Research on an Adaptive Algorithm of Service Selection in Pervasive Computing

Liu Wei[1], Zhao Lanfei[2], Deng Jianqiang[3]
[1]Department of Medical Information
[2]Xichang Satellite Launch Center
[3]Department of Basic Medical Sciences
[1, 3] Hainan Medical University. Hainan, China
liuwei165@foxmail.com

Journal of Digital Information Management

ABSTRACT: *An adaptive genetic algorithm of service selection in pervasive computing is presented in this paper. By means of matrix encoding, this algorithm carries out selection, crossover and mutation operations of genetic algorithm, with matrix as individual chromosome and matrix array as gene. Based on the elitism selection strategy and adaptive strategy, this algorithm replicates the optimal individual of every generation to the next generation, and adaptively adjusts crossover and mutation probabilities according to the fitness value of individual, which is aimed at protecting the individuals with high fitness and weeding out those with low fitness. Simulated comparison proves that this algorithm is of good convergence, which also shows more stable optimization ability, compared with ordinary genetic algorithm and random selection algorithm.*

Categories and Subject Descriptors:
G.1 [Numerical Analysis]: Adaptive and Iterative Quadrature; **C.4 [Performance Systems]:** Reliability, Availability, and Serviceability

General Terms:
Pervasive Computing, Adaptive Algorithm

Keywords: Pervasive Computing, Service Selection, Elitism Selection, Adaptive Algorithm

Received:
9 July 2013, Revised 2 September 2013, Accepted 9 September 2013

1. Introduction

Pervasive computing aims not only to make the service available for everyone from anywhere at any time, but also to turn computer-centered service into user-centered service. In the context of pervasive computing, human-computer interaction will become more flexible. Meanwhile, with the gradual expansion of wireless network coverage, people can get services at any time and any place through computers and all portable and wearable devices, which, as a result, requires that pervasive computing service system should achieve the ability of seamless migration of application service and the adaptive ability. The collaborative process, in which service provider publishes services and service requester applies all sorts of terminal devices to perceive and acquire services, requires pervasive computing service system to guarantee service quality while improving the utilization rate of server's limited resources on the premise of meeting customer demand according to the characteristics such as user's device diversity, location mobility and demand flexibility. Meanwhile, this system also needs to select optimal services from service list to form the target services that suit customer demand best.

To address the problems above, this paper analyzes the service selection model, and meanwhile, on the basis of genetic algorithm, it proposes an adaptive algorithm of optimal service selection based on multi-objective constraints. The contributions made by this paper are the designs of suitable fitness function based on multi-objective constraints, elitism selection strategy, and adaptive adjustment strategy, all of which are intended to heighten the effectiveness of genetic algorithm. Meanwhile, the contribution also includes the analysis of simulated experiment on this algorithm.

2. Analysis on service selection model

In the environment of pervasive computing, server publishes services in pervasive computing service system. Through the intelligent terminal, users can access the service at any time and any place without considering the service installation and maintenance details, and then achieve personalized on-demand computing environment. Figure 1 shows the working schematic diagram of pervasive computing service system.

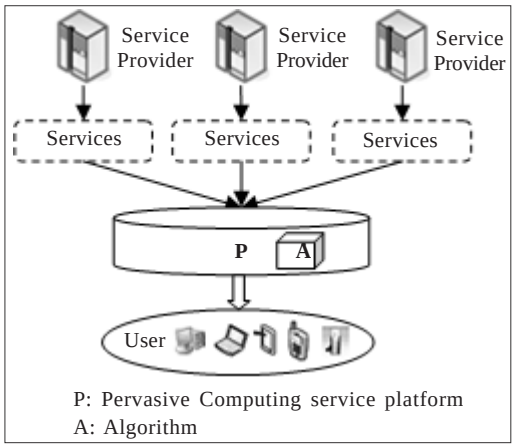

Figure 1. The working schematic diagram of pervasive computing service system

The process of service selection consists of three steps:

