International Journal of Computational Intelligence Systems, Vol. 7, Supplement 2 (July 2014), 26-34

A QoS-oriented Web service composition approach based on multi-population genetic algorithm for Internet of things

Qian Li
College of Management and Economics, Tianjin University, No.92, Weijin Road, Nankai District, Tianjin, 300072, China

Runliang Dou
College of Management and Economics, Tianjin University, No.92, Weijin Road, Nankai District, Tianjin, 300072, China

Fuzan Chen*
College of Management and Economics, Tianjin University, No.92, Weijin Road, Nankai District, Tianjin, 300072, China
E-mail: fzchen@tju.edu.cn

Guofang Nan

College of Management and Economics, Tianjin University, No.92, Weijin Road, Nankai District, Tianjin, 300072, China

Received 25 November 2013

Accepted 15 March 2014

Abstract

Internet of things (IoT) will create new opportunities to build applications that better integrate real-time state of the industry. With Web services accomplishing similar function proliferated, industrial enterprises have to choose appropriate Web services according Quality of Service (QoS) properties. It introduces the problem of QoS-oriented service composition (QSC). This study formulates the QSC problem as a multi-criteria goal programming (MCGP) model, and develops a multi-population genetic algorithm (MGA) to solve the model. MCGP not only automatically assigns high quality Web services to combine a composite service, but also finds non inferior composite services by relaxing QoS constraints to satisfy users' QoS requirements. Empirical comparisons demonstrate MGA outperforms to the GA. Moreover, the experiments indicate MGA is capable to solve the large-scale QSC problem in terms of efficiency and scalability.

Keywords: Internet of Things, Web service, Web service composition, Quality of service

*Corresponding author. E-mail addresses: fzchen@tju.edu.cn (Fuzan Chen)

1. Introduction

Nowadays, with forceful evolutionary of the Internet and the Web, the widespread deployment of spatially distributed devices, such as RFID tags, sensor and smart mobile machines, promotes the Internet of things (IoT). There is a trend that distributed devices and enable machines are able to interact and cooperate with others to reach common goals by IoT (Guinard et al., 2010). With IoT and Web service technologies, business conduction and information exchanging can be implemented more dynamic than before (Miorandi et al., 2012).

An increasing number of business processes are published as Web service across Web. Since the individual service somehow fails to satisfy the users' requirements, industrial organizations prefer to combine the existing simple individual Web service to a more complex composite service (Ko et al., 2008). Web service composition presents an opportunity to the rapid application development, service reuse and complex service consummation (Chen et al., 2006).

An example of alarm service is used to illustrate the idea of service composition. Suppose that a warehouse is equipped with different devices including temperature sensors, smoke detectors, infrared sensors, liquid immersion sensors, alarm bells, camera, scupper valves and water sprinklers. The alarm service is shown in Fig.1 where individual devices providing simple function can be combined to establish a more powerful alarm service. In this application, each device in the composite service is called a component service (or task) which can be bound to various Web services providing the same functionality but with different quality properties.

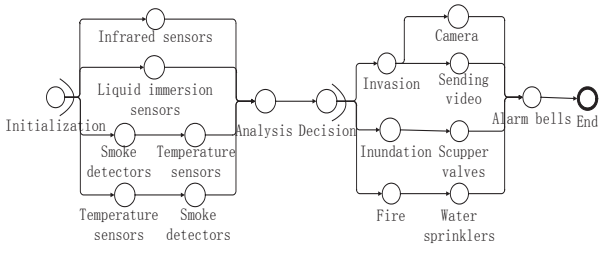


Fig.1. A composite service for the warehouse alarm

Generally, when a user submits a service request, overall QoS constraints can be referred along with, such as response time should be less than 2H and cost should be less than 10$. With the proliferation of Web service offering similar functionality, how to select an appropriate Web service to construct a composite service according to the non-functional properties, such as Quality of Service (QoS) has becomes a prominent issue (AllamehAmiri et al., 2013).

This paper addresses the QoS-oriented Web service composition (QSC) problem. The QSC problem is formulated to a multi-criteria goal programming (MCGP) model. In addition, a multi-population genetic algorithm (MGA) is developed to solve the presented MCGP model. Experiment results show that the proposed method is capable to find the optimal service composition for users' QoS constrains with low failure rate, especially when no Web service could exactly satisfy the users' QoS requirements.

