# Genetic Algorithm for Context-Aware Service Composition Based on Context Space Model

Zhichao Zhang*, Shaoqiu Zheng*†, Weiping Li‡, Ying Tan*†, Zhonghai Wu‡, Wei Tan§
* School of Electronics Engineering and Computer Science, Peking University
† Key Laboratory of Machine Perception (Ministry of Education), Peking University, Beijing, China
‡ School of Software and Microelectronics, Peking University
§ IBM T. J. Watson Research Center, USA
{z.zc,zhengshaoqiu}@pku.edu.cn, wpli@ss.pku.edu.cn, ytan@pku.edu.cn, wuzh@ss.pku.edu.cn, wtan@us.ibm.com

***Abstract*—The emergence of Web services has changed the Internet a lot, and greatly facilitated the development of service based software systems. How to select appropriate services and compose them according to given context to satisfy a user's requirement is a big challenge. This paper proposes a novel Genetic Algorithm (GA) method to synthesis web services in a context-aware environment. We first present a context space model to illustrate both contexts and services in a formal way; we utilize GA to compose context-aware services according to users' preference. We transform the problem of service composition to a multi-objective optimization problem. To resolve the conflict and dependencies among services in GA process, we propose a service similarity tree (SST) model to measure the similarity between services. Finally, we design a simulation experiment to evaluate our method. The experiment result shows that our method is a promising one to solve service composition problem in a context-aware environment.**

***Keywords*-Web Services; Context aware; Genetic Algorithm;**

## I. Genetic algorithm based service composition

For the advantage in solving combinational problem [1], Genetic Algorithm is chosen for context aware service composition solution. We model the candidate services as a 2-tuples of $< S, R >$, where $S$ represents the services and $R$ indicates the inter-relationships among the services and $R = R_c \bigcup R_d$, where $R_c$(or $R_d$) represent the confliction(or dependencies) rules with the form $(S_i, S_j)$ meaning that $S_i$ has confliction(dependencies) relationship with $S_j$.

Given a original situation $< c_1, c_2, ..., c_m >$ and a target situation $< c_{1g}, c_{2g}, ..., c_{mg} >$ described by Context Space Model [2], the proposed algorithm is designed to use a GA approach to automatically produce a sequence of web services, which can be operated successively on the original situation to obtain the goal situation.

As the candidate service set may be very large, before executing the algorithm, we should filtering unrelated services from the service set that do not affect the context attribute of the target.

### A. Encoding

In our algorithm, each chromosome represents a service selection sequence that can change the context space to the target situation. In the sequence, each service number N denotes the $N$-th service in the service set. Through the invocation of these services, one context attribute can be changed many times and eventually gets to the destined value. The length of the chromosome denotes the number of services that can help achieve the goal situation. For example, in Figure 1, the sequence denotes that service $1, 7, 9, ..., 3, 4, 2$ are selected to change current situation to destined situation.

### B. Fitness function

For each chromosome, we need to calculate the fitness value. To provide appropriate service sequence for different users, we define the fitness as a multiple objective function. Two objectives are used to measure the fitness of the chromosome. One is the distance between the result situation caused by service sequence and the target situation. The other is the length of service sequence.

Both of them should be as smaller as better. We have a parameter $\lambda$ to adjust the weight of the two objectives. The fitness function is defined as bellow:

$$F = (1 - \lambda)|\vec{m} - \vec{n}| + \lambda S_I \quad (1)$$

- $\vec{m}$, $\vec{n}$ represent the initial situation and target situation respectively. $|\vec{m} - \vec{n}| = \sum_{i=1}^{N} |m_i - n_i|$. $N$ represents the number of dimensions of the context space.
- $S_L$ represents the length of the service sequence
- $\lambda$ is a parameter and $0 \leq \lambda \leq 1$.

### C. Genetic Operators

Crossover operator and mutation operator will be used at each iteration of the genetic process. As conflicts and dependencies may exist among services [3], we need to deal with the conflicts and dependencies. For example, after crossover of parent P1 and P2, we get child C1 and C2. We find that service 3 and service 5 has conflicts shown in Figure 2. Also conflicts may happen in mutation as shown in Figure 3.

We design two functions, $genectic\_check()$ and $conflict\_resove()$, to respectively check whether the genetic operation is legal and resolve the confliction when the genetic operation is illegal.

In order to solve the confliction, we need to design a policy to adjust this genetic operation. As services have inter-relationship among themselves, we need to find a best


Fig. 1. Chromosome encoding

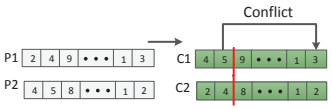



Fig. 2. Crossover and conflict

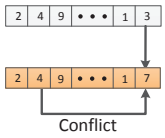



Fig. 3. Mutation and conflict

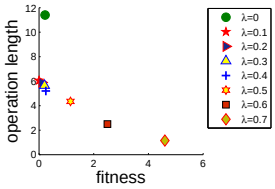



Fig. 4. Results with different $\lambda$

alternative which has the highest similarity with the conflict one. We use $S_i =< i_1, i_2, ..., i_m >$ to express a service which means the service $S_i$ can change the $i_j$-th $(1 \leq j \leq m)$ context attribute of the context space. We use an example to show how to deal with the confliction.