Step 1: Perceive the services published by service provider to work out service list. Assuming that P represents service provider, S means the service published by service provider, one service provider can publish multiple services, multiple service providers can publish the same service, and there are n P and m published services, then matrix can be used to express the relationship between service provider P and service S, which is recorded as PS matrix (1).

$$PS = \begin{bmatrix} P_1S_1 & P_1S_2 \ldots P_1S_m \\ P_2S_1 & P_2S_2 \ldots P_2S_m \\ \cdots & \cdots \ldots \cdots \\ P_nS_1 & P_nS_2 \ldots P_nS_m \end{bmatrix} \quad (1)$$

Where P_nS_m represents the state in which P_n publish S_m, $P_nS_m = 0$ means that P_n have published S_m, and S_m, are valid; $P_nS_m = -1$ suggests that P_n have published S_m but S_m are invalid; $P_nS_m = -2$ implies that P_n haven't published S_m.

Step 2: User makes a request to service, and only the published and valid service can receive the request. Assuming that U represents user, A means request, one user can make multiple service requests; one user can make only a valid default request to one service, and there are h users presenting k service requests, then matrix can be used to express the relationship between user U and request A, which is recorded as UA matrix (2).

$$UA = \begin{bmatrix} U_1A_1 & U_1A_2 \ldots U_1A_k \\ U_2A_1 & U_2A_2 \ldots U_2A_k \\ \cdots & \cdots \ldots \cdots \\ U_hA_1 & U_hA_2 \ldots U_hA_k \end{bmatrix} \quad (2)$$

Where U_hA_k represents the state in which users U_h send service requests A_k, $U_hA_k = -1$ represents that U_h haven't sent A_k, $U_hA_k = 0$ implies that U_h have sent A_k, but A_k are invalid; $U_hA_k = 1$ suggests that U_h have sent A_k, and A_k are valid.

User can only make requests to the published service, and if service provider hasn't published a certain service, then user cannot make a request to this service. Therefore, the set of user requests should be contained in the set of the services published by service provider, as shown in (3).

$$\{A_1, A_2, A_3, \ldots A_k\} \subseteq \{S_1, S_2, S_3, \ldots S_m\} \quad (3)$$

Step 3: Select services according to customer demand and objective constraints. In combination with PS matrix and UA matrix, an $n \times m$ matrix is applied to expressing the relationship between service provider P and user request A, which is recorded as R matrix (4).

$$R = \begin{bmatrix} R_{11} & R_{12} \ldots R_{1m} \\ R_{21} & R_{22} \ldots R_{2m} \\ \cdots & \cdots \ldots \cdots \\ R_{n1} & R_{n2} \ldots R_{nm} \end{bmatrix} \quad (4)$$

R matrix is transformed from PS matrix. R_{nm} represents the number of requests A to which service provider P responds. the value range of R_{nm} is $\{-2 \le R_{nm} \le x$, and R_{nm} is an integer$\}$, where –2 means the service does not exist; –1 means the service is invalid, and x represents the upper limit of responding to request.

According to the analysis on service selection model, service selection faces a many-to-many relationship, and to find the optimal solution of service selection is to work out the optimal R matrix under objective condition, with PS matrix (1), UA matrix (2) and relationship (3) as constraints.

3. Adaptive genetic algorithm of optimal service selection

According to characteristics of service selection, this paper comes up with an adaptive genetic algorithm based on matrix encoding with matrix constituting the individuals of population. Meanwhile, in accordance with multi-objective fitness function, this paper engages in individual selection and generates new generation of population after selection, crossover and mutation until the algorithm satisfies terminating condition. The detailed discussion about matrix encoding, fitness function design and adaptive strategy is presented as follows.