The following of this paper is organized as follows. Section 2 introduces the related work. The MCGP model for QSC and MGA solving algorithm are presented in section 3 and section 4, respectively. Section 5 analyzes and discusses the experimental results. Finally, conclusions and future work are given in Section 6.

2. Related work

One of the main challenges for service composition is to choose appropriate service instances for a service composition schema representing the abstract functionalities of tasks and the workflows of composite service. Since the length of a composite service and the candidates for each task are expected to proliferate, the scale of possible service compositions with the identical functionality but different QoS performance should be in general exponentially enlarging. This leads to an optimization problem on QSC (Zeng et al., 2004).

With perspective of multiple QoS constraints, multi-criteria decision making model and mathematical goal programming models were used to analyze the QSC problem by Cui et al. (2011). However the weights and priorities of the QoS objectives for goal programming are respectively required in the non-preemptive and preemptive goal programming model. Furthermore, if all the goals cannot be simultaneously achieved, the users have to decide which goal should be sacrificed to continue the programming procedures. Moreover, Yu et al. (2007) modeled the QSC problem as a multidimension multichoice 0-1 knapsack problem (MMKP) and a multi-constrained optimal path (MCOP)

problem, respectively. However, they didn't consider the possible failure in finding a feasible solution to satisfy the users' QoS constrains.

Moreover, it is a NP-Complete problem to select the optimal composite service from numerous solutions to satisfy user's QoS constrains, which can't be solved within a reasonable time. Hence, many efforts of intelligent and heuristic algorithms are concentrated in QSC to find an approximate optimal solution. Particle swarm optimization (PSO) (Poli et al., 2007) has been widely applied for QSC problem (Poli et al., 2007). Additionally, Zhao et al. (2012) proposed an improved discrete immune particle swarm optimization algorithm (IDIPSO) which includes an improved local best first stage based on mathematical analysis and a perturbing global best stage along the global best particle. Furthermore, Hu et al. (2009) transformed the QSC problem into a QoS multi-objective model, and developed an improved particle swarm optimization algorithm (IPSOA), where satisfying results can be obtained with adaptive weight adjustment and non-uniform mutation strategies, to solve the multi-objective model. Moreover, Ko et al. (2008) proposed a QoS constraint satisfaction based method for QSC problem. They automatically assigned high quality Web service for each task in a composition schema with an optimal algorithm. The algorithm is characterized as a meta-heuristics combining tabu search and simulated annealing. However, their work fails to the scalability respecting to the growth of the number of Web services.

Though the approaches with intelligent algorithms improve the efficiency of Web service composition with QoS constraints, they will fail in the case where none of feasible composite services exactly satisfies all the QoS constrains. With this perspective, Lin et al. (2011) attempted a relaxable QoS-based service selection (RQSS) method to discover feasible Web services based on QoS requirements. The RQSS is to not only find non inferior composite services but also reduce the computation complexity. The method of RQSS attempts a worthy solution for the QSC problem.

In summary, researches endeavor to find best services to compose optimal composite service in industrial applications. It is still a vital task to develop efficient methods for handling the situation where no feasible solution can fulfill the overall QoS constraints. Aiming to this issue, this paper formulates QSC problem to a multi-criteria goal programming (MCGP) model to find a solution with smaller amount of constraints violation. Then we develop a multi-population genetic algorithm to solve the MCGP model.

3. **Multi-criteria goal programming model for QSC problem**

3.1 *QoS model for QSC*

In respect to various QoS attributes for Web services defined by W3C working group, we define three QoS properties as the quality evaluation criteria of Web service according to domain application of IoT.

- Execution time (t): The execution time of a service is the average time between when the request is sent from the user and when the server response is received.
- Reliability (r): The reliability of a service is the percentage that a service request is completed successfully. It is measured by the rate of number of successful executions over to total number of service invoked.
- Execution cost (c): The execution cost is the payment of executing a Web service.

A composite service consists of a set of logically connected component services (or tasks). Each task can be bound to different candidate service instances with identical functionality but different QoS values (Chen et al., 2006). The most widely used workflows in service composition include sequential (a), Loop (b), Parallel (c) and Switch (d) (Zhao et al., 2012), as shown in Fig. 2.

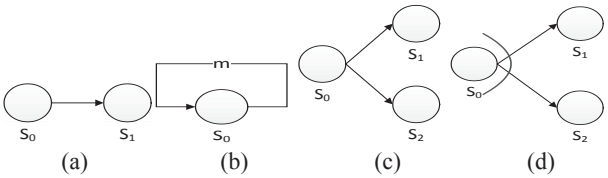


Fig.2. Workflows used in Web service composition

Accordingly, the QoS property values of composite service can be calculated with the following equations shown in Table 1.