Given five services, $S_1 = (1, 2)$, $S_2 = (2, 3)$, $S_3 = (1, 2, 4)$, $S_4 = (1, 5)$, $S_5 = (1, 3, 5)$. We can build a forest with three trees according to the context inclusion relationship among the services . When confliction happens, the most similar service will be chosen. We design a similarity measurement function to find out the best alternative to the conflict one.

$$sim(S_i, S_j) = \begin{cases} \frac{1}{|S_j|-|S_i|} & S_i \subseteq S_j \\ \frac{|common\{S_i, S_j\}|}{|S_j|} & S_i \cap S_j \neq \emptyset \\ 1 & S_i = S_j \\ 0 & other \end{cases} \quad (2)$$

In formula(2), $sim(S_i, S_j)$ presents the similarity between the two services. $|S_i|$ and $|S_j|$ respectively represents the number of context attributes that the service can change, while $|common\{S_i, S_j\}|$ denotes the number of shared context attributes between $S_i$ and $S_j$. We require that $|S_j| \geq |S_i|$ when using this formula.

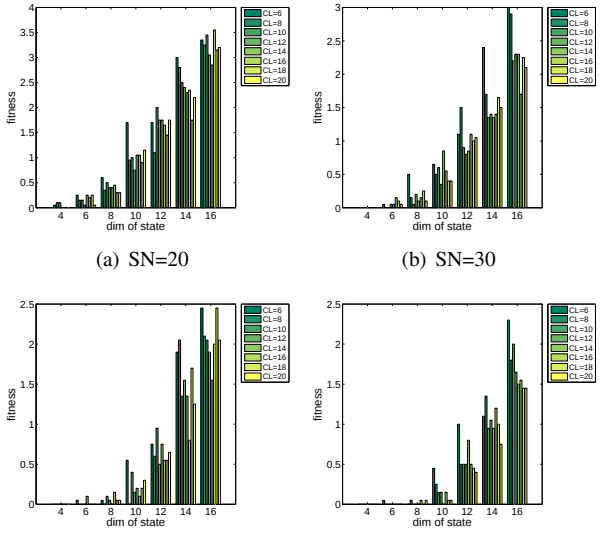
## II. Experiments and Evaluation

In our experiment, four parameters are included in the simulation, fitness value, number of dimensions in context space (dim_of_state), the length of chromosome (CL) and the number of services (service_num).

In the first experiment, shown in Figure 5 with service number (SN) is set to from 20 to 50, we find that: (1) With the increase of the service set, the fitness value become better and better.(2) The best CL should not be too longer or shorter, the fitness value is better when CL has value between 12 and 16. (3) When the number of dim_of_state increase, the fitness become worse.

In the second experiment, we involve $\lambda$ to evaluate the effect on these two objectives, fitness value and CL.To balance the two objectives, we use Pareto Front to find the best $\lambda$ value.Here, we present an example, assume that the parameters are SN=50, CL=14, dim_of_state=10, then we conducted the experiments with different $\lambda$ from 0.1 to 0.7. As shown in Figure 4, under these parameters, the proposed method with $\lambda = 0.1$ gains the highest performance.

From the first experiment result, when the service set increase bigger, the fitness value become better which indicates that the algorithm has more choices to choose services to achieve the goal context space. The CL of initial population in




(a) SN=20 (b) SN=30

(c) SN=40 (d) SN=50

Fig. 5. Experimental results with $\lambda = 0.1$.

GA algorithm cannot be too long or short, it looks better when the length has moderate value. The reason of this phenomenon is likely lying on that if the CL is too short, obviously, it cannot satisfy the goal context space with multiple dimensions; if the CL is too long, the complexity increases a lot to achieve the goal situation. We have no idea about whether there exist relationship between the dim_of_state and CL. With respect to the dim_of_state, when it scales up, the fitness value become larger which means the algorithm cannot achieve the goal well.

## Acknowledgment

Research work in this paper is partial supported by the Danish Strategic Research Council (Grant NO.2106-08-0046) and is partial supported by the National Natural Science Foundation of China (Grant NO.61033005, Grant NO.61170057, Grant NO.60875080).

## References

[1] Y. Vanrompay, P. Rigole, and Y. Berbers, "Genetic algorithm-based optimization of service composition and deployment," in The 3rd International workshop on Services integration in pervasive environments. ACM, 2008, pp. 13–18.

[2] Z. Zhang, W. Li, Z. Wu, and W. Tan, "Towards an automata-based semantic web services composition method in context-aware environment," in SCC 2012. IEEE, 2012, pp. 320–327.

[3] M. Tang and L. Ai, "A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition," in CEC 2010. IEEE, 2010, pp. 1–8.