3.1 Matrix encoding

Based on the prototype of R matrix, genetic algorithm in this paper designs $n \times m$ chromosome matrix by means of matrix encoding, with matrix as individual chromosome and matrix array as gene, as shown in (5).

$$\begin{bmatrix} g_{11} & g_{12} \ldots g_{1m} \\ g_{21} & g_{22} \ldots g_{2m} \\ \cdots & \cdots \ldots \cdots \\ g_{n1} & g_{n2} \ldots g_{nm} \end{bmatrix} \quad (5)$$

The value range of g_{ij} is $\{-2 \le g_{ij} \le x$, and g_{ij} is an integer$\}$, which has the same meaning with R matrix. After encoding confirmation, initial population will generate randomly in a certain range according to the constraints.

3.2 Multi-objective fitness function

Fitness function proves to be the key factor that guides genetic algorithm to approach the optimal solution. The greater the fitness value of chromosome is, the huger the probability that individuals are selected will be, for individual selection is dependent on the fitness value of chromosome.

This paper holds that objective constraint of service selection should give consideration to two aspects: (1) service provider is required to raise the utilization rate of server's limited resources so as to achieve the goal of high-efficient service release and low cost; (2) user is supposed to meet customer demand and ensure QoS (Quality Of Service). Thus, fitness function design is expected to take into account multi-objective constraints. This paper will figure out a solution by dint of fitness function in literature [1].

Literature [1] designs two fitness functions, namely $f1$ and $f2$. $f1$ represents fitness function of service provider, which takes load balance as high-efficient and low-cost target solution of service provider. See below for service node load balance formula (6) and $f1$ calculation formula (7).

$$Load = \begin{cases} 1 & load_C \geq 1 \text{ or } load_m \geq 1 \\ W_C \times load_C + W_m \times load_m & \text{else} \end{cases} \quad (6)$$

$$\begin{aligned} f_1 &= Load_{bal} = \min\{A \,.\, D\} \\ &= \min\left\{ \sum_{i=1}^{n} | Load(i) - Load_{ave} | \Big/ n \right\} \end{aligned} \quad (7)$$

$F2$ figures out whether individual chromosome satisfies QoS requirement from the perspective of users. Literature [1] insists that QoS of user should comprise the attributes, such as cost, time, availability, robustness and security. Assuming that QoS is depicted by r OoS attributes, namely, $\{Q_1, Q_2, \ldots, Q_r\}$, then f_2 calculation formula (8) can be shown as below:

$$f_2 = \sum_{i=1}^{r} (w_i \times Q_i) \quad (8)$$

Where $0 \leqq w_i \leqq 1$, $\sum_{i=1}^{r} w_i = 1$.

In combination with two fitness sub-functions, total fitness function formula (9) are worked out as follows based on multi-objective constraints:

$$f = w_L(1 - f_1) + w_Q \times f_2 \quad (9)$$

Where w_L and w_Q are weight value, $0 \leqq w_L, w_Q \leqq 1$, $w_L + w_Q = 1$.

3.3 Genetic operation

Genetic operation is composed of three basic genetic operators, including selection, crossover and mutation. The previous two perform a large number of search functions of genetic algorithm, with mutation increasing genetic algorithm's ability to find approximate optimal solution.

3.3.1 Selection

Selection operator of simple genetic algorithm encounters selection error. The selections made according to random number may include some selections that are unable to reflect individual fitness, which, as a result, leads to the removal of individuals with high fitness. To protect optimal individuals of every generation from being damaged, this paper carries out selection operation by means of elitism selection strategy and genetic roulette. First, the individuals with the highest fitness in the population will be directly replicated to the next generation without pairing and crossover. Then, genetic roulette will be applied to the population. Assuming that there are n chromosomes in the population, and $f(x_i)$ represents the fitness value of x_i, the ith chromosome, then the Probability $P(x_i)$ that individual x_i is selected will be:

$$P(x_i) = f(x_i) \Big/ \sum_{i=1}^{n} f(x_i) \quad (10)$$

3.3.2 Crossover

Crossover operation means selecting two individuals randomly in pairing pool according to a certain probability P_c, and then engaging their partial genes in random crossing-over to form two new individuals. Crossover operation gets involved in crossing-over based on the unit of matrix array, for the algorithm in this paper sees matrix as chromosome and matrix array as gene.

