iterative procedure that consists of a constant-size population. Every individual describes a solution. Basic evolution operations, including crossover, mutation and selection operations, make GA be apt to very effectively perform global search. The design of genetic algorithm has the greatest influence on its behavior and performance [12], especially the design of coding scheme of chromosomes, fitness function, evolution operations and selection mechanism will have direct effect on efficiency and global astringency of genetic algorithm. It is necessary for GA to accord with characters of web services composition in order to get global convergence.

In the literatures, a suitable genetic algorithm for web services selection with global QoS constraints has not been taken into account, although the presented genetic algorithms can attain service composition supporting QoS to some extent. They always adopted the one dimension coding scheme that can not represent effectively the composite service re-planning, cyclic paths. The one dimension coding scheme can also not express all paths of assemble service at the same time. They did not think more about how to overcome the premature phenomenon of GA. Therefore, they did not suit effectively the issue about how to select the best composite plan from many plans of many paths in order to satisfy global QoS constraints.

Following the above analyses, we proposed a novel relation matrix coding scheme of chromosomes in [21], the relation matrix coding scheme suits with web service

composition with global QoS constraints more than the one dimension coding scheme. In [20], we presented a population diversity handling mechanism. But, the fitness function in [20] is not very fit one for many QoS properties with large quantity difference. Furthermore, the mutation policy should be designed on the basis of relation matrix coding scheme. Aiming at these issues, we discuss how to construct fitness function, mutation policy and present an improved fitness function and an improved mutation policy. Finally, the simulated results show that improved fitness function and improved mutation policy accord with QoS-aware web services selection and relation matrix coding scheme.

The remainder of this paper is organized as follows. After a review of the literature of web services selection in section 2, Section 3 presents the discussion of fitness function and mutation policy in detail. Section 4 describes simulations about fitness functions and mutation policies and discusses results aiming to support the work. Finally, our conclusions are given in section 5.

2 Quality Computation-Based Selection of Web Services

According to Std. ISO 8402 [18] and ITU E.800 [19], QoS may include a number of nonfunctional properties such as price, response time, availability and reputation. Thus, QoS value of a composition service can be achieved by fair computation of QoS of every component web services. In this section, some traditional optimization techniques [7, 8, 9, 23] and Genetic Algorithm (GA) [10, 11] in the literatures are discussed in detail.

The QoS computation based on QoS matrix is a representative solution. [7] ranked web services by means of normalizing QoS matrix, however, it was only a local optimization algorithm but not a global one for services selection. Other works in the area of QoS computation include [8, 9], which proposed local optimization and global planning. The local optimization approach could not take global QoS constraints into consideration. For example, there are only one path that accords with the composite functions, 15 component functions in this composite path, and average 10 candidate web services for each component function. In this kind of composition scenario, the composite size should be about 10^{15}. When the size of composite service is very large, the overhead of global planning is quite enormous. Hereby, both had limitation to some extent. [23] proposed pattern-wise QoS selection, which split the difference between the global planning selection and local optimization selection. Although its execution is faster than a true global planning approach, it misses global perspective. Furthermore, it takes a lot of cost on identifying pattern elements while the composition patterns are very complex. Especially, it will work very badly if the given user-defined QoS requirements go beyond all of offered QoS.

The above means are not able to resolve effectively the issue of web services selection with global QoS constraints belonging to the class of NP-hard [10]. GA is more suitable for this issue. But, GA can play an important role only while the combinatorial size is very large. Some numerical simulations in [22] show that the linear integer programming outperforms GA while the combinatorial size is small. Two different GAs were proposed in [10, 11].

In [10], binary strings of chromosome were proposed for service selection. Every gene in chromosome represented a service candidate with values of 0 and 1. Thereby, the more service candidates or web services clusters were, the longer chromosome was. Since at most only single service candidate could be selected in each of web services clusters, only one gene was "1" and others were "0" in all of genes of every cluster. When the number of component services and the number of candidate services of each component service are all very big, the length of genome will be very long. This kind of manner resulted in poor readability. Further, the authors proposed only coding manner of chromosome for service selection with little further information about the rest parts of genetic algorithm, such as selection mechanism.

In [11], a genetic algorithm was also used to tackle the service selection problem. The one dimension coding way of chromosome was proposed to express services composition, and each gene represented an abstract service of composite service. The value of abstract service was one of concrete services. The length of genome was shorter than the one in [10]. The change of the number of concrete services could not influence the length of genome. Therefore, the stability of genome length was better than [10]. The coding way of chromosome and the fitness function were all of which were proposed in [11], but without more information about the algorithm.

