一种支持广义服务组合的遗传算法①

王显志　王忠杰　徐晓飞　莫　同　(哈尔滨工业大学 计算机科学与技术学院 黑龙江 哈尔滨 150001)

A Genetic Algorithm for Generalized Real-Life Service Composition

Xianzhi Wang, Zhongjie Wang, Xiaofei Xu, Tong Mo

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001)

Abstract: In the last decade, selection and composition of Web services have drawn increasing attentions. However, real-life services are not only web services but complicated eco-systems composed of various service elements like human, resources, environment, etc, and existing service composition methods cannot be directly applied to such real-life service composition scenarios. In this paper, we propose a conceptual model for generalized real-life service composition. In this model, service behaviors are abstracted as service components and uniformly described by XML. Various service requirements raised by customers are completely listed and classified. Based on these works, a genetic algorithm for real-life service composition is presented to select the best matching service components and compose them together. The algorithm's effectiveness of obtaining optimal solution is proved by a prototype system.

Key words: real-life service; service component; classification of service requirements; service composition; genetic algorithm

1 Introduction

Modern service system is a complicated eco-system composed of not only software, but also hardware, human, resource and environment, opposed to web service domain. Service composition in this sense is to combine distributed service resources and service capability among different organizations following certain rules and agreements, thus provides customer with integrated service in multiple modern manners. On the one hand, by the specialization, automatic classification, linking, and integration of service sections, customer is confronted with a huge and integral virtual resource, which not only brings forth enjoyment and experience that customers have never captured before, but also relieves the disadvantageous situation of low customer satisfaction caused by limited resources and service methods of single organization; on the other hand, through the integration of service capability all-round, novel service modes is promising to emerge, thus obtains added value impossible for previous single service provider, which achieves more extensive share of information, service resource and service capability.

Traditional composition of service is done manually, i.e., workers of service organizations choose service providers for each section of business process according to requirements of customers, then communicate with this providers and also the providers communicate with each other by means of phone, email, etc. Such approach proved to be low efficient as well as under low possibility of finding satisfactory composite results under circumstances concerning numerous resources, so we anticipate this work be done in an automatic manner.

① Supported by the National Natural Science Foundation of China under Grant Nos.60803091,60673025

Component-based software development (CBSD) settles a favorable foundation for automatic service composition. To rapidly represent service systems which have more complicated processes and much more resources, and to make agile reactions to more frequently altered service requirements, we encapsulate all relevant resources and service behaviors/processes into reusable logical units, each called a Service component (SC) [1]. SC provides to the outside with service behavior of a larger granularity and complicated service systems can be established by selecting and compositing these pre-defined SCs.

For software in service system, we encapsulate each function into SC for retrieval and invocation, while for activities performed by human or machine, SC virtualizes them into software functions. The abstract of real life service into SC not only facilitates service composition in automatic way, but also creates software-form interfaces for human/machine interfered operations in real life service system.

After describing service as SC, service composition turns into a problem of service components matchmaking, i.e., how to select a set of customer-satisfied SCs from SC repository according to their historical quality and Reputation to turn abstract service process into a concrete one that can be run directly.

In this paper, we take customer requirements classification as the main concern, and explore automatic generalized service composition aiming to improve customer satisfaction: Section 2 presents a conceptual service composition model under the background of modern service industry; in Section 3 and 4, methods for describing real life service as SC and customer requirements classification is discussed respectively; then in Section 5, service composition algorithm base on genetic algorithm(GA) is presented and experiments done is given in Section 6; finally is the conclusion and view for the future.

2 Conceptual Model of Real-Life Service Composition

The basic structure of service composition (as showed in Fig.1) consists three elements: input, output and SC matchmaking engine.

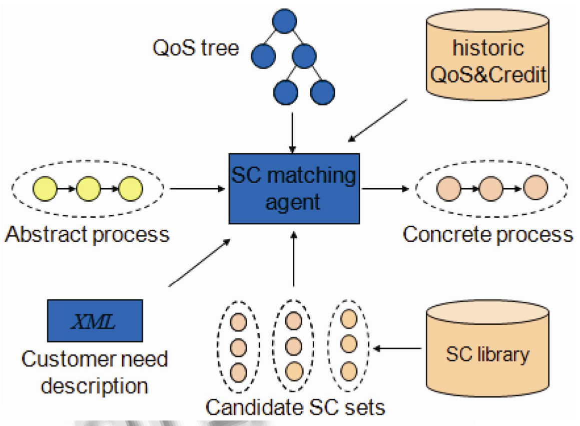

Fig.1 Conceptual model for service composition

2.1 Input

(1) Abstract service process

Web service uses BPEL4WS[2] and BPMN[3] to realize Orchestration and Chorography respectively, and semantic web service describes process following OWL-S[4] specification. But in real life, service usually sees human activity as an integral part of its process, so we describe abstract process using BPEL4People[5] specification with binding sections omitted, i.e., for each step of the process, only type of its function is specified instead of SC.