The greater crossover probability P_c is, the higher the speed of generating new individuals will be, which is however more likely to damage genetic mode and the individuals with high fitness. Nevertheless, if P_c is too small, the searching process will be slow without moving forward. According to the method described in reference [5], this paper design an adaptive adjustment formula of P_c (11), which can realize adaptive adjustment with the change of fitness as follows:

$$P_c = \begin{cases} \dfrac{f_{max} - f'}{f_{max} - f_{min}} \times k_1 + k_2' & f' \geq f_{avg} \\ \dfrac{f_{max} - f'}{f_{max} - f_{min}} \times k_1 + k_2 & f' < f_{avg} \end{cases} \quad (11)$$

Where f_{max} represents the best fitness value in population; fmin the worst fitness value; f_{avg} the average fitness, and f' the fitness value of the heavier individuals of two individuals that need crossover. The value interval of P_c can be adjusted by means of k_1, k_2, and k_2'. This formula can capacitate P_c to make adjustments automatically with the change of fitness values, so the individuals with fitness value better than average fitness value of the population to achieve relatively smaller P_c. In this way, such individuals will be protected and enter next generation. Meanwhile, this formula can also enable the individuals

with fitness value worse than average fitness value of population to achieve relatively high P_c and then weed out these individuals.

3.3.3 Mutation

Mutation operation is to change the values of certain chromo genes randomly with a very small probability Pm to produce new individuals. Mutation operation gets involved in variation based on the unit of matrix array, for genetic algorithm in this paper sees matrix as chromosome and matrix array as gene.

The excessively small mutation probability P_m will lead to difficulties in the generation of new individual structure, whereas the excessively large P_m will turn genetic algorithm into random search algorithm. According to the method described in reference [5], this paper designs adaptive adjustment formula of P_m (12), which can realize automatic adjustment with the change of fitness value as follows:

$$P_m = \begin{cases} \dfrac{f_{max} - f}{f_{max} - f_{min}} * k_3 & f \geq f_{avg} \\ \dfrac{f_{max} - f}{f_{max} - f_{min}} * k_3 + k_4 & f < f_{avg} \end{cases} \quad (12)$$

Where f_{max} represents the best fitness value in population, f_{min} the worst fitness value; f_{avg} the average fitness, and f the fitness value of individuals that need mutation. The value interval of P_m can be adjusted by means of k_3 and k_4. This formula can capacitate P_m to make adjustments automatically with the change of fitness values, so the individuals with fitness value higher than average fitness value of the population to achieve relatively smaller P_m. In this way, such individuals will be protected and enter next generation. Meanwhile, this formula can also enable the individuals with fitness value smaller than average fitness value of population to achieve relatively high P_m and then weed out these individuals.

Elitism selection strategy and the adaptive adjustment of P_c and P_m can protect optimal individuals of every generation from being damaged and replicate them to next generation. The closer the fitness value gets to the largest fitness value, the smaller the crossover and mutation rates will be, which can thus protect superior individuals. However, the smaller the fitness value is, the larger the crossover and mutation rates will be, which can damage inferior individuals.

3.3.4 Algorithm process

Based on fitness function, elitism selection strategy, crossover probability P_c and mutation probability P_m, this paper puts forward the process of genetic algorithm as follows:

Step 1: Based on PS matrix, generate m R matrices to form initial population in accordance with change rule;

Step 2: Evaluate fitness value of every chromosome; save the chromosomes with the best and worst fitness values as variables; and calculate the average fitness value.

Step 3: Judge whether terminating condition is satisfied. If the condition is satisfied, the result should be outputted and the algorithm should stop; otherwise, the algorithm should continue.

Step 4: Perform selection operation for population; replicate elitist chromosomes with best fitness value to next generation; and then make a selection among the population according to $P(x_i)$.

Step 5: Calculate crossover probability P_c to perform crossover operation;

Step 6: Calculate mutation probability P_m to perform mutation operation; turn to step 2.