In [20], a genetic algorithm with population diversity handling is presented in order to maximize user satisfaction during composition of web services. Evolution is directed effectively through the conservation of the historical optimal population and the competition between the historical optimal population and the current population.

In [21], a special relation matrix coding scheme of chromosomes is presented. It suits with QoS-aware web service composition more than the one dimension coding scheme. The relation matrix has the ability to represent simultaneously the composite service re-planning, cyclic paths and many web service scenarios.

In addition to coding schemes, the other parts of genetic algorithm should also be taken into account in order to accord into the special points of web services selection with global QoS constraints, for example, fitness function, mutation policy.

3 Improved Genetic Algorithm

In this section, we present an improved genetic algorithm in order to resolve quality-driven selection, mainly including the design of fitness function and mutation policy. The relation matrix coding scheme is firstly reviewed because the mutation policy is based on it.

3.1 Relation Matrix Coding Scheme

In [21], a special relation matrix coding scheme was introduced using neighboring matrix. In the case of that the number of component services and dependencies among component services in every path are different from each other, the relation matrix coding schemes can express all paths of assemble service at the same time. The coding scheme has the function to express not only the relation among tasks but also paths information.

In this matrix, "n" is the number of all tasks in the services composition. The following is the definition of the relation matrix coding scheme.

(1) The g_{ii} is located at the main diagonal of the matrix for the *i*th locus of the chromosome and presents a task. The possible values of g_{ii} are the following:

a) The number "0" that represents that the pointed task is not included in the special services composition.

b) The number "-1" that represents that the pointed task is implementing while the composite service re-planning.

c) The number "-2" that represents that the pointed task becomes invalid while the composite service re-planning, such as all of candidate services of the pointed task become invalid or the pointed task is canceled for some reasons.

d) The number "-3" that represents that the selected concrete service of the pointed task has some changes while the composite service re-planning. These changes include that the selected concrete service becomes invalid or some QoS constraints of the selected concrete service have some changes.

e) If the number "t" represents that the sum number of concrete services of the pointed task, any number in the range of [1, t] represents that the pointed task is included in the special services composition and one concrete web service is selected.

(2) The g_{ij} represents the direct relation between the *i*th task and the *j*th task. Here, $i \neq j$.

Before g_{ij} is defined, four values of k1, k2, k3 and k4 should be defined firstly. The four values are adjustable and represent the different situations of parallel invocations. They are boolean variables coded. They are integer number in hexadecimal idea (they may have values: 100, 200, 400, 800, etc.). The following is the definition of g_{ij}:

a) The number "0" represents that the *i*th task is not the immediate predecessors of the *j*th task.

b) The number "p" represents that the *i*th task is the immediate predecessors of the *j*th task and the *i*th task invokes the *j*th task with probability "p". Here, $0 < p \leq 1$.

c) The number "m" represents that the *i*th task is the immediate predecessors of the *j*th task and the *i*th task invokes the *j*th task with "m" times. Here, $1 \leq m < \mathrm{Min}\{k1, k2, k3, k4\}$.

d) The number "k1" represents that all of parallel invocations of immediate predecessors of the *j*th task belong to one identical parallel invocations group. The *i*th task is one of the immediate predecessors.

e) The number "k2" represents that all of parallel invocations of immediate successors of the *i*th task belong to the same group. The *j*th task is one of the immediate successors.

f) The number "k3" represents that all of parallel invocations of immediate predecessors of the *j*th task belong to the different groups. The *i*th task is one of the immediate predecessors.

g) The number "k4" represents that all of parallel invocations of immediate successors of the *i*th task belong to the different groups. The *j*th task is one of the immediate successors.

Obviously, in the case of k3 and k4, it is necessary to seek a table of parallel invocations to find out which parallel invocations belong the same group.

By means of the combination of the values of m, p, k1, k2, k3, k4, many web service scenarios, such as probabilistic invocation, parallel invocation, sequential activation, etc, can be represented by the relation matrix. Additionally, the values of m, k1, k2, k3, k4 should not influence the decomposition of the value of g_{ij}. For example, if value of m is less than 100, values of k1, k2, k3 and k4 can be set as 0x100, 0x200, 0x400 and 0x800. Thus, value of g_{ij} can be decomposed precisely.