Table 1. Aggregation function of QoS properties

QoS property	Sequential	Loop	Parallel	Switch
Execution time	$\sum_{i=1}^{n} t_i$	$\sum_{i=1}^{n} t_i$	$\max(t_i)$	$\sum_{i=1}^{n} p_i * t_i$
Reliability	$\prod_{i=1}^{n} r_i$	$\prod_{i=1}^{n} r_i$	$\prod_{i=1}^{n} r_i$	$\prod_{i=1}^{n} p_i * r_i$
Execution cost	$\sum_{i=1}^{n} c_i$	$\sum_{i=1}^{n} c_i$	$\sum_{i=1}^{n} c_i$	$\sum_{i=1}^{n} p_i * c_i$

Herein, p_i is the probability of executing each branch in the switch workflow.

As the scales of the QoS properties are quite different, QoS property values are normalized at first with Eq. (1).

$$Q_i' = \frac{Q_i - \overline{Q_i}}{\sigma_i} \tag{1}$$

In Eq. (1), σ_i is the standard deviation of each QoS property value, and $\overline{Q_i}$ is the mean.

Without loss of generality, Q_i represents the normalized QoS property value in the following parts of this paper.

3.2 *Multi-criteria goal programming model for QSC problem*

3.2.1 Goals for the programming model

Three objective functions can be derived in respect to the QoS properties defined in section 3.1.

- Execution time (T): It is expected to be as minimal as possible, where T_0 is the maximum overall execution time specified by user.

$$\min T = \text{Aggregation}(t_i),\ T \le T_0 \tag{2}$$

- Reliability (R): It is expected to be as maximal as possible, where R_0 is the minimum overall reliability specified by user.

$$\max R = \text{Aggregation}(t_i),\ R \ge R_0 \tag{3}$$

- Execution cost (C): It is expected to be minimized, where C_0 is the maximum overall execution cost specified by user.

$$\min C = \text{Aggregation}(c_i),\ C \le C_0 \tag{4}$$

Summarily, the three-dimensional objective optimization model with constraints is given as follows.

$$\begin{cases} F(\min T, \max R, \min C) \\ s.t. T \le T_0, R \ge R_0, C \le C_0 \end{cases} \tag{5}$$

3.2.2 *MCGP model of QSC problem*

We utilize multi-criteria programming (MCP) with a weighted sum model and multi-criteria goal programming (MCGP) (Cui et al., 2011) with a weighted sum model to formulate the QSC problem. MCP is a common model, while MCGP can be used in the case where no feasible solution can be discovered for the users' overall QoS requirements. The proposed approach works as follows.

(i) MCP: The value of QoS properties for each composite service can be aggregated by functions show in Table1. We use the weighted summary method to transform the multi-criteria programming goals into single objective. The MCP model for QSC with constraints is illustrated as follows.

$$\begin{cases} Min(w_1 \cdot T - w_2 \cdot R + w_3 \cdot C) \\ s.t. T \le T_0, R \ge R_0, C \le C_0 \end{cases} \tag{6}$$

Herein, W_i is the weight of each QoS property, $\sum_{i=1}^{m} W_i = 1$, m is the number of QoS properties.

(ii) MCGP: In the case that there is none of composite service exactly satisfying the user's QoS constraints, MCGP minimizes the QoS gap between Web service composition and user's requirements by means of relaxing constraints.

The difference of the actual QoS value larger than its constraint is denoted as d_i^+, while d_i^- represents the difference of the QoS value less than its constraint. They can be calculated Eq. (7) and Eq. (8), respectively, herein Q_i^0 is the constraint of QoS property.

$$d_i^+ = \begin{cases} Q_i - Q_i^0, Q_i > Q_i^0 \\ 0, Q_i < Q_i^0 \end{cases} \tag{7}$$

$$d_i^- = \begin{cases} 0, Q_i > Q_i^0 \\ Q_i - Q_i^0, Q_i < Q_i^0 \end{cases} \tag{8}$$

The QoS constraints can be modified to $T + d_1^+ - d_1^- = T_0$, $R + d_2^+ - d_2^- = R_0$ and $C + d_3^+ - d_3^- = C_0$, $d_i^+, d_i^- \ge 0$. Accordingly, MCGP model can be illustrated as follows.