4. Simulation and result analysis

Analysis of simulation experiment is made to verify the effectiveness of algorithm. The experimental environment is: CPU: Pentium (R) Dual-Core CPU E5300 @ 2.60GHz; Memory: 2 GB; OS: Microsoft Windows XP Professional SP3; Programming Language: Standard C; Compile Environment: Visual Studio.net 2005.

4.1 Algorithm convergence

De Jong evaluates the performance of genetic algorithm with on-line performance and off-line performance. The off-line performance represents the accumulated average of best performance value of the evolutionary generation in the process of algorithm operation, embodying the convergence performance of algorithm as the commonly-applied performance in genetic algorithm for the time being. See below for its calculation formula (13).

$$X_e(t) = \frac{1}{t}\sum_{i=1}^{t} f_e(i) \quad (13)$$

Where $f_e(i)$ is the objective function value of the best individual in the ith generation in environment e, and $X_e(t)$ represents the average of the objective function values of best individuals from the first and the tth generation.

In case when multiple users make requests in our simulation, the system will generate off-line performance of service selection algorithm. PS matrix, and UA matrix randomly according to a certain rule. With every server providing 0 to 5 services and every user requesting for 0 to 5 services randomly, crossover probability P_c and mutation probability P_m experience adaptive computing. If there are 1 user, 6 servers, 30 iteration groups, 100 generations, then the off-line performance of algorithm will be shown in Figure 2.

If there are 5 user, 6 servers, 100 iteration groups, 10000 generations, then the off-line performance of algorithm will be shown in Figure 3.

If there are 10 user, 6 servers, 150 iteration groups, 20000

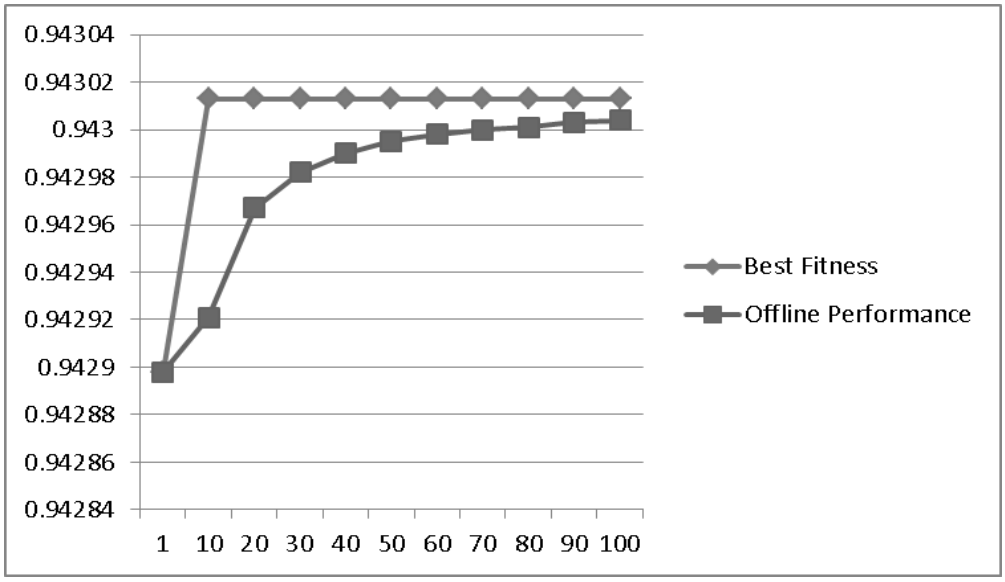

Figure 2. Diagram of comparison between off-line performance and best fitness of 1 user