(2) Description of customer requirements

Ref.[6] suggests acquiring requirement based on ontology, i.e., to guide customer to describe realistic system comprehensively using enterprise ontology and domain ontology so as to quickly acquire and accurately express customer requirements in the development process of information system; Ref.[7] propose a mapping between requirement and resource to solve the problem of system requirement description in software requirement analysis. However, current solutions for require ment acquisition in software domain can hardly be applied to service system, which possesses its own processes. So description of customer requirements should be specially defined. This paper reduces customer requirements in service system into several formal and reusable patterns and adopts XML for description. More detail will be found in Section 2.

(3) Candidate SC sets

Ref.[8] uses XML to describe SC library, the result is each selection will lead to a traverse among all SCs. In reality, SC library are designed not for one system alone, so we organize SCs to be a two-tier architecture, i.e., classify SCs first by service domain, and then by function type. Because each step in abstract process has a predefined function type, a corresponding group of SCs can easily be found as candidates in the library.

(4) QoS tree

Ref.[9] extends WSDL[10] to describe QoS information of SC; Ref.[11] makes extension to Service Profile part of OWL-S to achieve the same goal. QoS of service in real life may be a treelike architecture including basic data, QoS dimensions, and QoS indicators. For example, SMDA[1] depicts QoS in five dimensions: price & cost, time & efficiency, service content, resource & condition, risk & credit, each can be subdivided into a set of QoS indicators.

This paper uses QoS tree to describe the classification and hierarchy of QoS. Each node of the tree comprises calculation formula and basic data, while basic data contains references to historic QoS and credit repository which will be shown in (5) part of this section and the formula takes the effect of obtaining QoS value of certain level from calculation on basic data or low level value. SCs of different functions may have different sets of QoS attributes, so they make reference to different QoS tree individuals. Moreover, each QoS attribute of SC correspond to a node in the referred QoS tree, what creates further flexibility for SC QoS definition.

(5) Historic QoS and credit repository of SC

All attributes and value of SC can be obtained by two steps: firstly, find all QoS attributes of SC in QoS tree (see this section (4)) according to its function type, then retrieve the historic QoS and credit repository using identifier and attribute name of SC to find the corresponding record that contains value.

The historic QoS or Credit value is the weighted average of former records. Usually, we apply Normal distribution to give one record the value of a farther distance to present a weaker weight. For example, (-258, 258) occupies 0.99 of the whole area of. In we take 259 points uniformly inside (0, 258), then each height represents the weight of each single day in 0~258 days. After normalized, we get to historic value.

2.2 SC Matchmaking Engine

We view the problem of service matching, which is a detailed solution of service composition, as combinatorial optimization problem with complex constraints. As a stochastic search method which gets inspirations from the mechanism of natural selection and heredity, GA's strategy of population search and characteristics of information transmission make it showing incomparable performance to other traditional approaches in combinatorial optimization[12]. Ref.[13] designs a multi-objective GA to find a set of pareto optimal solutions as the globally optimized fulfillment for dynamic service selection; Ref.[14] takes the possibility of high fitness in next generation into account, and defines recessive fitness function as part of the assessment to individuals. The above methods all take advantage of GA but they are not applicable for rich and personalized customer requirements much, which could be significant for service system. This paper modifies GA to be customer requirements oriented so that it transforms into a SC matchmaking engine in accordance with real life service system feature.

2.3 Output

We adopt BPEL4People specification to depict the binding result due to the same reason with section 1.1(1).The result of SC matchmaking is that every step in service process is designated with a estimated start time for execution and filled with one concrete SC.

3 Component-Based Description of Real-life Services

Service process in real life is a flow of a series of service behavior and Service behavior can be encapsulated into SC, while SC may contains detailed processes, so SC and service process have a mutual inclusion relationship. Component-based description of realistic service includes two parts: component-based description of service behavior and expression of service process.

3.1 Component-based description of service behavior

We define SC in XML format, and the definition contains two aspects:

(1) As the query index for service behavior selection, we describe the basic information, state of usage, actor (including the provider) of the behavior, and the behavior itself in SC definition.

Actor of the service behavior can be human, machine, software or some of them. For example, actor of transportation behavior is a group formed by a driver and a chuck. Service behavior itself may contain several QoS levels conformed to SLA[15] or other criteria. A simple example is that there can be two typical QoS levels in land-transportation behavior, i.e., ordinary transportation and damping transportation, depending on type of the goods to be transported. Fig.2 shows the description structure of SC.