$$\begin{cases} \text{Min}(w_1 \cdot T - w_2 \cdot R + w_3 \cdot C + w_4 \cdot d_1^+ + w_5 \cdot d_2^- + w_6 \cdot d_3^+) \\ T + d_1^+ + d_1^- = T_0 \\ R + d_2^+ + d_2^- = R_0 \\ C + d_3^+ + d_3^- = C_0 \\ d_i^+, d_i^- \ge 0, i = 1,2,3 \end{cases} \tag{9}$$

If there is no feasible solution, MCGP can be used to find non inferior solutions to relax the QoS constraints and minimize the violated quality value. MCP is a special case that d_i^+ and d_i^- are zero.

4. MGA algorithm for solving the MCGP model

GA is an evolutionary algorithm rapidly growing in the area of artificial algorithm (Goldberg, 2005). Generally, GA gives less chance of survival (or none at all) for the low fitting individuals by mimicking the principle of natural evolution, i.e. 'survival of the fittest'. However, the worst individuals may result in better offspring if they have chance to mate.

Along with such a sense, multi-population genetic algorithm (MGA) partitions individuals into several groups or sub-populations on the basis of fitness values. Individuals within the same community are enabled to mate with each other. If an individual produces a high fitting offspring, the offspring migrates from its original group to the suitable group with higher fitness value, and vice versa. Thus, all individuals in the population achieve the equal opportunities no matter they are of high fitness or low fitness. This allows MGA to maintain the diversity of population, and then ensures faster convergence (Siva Sathya et al., 2013).

4.1 *The framework of MGA*

The framework of MGA is illustrated in Fig.3. During the evolution, when an individual comes up with a better fitness, the offspring leaves its group and migrate to the group with similar fitness value to it. MGA is good at faster convergence as each sub-population evolves independent (Kojima et al., 2008).

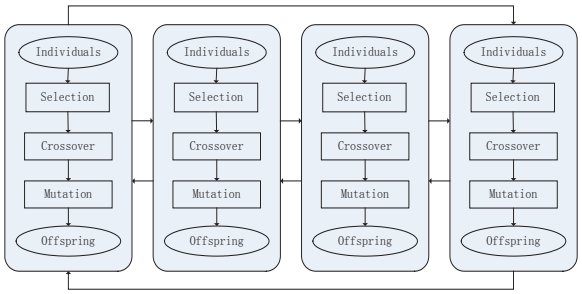


Fig.3. Framework of MGA

However, it is a pivotal task to specify judicious parameters or genetic strategies that heavily affect the performance of MGA, such as connection topology, migration method, migration policy and rate, and so on (Siva Sathya et al., 2009). The size of groups and the numbers of groups are fixed in the process of MGA. With respect to the dynamic migration of individuals, migration rate in MGA is influenced by the number of individuals that has improved their fitness. But connection topology, migration method and migration policy are not explicitly mentioned in current researches. In addition, MGA is usually more resistant to premature convergence. Consequently, we presented some effective genetic strategies to overcome these shortcomings.

4.2 *Fitness function*

Commonly, the fitness of individual (corresponding a candidate service composition generated by selecting certain service instance for each task) is evaluated by Eq.(10).

$$F = w_1 t - w_2 r + + w_3 c \tag{10}$$

Herein t, r and c is the aggregated values of response time, reliability and cost by aggregation functions in Table1, respectively; $0 < w_i < 1$ and $\sum_{i=1}^{3} w_i = 1$.

Particularly, in the case of no feasible solutions, the fitness function shown in Eq.(11) aims to minimize violations of QoS performance and the given QoS constrains for individuals.

$$F = w_1 t - w_2 r + w_3 c + w_4 d_1^+ + w_5 d_2^- + w_6 d_3^+ \tag{11}$$

Herein, $0 < w_i < 1$, $\sum_{i=1}^{6} w_i = 1$, $d_1^+ = t - T_0$, $d_2^- = R_0 - r$ and $d_3^+ = c - C_0$.

4.3 The design of MGA

(1) Encoding: A chromosome or individual represents a possible composite service where each task has been appointed a certain service instance. Assuming that all the candidate services for each task have been numbered with integers, the chromosome is represented by a $1 \times n$ integer vector, where n is the number of task included in composite service, and the value of each position in the vector indicates the order of candidate service for the task. For example, a vector (2,3,5,3,9,1) indicates there are 6 tasks in the schema of composite service, where task No.1 chooses its second candidate service, task No.2 chooses its third candidate service, and so on.