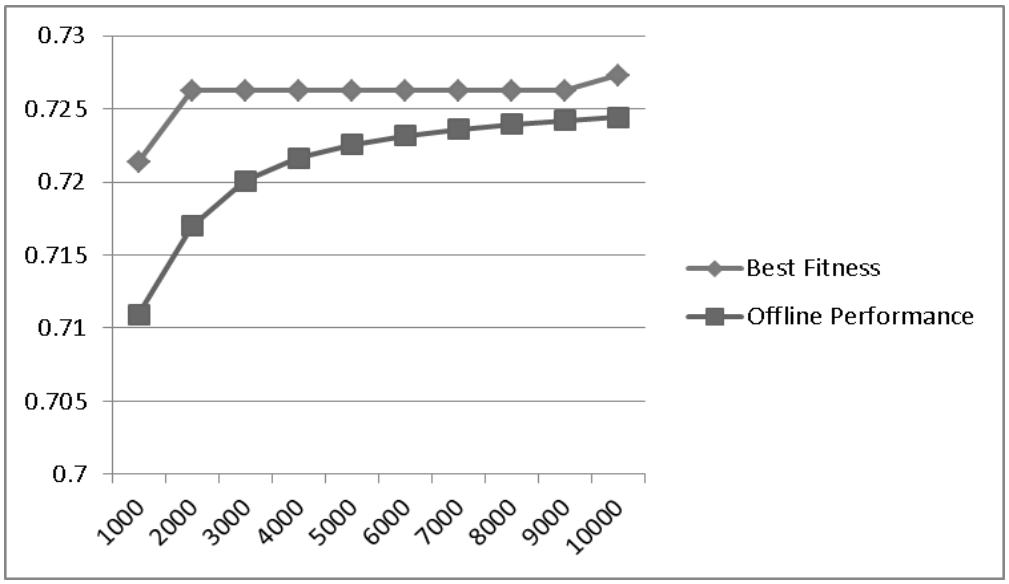

Figure 3. Diagram of comparison between off-line performance and best fitness of 5 user

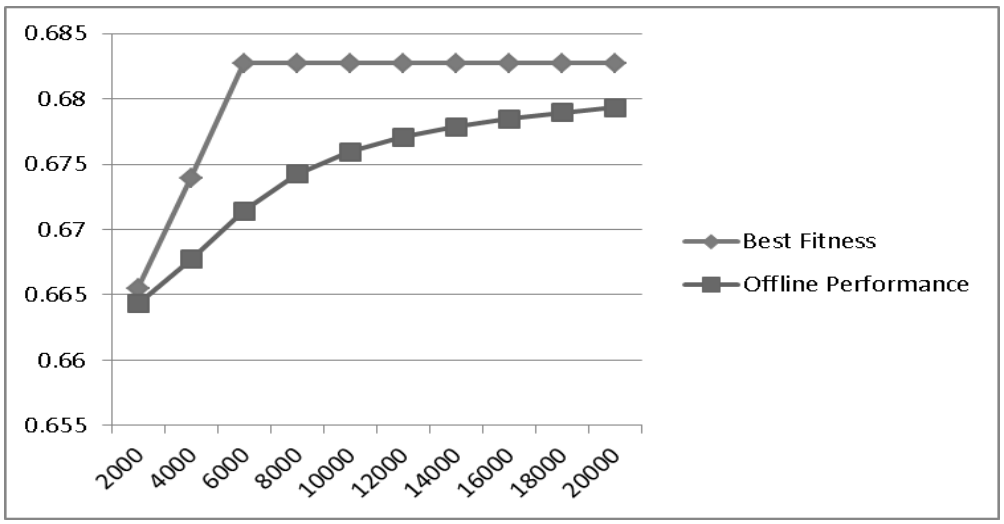

Figure 4. Diagram of comparison between off-line performance and best fitness of 10 user

generations, then the off-line performance of algorithm will be shown in Figure 4.

In Figure 2, Figure 3 and Figure 4, with the number of interations increasing, off-line performance curve becomes similar to best fitness curve gradually, which suggests that algorithm shows good convergence with the increase of the number of interaitons. When users increase from 1 to 10, off-line performance curve still has a tendence to best fitness curve, which reveals that the algorithm sill shows good convergence with the increase of user quantity.