Description Structure	Annotations
Service Component	Service Component
|--Static Pattern	|-- stands for section 3.1(1) content
|--isBusy	|--current state: is busy or not
|--Basic Information	|--Basic Information
|--Actor	|--Actor of the behavior
|--Provider	|-- service provider of SC
|--ActorGroup	|-- real actors form a group
|--groupName	|--groupName
|--ActorInformation(multiple)	|--group may contains many actors
|--AvailableTime (multiple)	|-- when SC is available
|--SupportiveResource(multiple)	|--the resources needed for behavior
|--Behavior	|--Behavior
|--FunctionType	|--type of the SC function
|--Input	|--input of behavior
|--InputResource(multiple)	|-- the resources SC needs
|--InputInformation(multiple)	|--the infomation SC needs
|--Output	|--output of behavior
|--OutputResource(multiple)	|-- tangible output of behavior
|--OutputInfo (multiple)	|--intangible output of behavior
|--Levels(multiple)	|-- behavior may has many levels
|--levelName	|--levelName
|--levelDescription	|--levelDescription
|--SupportResource(multiple)/	|--special resource of this level
|--QoSTreeID	|-reference to SC QoS tree
|--processAddress	|-- the reference to process file
|--Dynamic Pattern (omitted)	|-- stands for section 3.1(2) content

Fig.2　Description structure of SC

(2) As the call interface for SC execution, description of SC includes the generic and specialized interfaces of the behavior, task list, and log. Calls for behaviors that need human/hardware interference can be implemented referring to WS-HumanTask specification[16], which virtualizes human activity into software activity.

3.2 Expression of service process

Similar to web service composition, service behavior of SC may be a process composed of many other behaviors. We adopt non-binding information BPEL4People specification to describe process that having human-machine interaction, and store this description in a separate file with the same name to the SC, so we just make reference to this file in SC definition in section 3.1(1).

4　Classifications and Descriptions of Customer Requirements

Typical Customer requirements includes requirement for bearing time, price, reliability, and credit of the service provider. Besides, it may also include requirement for convenience of procurement, response time, service period, punctuality, consistence, compensation, success rate, and other factors depending on different service domains.

Only customer requirements are finely presented, can program comprehend customer goal in a concrete way with operability, and then proceed to get a continuously improving composition result during SC match process by comparing temporal real result with the customer expected one. Service system comprises a lot of interaction and customer requirements, and is often complicated, which can hardly be described one by one, so we think about depicting user requirements in a formal and reusable manner, that why we do classification. We summarize several circumstances that most frequently occur in customer requirements to form patterns for automatic program processing, and then define priority types to describe customer emphasis.

4.1 Customer requirements classification

Customer puts forward requirements from either local or global angle, or even from a fragment of process or discontinuous steps of the process view. Some usual customer requirements types and cases in ocean logistic service are listed in Table 1.

4.2 Customer requirements description

Customer requirements is reflected in service process by setting constraints on it, that's why we can summarize customer requirements into several frequently used constraint patterns, each of which can be about the provider, actor, or QoS metrics of the service. Basically, there are three constraints patterns: basic-type constraint, compare-type constraint, and rule-type constraint.

4.2.1 Basic-type constraint

This type is most frequently used, it's a pair: Basic (Function, Condition). Function is a set of function types, and Condition is a triple: Condition (Attribute, Arithmetical Relation, Value).

Table 1　Customer requirements description

Category	Type/constraint aspect		Cases
Global	1	Inherent constraint of the process	Logistic export service comprises five behaviors: orderCabinProxy, selectBoxStation, fetchGoods, customs, and setinPort.
	2	Inherent constraint due to business rule	If owner exports FCL (Full Container Load) goods, we'll use container in land transportation; otherwise, we use Flatbed truck.
	3	Customer decision on branch of process	If owner selects to send goods to boxStation himself, the process of logistic export service will not include fetchGood behavior.
	4	Global scale QoS constraint	The whole process of logistic export service should not last for more 72 hours.
Partial	5	Relation of single attribute among/between steps	Inside fetchGoods, under level behaviors getEmptyBox and backFullBox must be done by the same provider.
	6	Relation of single attributes of multiple steps	If getEmptyBox costs more than three hours, the time spent on backFullBox should be less than two hours.
	7	Relation among/between multiple attributes of multiple steps	If the credit grade of orderCabinProxy is lower than three, the price of selectBoxStation should be no more than 400 RMB Yuan.
	8	total QoS of multiple steps(not all)	fetchGoods(including getEmptyBox and backFullBox) should cost no more than 10 hours.
	9	Interaction Quality between two steps	The good loading time between getEmptyBox and backFullBox should be no more than 4 hours.
	A	behavior input or output	The format of CabinBookingSheet as output of orderCabinProxy should be the same as the input of selectBoxStation.
Single	B	Provider of a single step	The service provider of behavior customs must be the custom broker named HaiLong.
	C	One attribute of one step	Time spent by backFullBox behavior should be less than 4 hours.
	D	Relation among/between multiple attributes of one step	If the time spent is more than 3 hours, the broken rate must be less than 0.01.