(2) Genetic Operators: Individuals are chosen into mating pool to produce offspring by using crossover and mutation operators. The roulette wheel selection with simple elitism where fitter individuals will tend to have a better probability of survival is used to select pairs to form the mating pool for the next generation. We adopt the single-point crossover operator which exhibits the maximum positional bias and the lowest disruption rates (Jaeger et al., 2005). Offspring can be generated by exchanging segment of parents at the selected crossover point (Mardukhi et al., 2013). The mutations operator aims to maintain diversity of individuals, i.e. genes of individual are randomly selected and their values are converted with a certain probability.

The main process of MGA is shown in Table 2.

Table 2 Process of MGA algorithm

```
//P: the population of current generation
//n: the number of individuals
//m: the number of groups
//t: the number of current generation
//T: the maximum number of generations
Begin
    Initialize P(n)
    t=0
    while (t<=T) do
        Calculate Fitness for P(n) using Eq.(10)
        if noFeasible(P) then
          Calculate Fitness for P(n) using Eq.(11)
        end if
        Sort P(n)
        for i = 1 to n/m
            Select P(i)
            Crossover P(i)
            Mutation P(i)
        end for
        for i = (n/m)+1 to (n/m)*2
            Select P(i)
            Crossover P(i)
            Mutation P(i)
        end for
        …
        for i = n-(m-1)*(n/m) to n
            Select P(i)
            Crossover P(i)
            Mutation P(i)
        end for
        t = t+1
    end while
end
```

4.4 Advantages of MGA

Firstly, the strategy of multiple populations helps to maintain the diversity of population, and thus address the precocity problem. The low fit individuals are given the same chance to mate. This may result in better offspring which migrating to a better group (Siva Sathya et al., 2010). Moreover, such strategy generally helps the algorithm to converge faster.

Secondly, MGA is easy to be parallelized because the entire population is divided into several sub-populations. If the population size is very large, sub-populations can independently evolve with different processors.

5. Experimental investigations

In this section, we investigate the performance of proposed MGA for large-scale QSC problem by comparing to the genetic algorithm (GA).

5.1 *Experimental setup*

Algorithms were coded with Matlab 2010a, and ran on an Intel Pentium Dual Core E5800 3.2 Ghz, 2 GB RAM desktop PC with Window XP.

In order to evaluate the performance of the methods, we illustrated composite service schemas containing different number logically connected tasks with general workflows. Each task varies in the number of candidate services.

Additionally, QoS property values for candidate services were synthetically generated referred to most of QSC researches (Pastrana et al., 2011). For each task, values of QoS properties were randomly generated with the normal distribution, and the same parameters as Pastrana et al. (2011) were used. Moreover, constraint of each QoS property is set by Eq.(12).

$$Q_i = n * Q_i^{\max} * CF, i = 1,2,3 \tag{12}$$

Herein, n is the number of tasks in a composite service, $Q_i^{\max}$ is the maximum value of a QoS property, CF is the factor ranging [0,1], which is used to adjust the impact of each QoS property. In the generation of the QoS constraints, all the QoS metrics are assumed as the additive metrics. Since the reliability is multiplicative metric, the constraint of reliability is set to $\left(Q_i^{\max}\right)^{-n*CF}$. CF is 0.8 in this study. All the QoS constraints are shown in Table 3.

Table 3 Range and constraint of QoS property

QoS property	Range	Constraint
Execution time	[0.1,3]	$n*3*CF$
Reliability	[0.7,0.9]	0.9^{-n*CF}
Execution cost	[1,100]	$n*100*CF$

In order to present comparisons of MGA and GA, MGA and GA used the same genetic operators and parameters. However, the strategy of multiple populations was used in MGA. Probabilities of crossover and mutation are 0.7 and 0.05, respectively. Population size N is 80. The number of sub-populations in MGA is 4, i.e. each sub-population includes 20 individuals. Moreover, we run GA and MGA 30 times independently, and use the averaged values for performance evaluation.

5.2 Fitness values investigation

Fitness value is an important guideline for GA performance. The less value indicates the better performance. Consequently, we compared the best fitness obtained by GA and MGA for various scales of QSC problem, as shown in Table 4. The number of tasks in the composition schema is n and the number of candidate services per task is m. As shown in Table 4, MGA always got less fitness than that of GA for any scale of QSC problem. It indicates that MGA can find more suitable composite service than GA.