4.2 Optimization stability of algorithm

To verify the stability of optimization stability, this paper compares adaptive genetic algorithm of service selection, common genetic algorithm and random selection algorithm. The same simulation software and hardware environment is adopted. There are six service providers. One user selects and accesses service at a time and after thirty times selection, the statistics of influence to the load of service provider is made. Let the weight w_L of f1 be 0.9 and let the weight w_Q of f_2 be 0.1. The algorithm runs ten times continuously and the load balance average deviation of service provider in each time is calculated. See below for the calculation formula of average deviation (14):

$$A.D. = \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}| \qquad (14)$$

Figure 5 and Figure 6 show the comparison for load average deviations of CPU occupancy rates and memory utilization rates among adaptive genetic algorithm of service selection (AGA), common genetic selection (GA) and random selection algorithm (RA).

As shown by three comparison curves in Figure 5 and Figure 6, random selection algorithm achieves large value of load average deviation and strong jitter. The curve of common selection algorithm is nearly parallel to X axis and smoother than that of random selection algorithm,

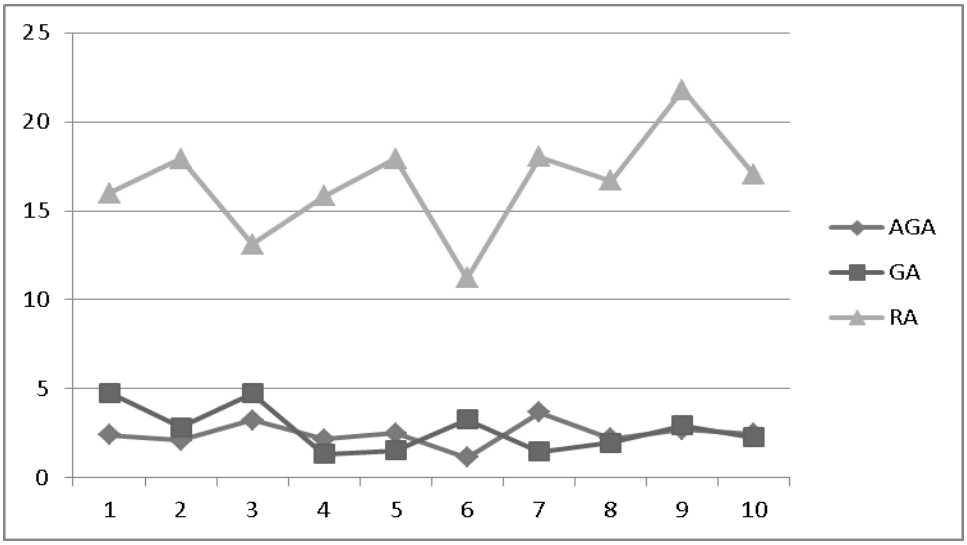

Figure 5. Comparison for load average deviations of CPU occupancy rates among AGA0GA and RA

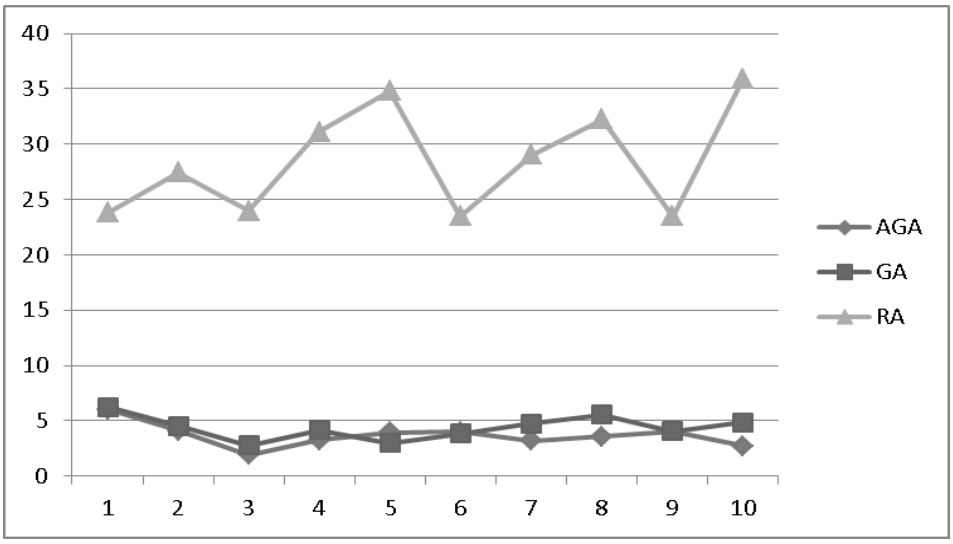

Figure 6. Comparison for load average deviations of memory utilization rates among AGA0GA and RA