Basic type is subdivided into three types according to differences in scope of constraint, as follows:

(1) Local constraint: subjects to single step of the process. Cardinality of Function is one. Case 6 in table 1 can be presented as Basic ({backFullBox}, (time, <, 2)).

(2) Global constraint: subjects to the entire process. Cardinality of Function equals zero. Case 4 in table 1 can be presented as Basic (∅, (time, <, 72)).

(3) Partial constraint: subjects to some steps of the process. Function has elements of all function types related to these steps. Case 9 in Table 1 can be presented as Basic ({getEmptyBox, backFullBox}, (time, <, 4)).

4.2.2 Compare-type constraint

This type is defined from the comparative view of multiple steps of service process. It's a triple: Compare (Function, Attribute, Arithmetical Relation). Case 5 in table 1 can be presented as Compare ({getEmptyBox, backFullBox}, provider, =).

4.2.3 Rule-type constraint

It's a pair: Rule (If, Then). If represents pre-condition of the rule, and then means the post-condition. They are triples of the same format: If (Basic, and, Compare); Then (Basic, and, Compare). It can be proved that or-logic can be represented by and-logic, and then we find it a full logical system taking relations in Basic and Compare in count. Case D in table 1 can be presented as Rule (({({backFullBox}, (broken rate >, 0.01))}, and, ∅), ({({backFullBox}, (time, <, 3))}, and, ∅)).

If the pre-condition is null, Rule-type constraint falls into one or several Basic-type or Compare-type constraints. For example, Case 5 in section 3.2.2 can also be presented as Rule (null, (∅,and, {({getEmptyBox, backFullBox}, actor, =)})).

4.3 Priority of customer requirements

Customer requirements classification reduces customer requirements into several types of constraints, thus simplifies the program logically for service composition. Meanwhile, customer may want to define which is more important among the constraints and possibly provide certain policy related information, so it's necessary to endow each constraint with measurable importance. In the view of this necessity, we divide customer constraints into four types of priority: mandatory, optional, one from many and many from many.

4.3.1 Mandatory level

This level is of top priority, if not satisfied, service composition will turn invalid, and customer satisfaction will be zero.

4.3.2 Optional level

Constraints of optional level can either be satisfied or not without voting the solution down, but once they're satisfied, customer satisfaction will increase forcefully.

4.3.3 One/Many form many level

To define this level, we'd have to give formal description of the former 2 levels first: Suppose is the set of all constraints transformed from customer requirements, and $d(c)$ is the priority of constraint, and let its domain be. Then set of all mandatory constraints can be represented by $S_m = \{c \mid d(c) = 1\}$, and set of optional constraints by $S_o = \{c \mid d(c) < 1\}$, and $S_m \bigcup S_o = S$.

Suppose S_p is a set of multiple constraints, and $S_p \subseteq S$. Let $|S_p| = m$ and customer want to express message that at least n of the m constraints are satisfied as mandatory require. It does not mean $\exists S' \subseteq S$, $S' \bigcap S_c \neq \varnothing$, $S' \bigcap S_o \neq \varnothing$, and $S' = S_p$ stands, but is reasonable in many from many meaning. Specially, when $n = 1$, "many from many" falls into "one from many".

As for as the relation among these priority levels is concerned, if $n = 0$, "many to many" transforms into optional level; similarly, if $S_p \subseteq S_o$, $n = m$ stands, and "many from many" transforms into mandatory one.

5 A Real-Life Service Composition Method Based on Genetic Algorithm

As a further explanation to the SC matchmaking engine we proposed in Section 2.2, this section is organized as following: chromosome encoding, the way that we encode the individuals in; reality featured techniques for evolution, the methods we use to solve this specific type of problem; evaluation function, the assessment to individual; termination condition, concerning when to stop computing; failure tackling, what to do when solution fails to find satisfactory results; the algorithm, detailed steps of composition; experiment and analysis, the experimental verification of our plan.

5.1 Chromosome encoding

We adopt integer coding with fixed length equal to step number of service process, as Fig.3 shows. A gene is a specific SC fulfilling the corresponding function of one step in process, and a chromosome is an individual in population standing for one possible composition result of the solution.