Table 4 Best Fitness values of GA and MGA

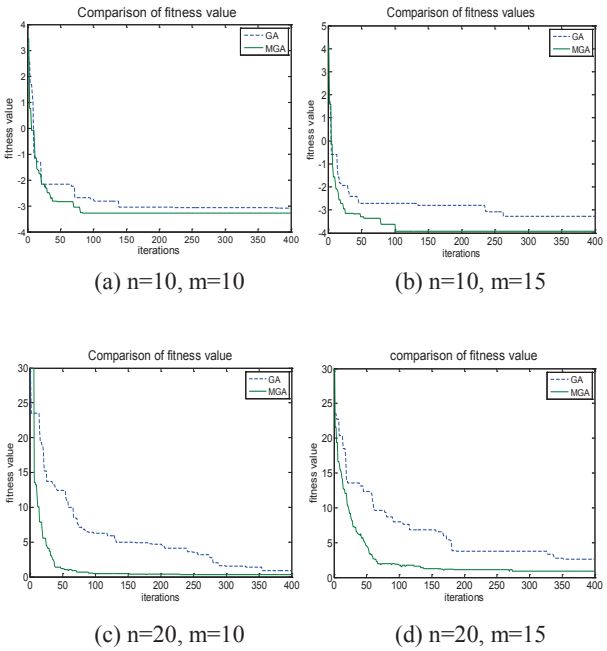
(a) Comparisons with n=10 varying m

Num. Of candidate services	10	15	20
GA	-21.88	-25.67	-29.21
MGA	-23.90	-28.12	-31.81

(b) Comparisons with m=15 varying n

Num. Of tasks	10	15	20
GA	-25.67	-32.38	-6.44
MGA	-28.12	-35.61	-11.51

Additionally, Fig. 4 shows the comparison of GA and MGA for various scales of service compositions in terms of fitness for each generation. Case (a) and (b) have the same number of tasks but various candidate services per task (10 and 15); and (c) and (d) vary to tasks (10 and 15) but each task has the same number candidate services.



(a) n=10, m=10 (b) n=10, m=15

(c) n=20, m=10 (d) n=20, m=15

Fig.4. Fitness values comparisons of GA and MGA

Fig. 4 indicates that MGA outperforms GA greatly. The best fitness values of MGA surpass GA quickly for various scales of service compositions. The results demonstrate the powerful search ability of MGA for promising solutions. It also can be seen that MGA converges to a better minimal fitness value than GA. According to the results, it is concluded that MGA works well for large-scale QSC problem. It is capable to find better solution even though either the task number or the candidate services per task increases.

5.3 Time cost investigation

In order to evaluate the efficiency and scalability, we compared the computation time of MGA and GA with various scales of problem, i.e. there are approximate m^n possible composite services with different QoS performances. Table 5(a) shows the execution times of cases where n=10 and m ranges from 10 to 20 with a step of 5, and Table 5(b) shows the execution times of cases where m=10 and n ranges from 10 to 20 with a step of 5.

Table 5. Time comparisons of GA and MGA

(a) Comparisons with n=10 varying m

Num. Of candidate services	10	15	20
GA	2.80	2.80	2.83
MGA	2.14	2.16	2.16

(b) Comparisons with m=15 varying n

Num. Of tasks	10	15	20
GA	2.80	3.53	4.00
MGA	2.16	2.84	3.41

The results of Table 5 shows that the superiority of MGA to GA is more and more prominent as either task number or number of candidate services per task grows. This indicates that MGA has better efficiency and scalability than GA, especially when the scale of candidate composite services becomes larger and larger. Summarily, MGA is an effective and scalable method for large scale QSC problem, which can find appreciate composite service to satisfy the user's QoS constrains within a reasonable time. We believe it is essential for the development of IOT and other distributed applications.

6. Conclusion

It can be expected that more and more Web services offering the identical functionality could be published across the Web. QoS properties will become the import aspect for service composition. This paper addresses the issue of large scale QSC problem.

We formulate QSC problem to a MCGP model to discover the appropriate composite service fulfilling the users' QoS constraints. Especially, when none of composite service could strictly satisfy the user's overall QoS constraints, it can also recommend feasible solutions by relaxing the QoS constraints. Moreover, an effective MGA is developed to solving the MCGP model. Experimental results validate the excellent performance of MGA in terms of powerful searching ability and excellent convergence ability.