Following the definition of the relation matrix, the objects of the evolution operators are all of elements along the main diagonal of the matrix. The chromosome is made up of these elements. The other elements in the matrix are to be used to check whether the created new chromosomes by the crossover and mutation operators are available and to calculate the QoS values of chromosomes.

As stated the above, the abilities of the relation matrix are the following:

(1) The ability to seek simultaneously all of paths: since every locus of the chromosome can randomly be set to value "0", the chromosome has the ability to express all of paths of services composition.

(2) The abilities of the path re-planning and the task re-planning thanks to the introduction of values of "-1/-2/-3" to g_{ii}.

(3) The ability to resolve the cyclic paths thanks to the introduction of values of "m" to g_{ij}.

(4) The ability to represent simultaneously many web service scenarios, such as probabilistic invocation, parallel invocation, sequential activation, etc.

The following is one example of the relation matrix coding scheme.

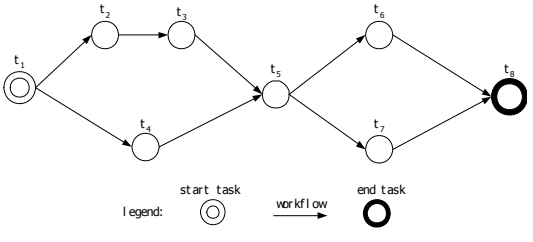

Fig. 1. Statechart of a web services composition

Figure 1 is one example of a web services composition. Its coding scheme is shown in figure 2.

$$\begin{pmatrix} t_1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & t_2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t_3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_5 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & t_6 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & t_7 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_8 \end{pmatrix}$$

Fig. 2. Coding scheme

In figure 2, all of numbers at the main diagonal of the matrix will be set at concrete genome.

3.2 Fitness Function

In [20], objective function is defined in (1) and it is named ObjectiveFunction 1(OF1 is its abbreviation):

$$f(g) = \frac{\sum_{j} (Q_j \times w_j)}{\sum_{k} (Q_k \times w_k)}. \tag{1}$$

Where $w_j, w_k \in [0,1]$, and w_j, w_k are real positive weight factors, represent the weight of criterion j and k. By providing w_j, w_k respectively, end users show their favoritism concerning QoS. The sum of all of them is 1. Q_j and Q_k denote values of the *j*th and *k*th QoS properties of the individual respectively. All of negative QoS properties (for example, price, time etc.) will be selected for the denominator (k). All of positive QoS properties (for example, reputation, availability etc.) will be selected for the numerator (j).

The formula (1) is not fit for the great quantity difference that QoS properties have. For example, response time and availability have large quantity difference. Furthermore, may be the same QoS properties have huge quantity difference in different web services. So, different QoS properties have not same influence on fitness function. It is not equitable for these QoS properties with low quantity level.

A method should be taken to transform values of all QoS properties into the range of [0,1]. In this way, all of QoS properties will have same influence on fitness function. A proportional fitness function is defined in formula (2) and it is named ObjectiveFunction 2(OF2 is its abbreviation):

$$f = \sum_{i=1}^{m} (w_i \times Q_i') \tag{2}$$

In formula (2), w_i is the same as formula (1). Q_i' denote the value of the *i*th QoS property of the individual. For negative QoS properties, values are scaled according to (3). For positive QoS properties, values are scaled according to (4).

$$Q_i' = \begin{cases} \dfrac{Q_i^{max} - Q_i}{Q_i^{max} - Q_i^{min}} & if\ Q_i^{max} - Q_i^{min} \neq 0 \\ 1 & if\ Q_i^{max} - Q_i^{min} = 0 \end{cases} \tag{3}$$

$$Q_i' = \begin{cases} \dfrac{Q_i - Q_i^{min}}{Q_i^{max} - Q_i^{min}} & if\ Q_i^{max} - Q_i^{min} \neq 0 \\ 1 & if\ Q_i^{max} - Q_i^{min} = 0 \end{cases} \tag{4}$$

In formula (3) and (4), Q_i^{max} and Q_i^{min} are the maximum and minimum of the *i*th QoS property of all individuals respectively. Q_i is the same as formula (1).