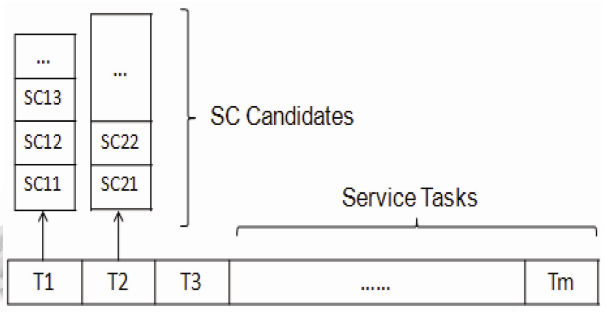

Fig.3 Chromosome encoding

5.2 Reality featured techniques for evolution

The constraints transformed from customer requirement affect performance of GA dramatically on the stability of evolving toward the optimal direction. Here we introduce evolutionary stability keeping strategy of local taboo and correlation degree based gene selection for efficient evolution. Other operations like individual selecting, crossing, and mutating can be done in ordinary manner of GA.

5.2.1 Local Taboo strategy for evolutionary stability keeping

As an intelligent optimization algorithm with randomization and probability features, GA finds its own evolutionary direction under the guidance of schema theorem, thus decreases computational cost by recursive computing within incomplete solution space. But after the introduction of customer requirements and with the enhancement of customer requirements representation, GA gradually loses its directionality which used to be relatively stable, and show itself more and more with randomized feature.

This paper suggests local taboo strategy as a preliminary solution to ensure the continuously improvement of evolution, i.e., do local operations to mandatory constraints that involving multiple steps of the process. Take rule-type constraint for example, suppose $IF\ Basic_1\ THEN\ Basic_2$, where $Basic_1$ is about function A, and $Basic_2$ is related to function B, then we say A and B has correlation. If the number of relevant functions is low (in this example it's 2; the maximum number needs to be modified depending on specific situations), we mark all SC collocations of relevant steps that do not satisfy this constraint as a fatal fragment (can be continuous or discontinuous). Through full-space computing to relevant fragments and eliminating the occurrence of these SC collocations in population, we save duplicated computation on invalid individuals in each recurrence. It's easy to prove, that this strategy has no conflict with the strategy of keep fixed proportion of invalid individuals in population, which is frequently used in GA to maintain diversity.

5.2.2 Gene selection by correlation degree

Local taboo strategy cannot afford checking all mandatory constraints to ensure no invalid individual occurs when we take computational amount into account. In most cases, we check only constraints concerning less than 3 steps. Though these constraints count for the absolute majority of all constraints, invalid individuals are still possible to occur.

We define the sum of priorities of all unsatisfied constraints related to one gene of the selected individual as the correlation degree of this gene to these constraints, and call it correlation degree for short. For example, the correlation degree of the k-th gene of a certain individual can be presented as following:

$$R_k = \sum w_i \quad , \ i = 1,2,...,m_k \tag{1}$$

where m_k is the number of constraints relevant to the k-th gene, and we further define the mutation probability of the k-th gene of a certain individual as a ratio, i.e., the proportion of this gene' s correlation degree in the sum of all genes'correlation degrees, as follows:

$$P_{mg,k} = R_k / \sum R_i \quad , \quad i = 0,1,...,m \tag{2}$$

where R_i is the correlation degree of the i-th gene, and is the number of gene one individual contains, which we call chromosome length, too.

Experimentally, it's of dominant probability that customer set mandatory and optional constraints of the same type at the same time on single attribute of one gene, to represent the lowest and satisfied quality of fulfillment respectively. In addition, suppose the constraints involving more than one gene just cover all attributes of all genes and they are all optional, then we get the priority domain of optional constraints as [0, 0.5] to ensure at each attribute of each gene, the sum of all mandatory constraints' priorities greater than that of optional, thus make sure mandatory constraints are the determinative factor of an individual.

5.3 Evaluation function

There' s no unified method to deal with optimization problem with constraints currently. One generally accepted method is Penalty function method, there're also other methods[17]. Ref.[18] combines direct-compare method with strategy of keeping self-fitted proportion of infeasible solution to tackle constraints, but direct-compare method compares two individuals using different standards in different situation, what increases the complexity of processing; the theoretical basis of strategy of keeping self-fitted proportion of infeasible solution contains pre-condition of its utilization that infeasible solution should take a great proportion in diversity space of characteristics, but after local taboo strategy, infeasible solutions get really rare. This paper defines unified evaluation function as following:

$$F(x) = P(x) - N(x) \tag{3}$$

where $P(x)$ and $N(x)$ represent positive and negative measurement of individual fitness respectively, and they're both non-negative functions. $F(x)$ has the following characteristics:

① If individual x does not violate any mandatory constraint, $P(x) > 0$ and $N(x) = 0$;

② If individual x violate one or more mandatory constraints, $P(x) = 0$ and $N(x) > 0$;

③ As a conclusion, an individual in (1) must get a higher fitness than that of the one in (2).