which implies that compared with random selection algorithm, common selection algorithm has a smaller value, with the jittering degree of the curve substantially decreasing. However, compared with the previous two algorithms, adaptive genetic algorithm of service selection achieves a smaller value, with the curve nearly parallel to X axis. According to the average deviation operation for the load values of three algorithms, in terms of average deviation of CPU occupancy rate, adaptive algorithm of service selection is 0.54 smaller than common genetic algorithm and 1.55 smaller than stochastic selection algorithm, while in terms of the average deviation of memory utilization rate, adaptive algorithm of service selection is 0.88 smaller than common genetic algorithm and 3.4 smaller than stochastic selection algorithm. This suggests that adaptive genetic algorithm of service selection can work out the solution that approaches optimal solution compared with stochastic selection algorithm and common selection algorithm. The adaptive genetic algorithm of service selection designed in this paper shows highly stable optimization ability.

5. Conclusions

Based on the analysis about service selection model, this paper not only reveals the relationship among service provider, service and user with matrix, but also designs and realizes an adaptive algorithm of service selection according to service selection based on multi-objective constraints. By means of matrix encoding, this algorithm replicates optimal individuals of every generation to next generation through elitism selection strategy, as well as carries out adaptive adjustment for mutation probability and crossover probability with the change of individual fitness value to perform evolutionary operations, such as about selection, crossover and mutation. According to an array of simulation data, the adaptive genetic algorithm of service selection based on multi-objective constraints in this paper can effectively address the problems of service selection. Besides, off-line performance analysis and comparisons about random selection algorithm and traditional genetic algorithm prove that this algorithm shows good convergence and more stable optimization ability.

References

[1] Wei, Liu., Binwen, Huang. (2013). Design and simulation of an optimum service selection arithmetic in pervasive computing. *In*: Proceeding of the 5th International Conference on Computational and Information Sciences. Hubei, China, p. 9-12.

[2] Ye, Juan., Dobson, Simon., Nixon, Paddy. An Overview of Pervasive Computing Systems. *Microsystems*. 2009, 18, p. 3-17.

[3] Mobedpour, Delnavaz., Ding, Chen. (2013). User-centered design of a QoS-based web service selection system, *SOCA*, 7, p. 117-127.

[4] Yang, Kun., Galis, Alex., Chen Hsiao-Hwa. (2010). QoS-Aware Service Selection Algorithms for Pervasive Service Composition in Mobile Wireless Environments. *Mobile Networks and Applications*. 15 (4) 488-501.

[5] Srinivasa, K. G., Sridharan, Deepa Shenoy, K. P., Venugopal, K. R., Patnaik, Lalit M.. (2005). A Dynamic Migration Model for Self-adaptive Genetic Algorithms. Lecture Notes in Computer Science. 3578, p. 555-562.

[6] Tsai, Chih-Fong., Eberle, William., Chu, Chi-Yuan . (2013). Genetic algorithms in feature and instance selection, *Knowledge-Based Systems*, 39, p. 240-247.

[7] Baodong, Shao., Lifeng, Wang., Jianyun, Li., Heming, Cheng. (2011). Multi-objective optimization design of a micro-channel heat sink using adaptive genetic algorithm. *International Journal of Numerical Methods for Heat & Fluid Flow*, 21 (3) 353-364.

[8] Maa, Yue., Zhang, Chengwen. (2008). Quick convergence of genetic algorithm for QoS-driven web service selection, *Computer Networks*, 52 (5) 1093-1104.