In order to express the difference of negative QoS properties and positive QoS properties further, the formula synthesizing the formula (1) and (2) is provided in formula (5) and it is named ObjectiveFunction 3(OF3 is its abbreviation):

$$f = \frac{\sum_{j} \left(\frac{Q_j - Q_j^{min}}{Q_j^{max} - Q_j^{min}} \times w_j\right)}{\sum_{k} \left(\frac{Q_k - Q_k^{min}}{Q_k^{max} - Q_k^{min}} \times w_k\right)},\ if\ Q_k^{max} - Q_k^{min} = 0 \text{ then } \frac{Q_k - Q_k^{min}}{Q_k^{max} - Q_k^{min}} = 1 \tag{5}$$

In formula (5), positive QoS properties are in the place of numerator and negative QoS properties are in denominator. The fitness function with penalty character is defined in formula (6):

$$Fit = f - \sum_{j=1}^{n} (\lambda_j \times \frac{\triangle P_j}{R_{jMax} - R_{jMin}}) \tag{6}$$

In formula (6), P_j represents the calculation value of a Q_i or some Q_is and these values are limited by a quality constraint. R_{jMax}, R_{jMin} are the maximum value and minimal value of calculation formula of the No.j quality constraint in all of web services composite plans. n is the number of quality constraints. λ_j is the calculation value of a Q_i or some Q_is and these values are limited by a quality constraint. It is a parameter used to adjust the scale of penalty value. The reason why λ_j is used: the higher users show their favoritism is, the bigger the penalty value is. Formula (7) is the definition of $\triangle P_j$:

$$\Delta P_j = \begin{cases} P_j - \min\{R_{jMax}, P_{jMax}\} & if \quad P_j > \min\{R_{jMax}, P_{jMax}\} \\ 0 & if \quad \max\{R_{jMin}, P_{jMin}\} \le P_j \le \min\{R_{jMax}, P_{jMax}\} \\ \max\{R_{jMin}, P_{jMin}\} - P_j & if \quad P_j < \max\{R_{jMin}, P_{jMin}\} \end{cases} \tag{7}$$

In formula (7), P_{jMax}, P_{jMin} are the maximum value and minimal value of the No.j quality constraint respectively.

In section 4, some simulations about these fitness functions will be provided.

3.3 Mutation Policy

Here, some mutation policies are proposed and discussed on the basis of the relation matrix coding scheme.

The first one is named MutationPolicy 1(MP1 is its abbreviation.): The probability of mutation is for the locus. Every locus in every chromosome will be asked whether mutating or not according to mutation probability. The child chromosome must be the same path as the mother chromosome during mutation operation. After the mutation operation, only the chromosome with the biggest fitness values will survive among the mother and child chromosomes. In fact, the mutation operation enhances the ability to search composition plans of every path.

The second one is named MP2: In the standard genetic algorithm, the probability of mutation is for the locus of chromosome. Here, in order to promote the probability to create different paths from the mutated path, the probability of mutation is for the chromosome instead of the locus. The concrete policy is as follows: before mutation operation of every chromosome, the probability of mutation is used to confirm whether the chromosome mutates or not. If mutation, the object path will be confirmed firstly whether it is the same as the current path expressed by the current chromosome. If difference, the object path will be selected from all available paths except the current one. If the object is itself, the new chromosome will be checked whether the new chromosome is the same as the old chromosome. Same chromosome will result in the mutation operation again. If the objects are different paths from the current path, a new chromosome will be created on the basis of the object path. Obviously, it is not necessary to check whether new and old chromosomes are same.

The third one is named MP3: It is similar to MP2. The different point is that the object path of the current path will be selected directly from all of available paths including the current path. This means that the objects are permitted to have the same paths as the current path

The fourth one is named MP4: The probability of mutation is for the locus. During the mutation of one task, the selection probability of every concrete service and the one of the "0" value are equal.

In section 4, some simulations about these mutation policies will be provided.

4 Experiments

To verify the excellence of GA we have proposed, numerous simulation comparisons had been performed on QoS-aware web services selection. All experiments were taken on same software and hardware, which were Pentium 1.6GHz processor, 512MB of RAM, Windows XP Pro, development language JAVA, IDE Eclipse 3.1. Same data were adopted for two compared GAs, including workflows of different sizes, 15-50 concrete web services for each task and 5 QoS data for each web service. A simplified representation of web service was used, including an ID number, some QoS data that were retrieved randomly in the range of defined values.