5.3.1 Positive Measurement of Individual Fitness

Positive measurement $P(x)$ is the aspect assessing the trend of fitness increasing.

$$P(x) = \sum w_i / \sum w_{\text{all}} + \alpha(\lambda \sum w_j Q_j / \sum w_j Q_{\max} + (1-\lambda) N_p / N_{\text{all}}) \tag{4}$$

$$i = 1,2,...,m \ , \quad j = 1,2,...,q$$

where m is the number of optional constraints satisfied, W_i is the priority of i-th optional constraint satisfied; q is the number of QoS attributes of SC, W_j is the weight of j-th attribute; α is a percentage representing the relative weight of non-constraint factors compared with constraint; λ is a decimal between [0,1] used to represent the proportion of comprehensive QoS measure- ment and handover times of same provider in total weight of non-constraints, and usually λ has a value of 0 or 1 according to different business domains.

Equation (4) comprises three parts: the extent that the constraints are satisfied, which is our main concern; the comprehensive QoS Measurement, whose effect can be extremely significant when all constraints are satisfied; and handover times of same providers, which is designed to embody real life service characteristics in a relatively narrow way.

(1) Comprehensive QoS measurement

QoS of results is measured in execution time, price, credit, and reliability aspects, and the comprehensive value is calculated based on the single value of each aspect.

Each individual is a composite result with schedule, and each SC of it is designated with a start time representing flexible range. For both SC and individual, they are accepted as possible result as long as their optimal possible performances fit, so they get possibility of failure, too. That's why we take reliability into account representing the possibility of successful execution. Suppose is the deadline, and the range of an individual is, we apply uniform distribution or normal distribution to this range, and the proportion of the area before the deadline within that of whole range is the reliability. Other three attribute can be obtained directly from SC description.

We adopt the QoS calculation method in Ref.[13] for composite process and take the weighted average as a comprehensive measurement for QoS.

(2) Handover times of same provider

If two adjacent steps of service process have the same provider or their providers belong to a same business alliance, add 1 to handover time. In some cases, services provided by same provider bring potential discount or higher quality to customer, so preliminary we count it into the positive aspect of factors.

5.3.2 Negative measurement of individual fitness

Negative measurement $N(x)$ assesses the extent that the constraints are violated.

$$N(x) = (n_c + \sum w_i)/(N_c+N_o) \quad (5)$$

where n_c and n_o represent the number of the violated cons- traints that are mandatory and optional respectively, and $i = 0,1,...,n_o$. For "n from m" type constraint, the number of mandatory constraints violated is the number that satisfied constraints number smaller than n, and the number of optional constraints violated is $m - Max\{n, satisfied\_number\}$.

5.4 Termination condition

Easy to see from $F(x)$, that the lowest fitness satis- fying customer requirements is a positive decimal infini- tely close to zero. So we set longest computing time for one request according to capacity of SC library and workload of the server, to deal with cases the customer set the satisfied fitness too high; Besides, we can set upper bound to customer defined satisfied fitness according to historic analysis or level of membership of the customer. Above are the two terminating conditions of the algorithm.

5.5 Failure tackling

We react from two angles when program failed to provide satisfactory results as following:

①Still return to customer several relatively superior results for choice, meanwhile, echo the constr- aints that could be too rigorous for each result.

②Review on matching adjacent step pairs to reveal possible restrict factors or nodes leading to global failure. Interaction time for web service is usually short and can be neglected, but in realistic service, both inside and inter SC exist lots of human participation. If SC takes a long time to react or feedback to its precedent, we say these two SCs have bad interaction performance in time, and we should consider replacing SC having the slow response with another one, or send advice to its provider for improvement to satisfy customer requirements.

5.6 The algorithm

By combining generic GA with the above specia- lized solution with service feature, we get a new GA base algorithm, and call it SC-GA. Service matching process of SC-GA has two stages: pre-processing stage and execution stage, as follows:

Pre 1: parse customer requirements description file, acquire information of customer initiating this request, and then find and parse service process description file according to specified business domain.

Pre 2: retrieve SCs from library according to service process description, query relevant information of SC from QoS tree and historic QoS and credit repository, and finally filter SCs do not satisfy local constraints within each candidate SC set.

Step1: set input parameters, including proportion of individuals participating cross operation each time with in population, the upper bound of related function types of the single mandatory constraint that local taboo strategy can apply to, the number of final results N_r , and the time limit of one execution $T_{\max}$; set program internal parameters, including the generation count $gen = 0$, the final result set $S_r = \varnothing$; Initialize population $P(0)$ under local taboo strategy, and compute the fitness of each individual, presented by $F(x)$.

Step2: select N_r globally best individuals from both population of generation gen and S_r .

Step3: if all individuals in S_r are satisfactory or compu- ting time reaches $T_{\max}$, algorithm terminates.