As a future work, we suggest more comprehensive QoS measurement and effective algorithm to solve the optimization model. Dynamic Web service composition with QoS constrains may be another area we will extend.

Acknowledgements

The work was supported by the National Science Fund for Distinguished Young Scholars of China (No. 70925005) and the General Program of the National Science Foundation of China (No.71101103, No. 71201115 and No.71271148).

References

AllamehAmiri, M., Derhami, V., Ghasemzadeh, M., 2013. *QoS-Based web service composition based on genetic algorithm*. Journal of AI and Data Mining 1(2), 63-73.

Chen, Y., Zhou, L., Zhang, D., 2006. *Ontology-Supported Web Service Composition: An Approach to Service-Oriented Knowledge Management in Corporate Services*. Journal of Database Management 17(1), 67-84.

Cui, L.Y., Kumara, S., Lee, D., 2011. *Scenario Analysis of Web Service Composition based on Multi-Criteria Mathematical Goal Programming*. Service Science 3(4), 280-303.

Goldberg, D.E., 2005. *Genetic Algorithms in search, Optimization, and Machine Learning*, ninth ed., Pearson Education.

Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., Savio, D., 2010. *Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services*. IEEE Transactions on Services Computing 3(3), 223-235.

Hu, C.H., Chen, X.H. Liang, X.M., 2009. *Dynamic services selection algorithm in Web services composition supporting cross-enterprise collaboration*, Journal of Central South University of Technology 2, 269-274.

Jaeger, M.C., Muhl, G., Golze, S., 2005. *QoS-aware composition of web services: an evaluation of selection algorithms*. Lecture Notes in Computer Science 3760, 646-661.

Ko, J.M., Kim, C.O., Kwon, I., 2008. *Quality-of-service oriented web service composition algorithm and planning architecture*. Journal of System and Software 81(11), 2079-2090.

Kojima, K., Ishigame, M., Chakraborthy, G., Hatsuo, H., Makino, S., 2008. *Asynchronous parallel distributed genetic algorithm with elite migration*. International Journal of Computational Intelligence 4(2), 105-111.

Lin, C.F., Sheu, R.K., Chang, Y. S., Yuan, S.M., 2011. *A relaxable service selection algorithm for QoS-based web service composition*. Information and Software Technology 53(12), 1370-1381.

Mardukhi, F., NematBakhsh, N., Zamanifar, K., Barati, A., 2013. *QoS decomposition for service composition using genetic algorithm*. Applied Soft Computing 13(7), 3409-3421.

Miorandi, D., Sicari, S., Pellegrini, R.D., Chlamtac, I., 2012. *Internet of things: Vision, applications and research challenges*. Ad Hoc Networks 10(7), 1497-1516.

Poli, R., Kennedy, J., Blackwell, T., 2007. *Particle swarm optimization, Swarm Intelligence* 1(1), 33-57.

Siva Sathya, S., Kuppuswami, S., Syam Badu, K., 2009. *Nomadic genetic algorithm for multiple sequence aligement*. International Journal of Adaptive and Innovative Systems 1(1), 44-59.

Siva Sathya, S., Kuppuswami, S., 2010. *Analyzing the migration effects in nomadic genetic algorithm*. International Journal of Adaptive and Innovative Systems 1(2), 158-170.

Siva Sathya, S., Radhika, M.V., 2013. *Convergence of nomadic genetic algorithm on benchmark mathematical functions*. Applied Soft Computing 13(5), 2759-2766.

Yu, T., Zhang, Y., Lin, K.J., 2007. *Efficient algorithms for web services selection with End-to-End QoS constraints*. ACM Transactions on the Web 1(1), Article 6.

Zeng, L.Z., Bouguettaya, B., Ngu, A.H.H., Jayant, K., Henry, C.,2004. *QoS-aware middle ware for Web Services composition.* IEEE Transactions on Software Engineering 30(5), 311-327.

Zhao, X.C., Song, B.Q., Huang, P.Y., Wen, Z.C., Weng, J.L., Fan, Y., 2012. *An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition*, Applied Soft Computing 12(8), 2208-2216.

Pastrana, J.L., Pimentel, E., Katrib, M., 2011. *QoS-enabled and self –adaptive connectors for Web services composition and coordination*, Computer Language, Systems & Structures 37(1), 2-23.