The following is about how these simulation data are produced. Firstly, there are three simulated compositions in all: the number of component functions is 10, 25 and 30 respectively. Composition state charts are used to represent the dependencies among component functions. Secondly, candidate services for each task are randomly created with one ID and values of five QoS properties. Finally, some global constraints of some QoS properties are provided for every composition. Then, these three composition situations are saved for the use later. Comparisons will be made in the same composition situation. In the three composition situations, the composite size is all very large. For example, there are 64 paths in the case of tasks 30, average 19 tasks in every path and average 15 candidates services for every task. Thus, the composite size is very enormous: 64×15^{19}.

The compared GAs were set up with same population size, crossover operation and probability, mutation probability. QoS model in [9] was used for them. The penalty technique is used for constrained optimization problems in algorithms. These algorithms have same selection mechanism of individuals, that is the "roulette wheel selection".

The fitness function and the mutation policy are the different points among the compared GAs.

4.1 Experiments on Fitness Function

There are three kinds of fitness function comparison between GA with the relation matrix coding scheme (the capital letter "A" represented it) and GA with the one dimension coding scheme (the capital letter "B" represented it). The three kinds of fitness function comparison are based on OF1, OF2 and OF3 respectively.

The following is the experiments of the three kinds of fitness function comparison. Some same parameters are population size 400, crossover probability 0.7, mutation probability 0.1, iterations 500, running times 50. The unit of time is *ms*. As shown in table 1, the statistic data of the average fitness, time and generation of the maximal fitness value were collected.

Table 1. Fitness, time and generation

Tasks Num	Comparison Name	Average Maximum Fitness	Average Time	Average Generation
10	A with OF1 : B with OF1	0.197 : 0.197	216 : 1582	5 : 518
10	A with OF2 : B with OF2	0.672 : 0.631	1298 : 4543	33 : 1027
10	A with OF3 : B with OF3	2.446 : 1.998	332 : 4769	7 : 1077
25	A with OF1 : B with OF1	0.217 : 0.066	3046 : 8891	42 : 8453
25	A with OF2 : B with OF2	0.638 : 0.538	15330 : 16369	188 : 8058
25	A with OF3 : B with OF3	1.658 : 0.601	12202 : 16677	146 : 8149
30	A with OF1 : B with OF1	0.191 : 0.053	6551 : 11361	78 : 16608
30	A with OF2 : B with OF2	0.628 : 0.529	20906 : 22746	219 : 16223
30	A with OF3 : B with OF3	1.515 : 0.541	18981 : 23198	199 : 16414

The above simulations show that GA with the relation matrix coding scheme and Objective Function OF3 is excellent than other GAs at the average fitness, time and generation. In table 1, "A" GA with OF3 can gain higher fitness value than other GAs. "A" GA with OF3 can have faster convergence speed than "A" GA with OF2. The reason is that OF3 express exacter comparison standard than OF1 and OF2.

4.2 Experiments on Mutation Policy

The mutation policy is the only different points between the compared GAs. Clearly, the compared GAs should adopt the relation matrix coding scheme. Some same parameters are population size 400, crossover probability 0.7, mutation probability 0.1, iterations 500, running times 50. The unit of time is *ms*.

The table 2 is the results of experiments among MP1, MP2, MP3 and MP4.

Table 2. Fitness (MP1:MP2:MP3:MP4)

Tasks Num	Average Maximum Fitness
10	0.196:0.191:0.191:0.196
25	0.193:0.165:0.141:0.089
30	0.152:0.132:0.108:0.069

In table 2, the largest fitness value is from MP1. These mean that the MP1 is the most effective mutation policy among MP1, MP2, MP3 and MP4. MP2 and MP3 increase the probability to create the different paths from the mutated path. This is the reason that MP2 and MP3 are better than MP4. If mutation, the probability to hold the mutated path in MP2 is 0.5, but the probability to hold the mutated path in MP3 is the value of 1 divided by the number of all paths. So, MP3 has higher probability to lose the good genetic information from the predecessor populations than MP2. This is why MP2 has better fitness than MP3. The MP1 increases the probability to search more composition paths and only the chromosome with the biggest fitness values will survive among mother and child chromosomes. Thus, the evolution direction is enhanced. These are why MP1 is best one among these mutation policies.