Step4: let $gen = gen + 1$;

Step5: select individuals using roulette method from $P(gen-1)$ to form $P(gen)$, ensure the best individual in $P(gen-1)$ is also in $P(gen)$.

Step6: do cross operation to individuals of $P(gen)$ according to R_c and cross rate P_c .

Step7: Select individual from $P(gen)$ according to mutation probability P_m , and do mutation operation according to the mutation probability of each gene on the selected individual P_{mg} . If the mutation results do not satisfy local taboo, redo Step 7;

Step8: Calculate the fitness of each individual $F(x)$ in $P(gen)$;

Step9: *goto* Step 2.

6 Experiments and Analysis

To realize service composition in real life, firstly, we have to define rules for the specific business domain and standards for each type of the SCs including QoS architecture, and then attract and gather numerous service providers to publish their service ability in form of SC onto the Service platform, Finally, service platform takes the responsibility of matching different SCs according to customer requirements and internal policies and of tracking the quality of SCs in execution for making advice to service joiners of both sides in the future.

In prototype system, we define a two-level service process with eight top level steps and nine types of function. As shown in Fig.4, B and C are paralleled branches of A; C is a virtual node or sub-process composite by C1 and C2; D and E are optional branches of B with historic possibility of P1 and P2 respectively; and F is a recurrence of itself in process.

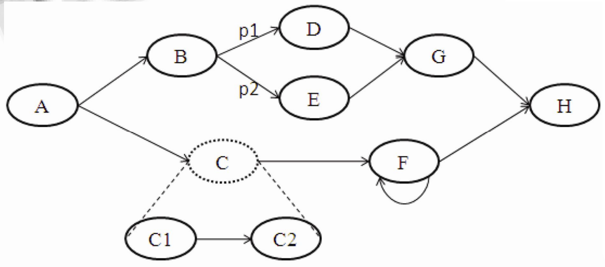

Fig.4 Service process in prototype system

Firstly, we generate sixty to ninety SC descriptions randomly in SC library for each function type. All the SC descriptions are in accordance with the SC definition in section 2.1, and the QoS values of them are showed in Table 2.

Table 2 QoS Range of Randomly Generated SC

QoS attributes		Range of value
Generic attributes	time	10~30 minutes
	cost	200~1000 RMB Yuan
	credit	1~5 grade
	provider	BlueStar/MinSheng/Waiwell/FESCO
Specialized attributes	Success rate	0.90~1.00
	Broken rate	0.00~0.02
	Dead line	Manually defined

Secondly, we write programs to generate description file for customer requirements according to the description model with classification feature we defined in Section 4.2. This file contains constraints of multiple types with different priorities, as shown in Table 3.

Finally, let the result number be 2, we execute our algorithm based on the SCs and customer requirements description above. The results are showed in Table 4, where time point is presented in "Month DD HH:MM" format and time length in "HH:MM" format.

From Table 4, we can see step A gets a designated concrete SC which is identified by sc92609500 & level 1 as part of the composite solution and this SC is supposed to execute from 12:30 on Oct. 16 with no float time range.

Fig.5 shows the statistical information during genetic process. We can see that the average fitness of population keeps a generally steady increase within 150 generations. Multiple calculations show that this result gets a fitness reaching 0.95 of the optimal one. Our solution proved fine efficiency in finding optimal results.

Table 3　Demo constraints in prototype system

Constraint type	Target section	Constraints	Priority
Local	A	cost < 800	1
		finishDate < 1224132913937	0.2
		succeedProbobility >= 0.95	1
	B	cost < 800	0.4
		time < 1500000	1
		creditGrade >= 2	0.3
	C	time < 1500000	1
		brokenProbobility < 0.1	1
	D/E	cost < 800	0.13
	F	time < 1500000	0.21
Global	All	cost < 4000	1
		finishDate < 1224138913937	1
Partial	C, F	cost < 1500	0.22
	C1, C2	creditGrede>=3	1
		finishDate < 1224132913937	0.3
Compare	B, G	B.provider= G.provider	1
Rule	A, C, D, E	if A.cost < 800 and C.provider=D.provider then E.cost < 700	0.4