5 Conclusions

The web services selection with global QoS restrictions is an active research area. In this paper, we discuss fitness function and mutation policy of GA on the basis of the relation matrix coding scheme of genome. After discussion, the improved fitness function and mutation policy are proposed. They direct the evolution of GA. They also improve the convergence speed of GA. While GA includes the relation matrix coding scheme and Objective Function OF3, it can gain 30 times fitness value than the GA with one dimension coding scheme and Objective Function OF1. The results of experiments show that genetic algorithm with improved fitness function and mutation policy can get more excellent composite service plan.

To provide adaptive capability of genetic algorithms is an active research area [14]. Therefore, how to design a self-adaptive genetic algorithm for QoS-aware selection is one of our future works.

References

1. W3C.Web Services Architecture (2004) http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
2. Menascé, D.A.: QoS Issues in Web Services. IEEE Internet Computing 6(6), 72–75 (2002)
3. Menascé, D.A.: Composing Web Services: A QoS View. IEEE Internet Computing 8(6), 88–90 (2004)
4. Tian, M., Gramm, A., Ritter, H., Schiller, J.: Efficient Selection and Monitoring of QoS-Aware Web Services with the WS-QoS Framework. IEEE/WIC/ACM International Conference on Web Intelligence (WI'04) (2004)
5. Soydan Bilgin, A., Singh, M.P.: A DAML-Based Repository for QoS-Aware Semantic Web Service Selection. In: Proceedings of the IEEE International Conference on Web Services (ICWS'04) (2004)
6. Zhou, C., Chia, L.-T., Lee, B.-S.: DAML-QoS Ontology for Web Services. In: IEEE International Conference on Web Services (ICWS'04) (2004)
7. Liu, Y., Ngu, A.H., Zeng, L.: QoS Computation and Policing in Dynamic Web Service Selection. In: Proceedings of the 13th International Conference on World Wide Web (WWW), pp. 66–73. ACM Press, New York (2004)

8. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality Driven Web Services Composition. In: Proc. 12th Int'l Conf. World Wide Web (WWW) (2003)
9. Zeng, L., Benatallah, B., Ngu, A.H.H., et al.: QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)
10. Zhang, L., Li, B., Chao, T., et al.: On Demand Web Services-Based Business Process Composition. IEEE, pp. 4057–4064 (2003)
11. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: A Lightweight Approach for QoS–Aware Service Composition. ICSOC (2004)
12. Ignacio, R., Jesús, G., Héctor, P., et al.: Statistical Analysis of the Main Parameters Involved in the Design of a Genetic Algorithm. IEEE Transactions on Systems, Man, and Cybernetics—Part. C: Applications and Reviews 32(1), 31–37 (2002)
13. Srinivas, M., Patnaik, L.M.: Genetic Algorithm: a Survey. IEEE, pp. 17-26 (1994)
14. Hinterding, R., Michalewicz, Z., Eiben, A.E.: Adaptation in Evolutionary Computation: a Survey. IEEE EC, pp. 65–69 (1997)
15. Curbera, F., Khalaf, R., Mukhi, N., et al.: The Next Step in Web Services. Communication of the ACM 46(10), 29–34 (2003)
16. Milanovic, N., Malek, M.: Current Solutions for Web Service Composition. IEEE Internet Computing, pp. 51–59 (2004)
17. Orriens, B., Yang, J., Papazoglou, M P: Model Driven Service Composition. In the First International Conference on Service Oriented Computing (ICSOC'03) (2003)
18. ISO 8402, Quality Vocabulary
19. ITU-T Recommendation E.800, Terms and Definitions Related to Quality of Service and Network Performance Including Dependability (1994)
20. Zhang, C., Su, S., Chen, J.: Efficient Population Diversity Handling Genetic Algorithm For Qos-Aware Web Services Selection. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J.J. (eds.) ICCS 2006. LNCS, vol. 3994, pp. 104–111. Springer, Heidelberg (2006)
21. Zhang, C., Su, S., Chen, J.: A Novel Genetic Algorithm For Qos-Aware Web Services Selection. In: IEEE CEC'06 and EEE'06, USA. LNCS, vol. 4055, Springer-Verlag, Berlin (2006)
22. Canfora, G., Di Penta, M., Esposito, R., et al.: An Approach For QoS-Aware Service Composition Based On Genetic Algorithms, Genetic and Evolutionary Computation Conference (GECCO), Washington DC, USA, vol.1, pp. 1069–1075 (2005)
23. Grønmo, R., Jaeger, M.C.: Model-Driven Methodology for Building QoS-Optimised Web Service Compositions. The 5th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS), Athens, Greece (June 2005)