Table 4　Service composition results

sequence	1	2
A	Oct. 16 20:30 ± 00:00 sc1312 Level1	Oct. 16 20:30 ±00:00 sc1312 Level1
B	Oct. 16 20:45 ± 00:04 sc4515 Level1	Oct. 16 20:45 ± 00:04 sc2593 Level1
C_1	Oct. 16 20:47 ± 00:01 sc2359 Level0	Oct. 16 20:47 ± 00:01 sc2359 Level0
C_2	Oct. 16 21:08 ± 00:08 sc3984 Level0	Oct. 16 21:08 ± 00:08 sc5437 Level0
D		
E	Oct. 16 21:04 ± 00:10 sc4234 Level0	Oct. 16 21:06 ± 00:05 sc4234 Level0
F	Oct. 16 21:29 ± 00:09 sc6093 Level0	Oct. 16 21:25 ± 00:15 sc6093 Level0
G	Oct. 16 21:26 ± 00:11 sc2078 Level0	Oct. 16 21:27 ± 00:07 sc2078 Level0
H	Oct. 16 21:42 ± 00:16 sc4859 Level1	Oct. 16 21:49 ± 00:14 sc4859 Level1
fitness	58.14	57.78

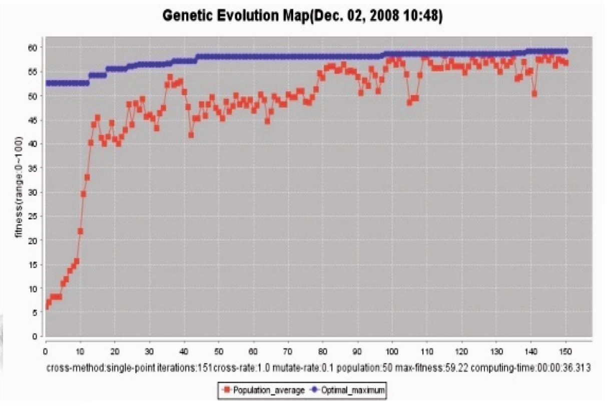

Fig.5　Statistics during evolutionary process

7　Conclusion and Future Work

Current research on service composition is limited to web service domain. To integrate realistic service in a similar way, this paper puts forward one service composition method with classification for customer requirements as the main clue. To put it in detail, firstly we make an abstraction from realistic service to service component, and propose a classification and reduction method for customer requirements, and then based on those, give the genetic approach as a solution. Tests show that this framework can not only generate the scheduling composite results that satisfy customer requirements, but also ensure fairly nice efficiency on finding the optimal results.

Integration of realistic service is a big area. This paper emphasizes on service composition regarding SC functionality more as a query index. For future work, we will continue to define more reasonable genetic techniques and evaluation methods. We will anticipate a consummation in exploring the generic practices or enhancing methods that may exist when we consider service related feature a critical clue in tackling service composition problem using intelligent optimization algorithms. Furthermore, as a basis, we will pay effort on extracting meaningful information in realistic service to form simpler and efficient service component description.

References

1 Xu XF, Wang ZJ, Mo T. Methodology for Service Engineering. Computer Integrated Manufacturing Systems, 2007,13(8):1457－1464.

2 http://www.oasis-open.org/committees/wsbpel.

3 http://www.bpmn.org/ Documents/IntroductiontoBPMN.pdf.

4 http://www.daml.org.services/owl.

5 http://xml.coverpages.org/bpel4people.html.

6 Jin Z.Ontology-Based Requirements Elicitation. Chinese Journal of Computers, 2000,23(5):486－492.

7 Wen BL, Zhang Q, Ma SB. A Requirement Description Method Based on Resources. Computer Engineering & Science, 2008,30(10):135－138.

8 Yao QZ, Li XL, Meng L. The research of component library management framework based on XML. Computer Engineering and Applications, 2006,42(21): 78－80.

9 Chen YP, Li ZZ, Guo ZS, Jin QX, Wang C. Service Selection Algorithm Based on Quality of Service and Its Implementation for Web Services Composition. Journal of Xi'an Jiao Tong University, 2006,40(8):897－905.

10 http://www. w3.org/TR/wsdl.

11 Wang JH. Extension for Semantic Web Services Description Based on OWL-S. Journal of Guangxi Normal University (Natural Science Edition), 2008,26(1):158－161.

12 Zhang LY, Luo G, Lu LN. Genetic Algorithms in Resource Optimization of Construction Project. Journal of Tianjin University, 2001,34(2):188－192.

13 Liu SL, Liu YX, Zhang F, Tang GF, Jing N. A dynamic Web services selection algorithm with QoS global optimal in Web services composition. Journal of Software, 2007,18(3):646－656.

14 Feng D, Lu C. Constrained optimization research based on the additional recessive inheritance of genetic algorithm. Henan Science, 2005,23(6):884－887.

15 http://www.sla-zone.co.uk/index.htm.

16 http://download.boulder.ibm.com/ibmdl/ pub/software /dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf.

17 Sun YF, Zheng JQ, Wang DX, Wu H. A Survey of constraint optimization method based on genetic algorithm. Journal of Northern Jiao Tong University, 2000,24(6):14－19.

18 Lin D, Li MQ, Kou JS. A GA-Based Method for solving constrained optimization problems. Journal of Software, 2001,12(4):628－632.