http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (4): 807-815, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.807.815

Optimizing Performance of Composite Services in Multiple Networks Enterprise Environment

Shuai Zhang, Jianling Sun, Yuanhong Shen, Yixi Chen and Aleksander J. Kavs
Room 521, Dorm 25, Yuquan Campus, Zhejiang University, Hangzhou, Zhejiang, 310027, China

Abstract: Many approaches have been proposed for performance optimization of centralized services composition. However, with the prevalence of web services, services orchestration evolves from intra-enterprise integration to cross-enterprise integration and involves multiple networks. In such environment, the centralized orchestration suffers from several performance and reliability issues due to the limitation of centralized execution engine and the network conditions. In this paper, we try to optimize the performance of composite services in enterprise environment by adjusting the deployment location of component services. First, we propose a decentralized orchestration model for multiple networks environment. Based on this model, we propose the concept called service interaction strength, which is denoted by the data interaction between component services in a composite service as well as the importance of composite service itself. Then we design an objective function for all composition services with overall minimum cross domains service interaction costs and thus reduce the response latency and increase the performance of composite services. At last we introduce a mixed repair genetic algorithm to get the optimized deployment plans for all component services. Experimental results on simulation prototype show the optimization effectiveness of our approach and the performance of algorithm itself is also acceptable.

Key words: Service composition, enterprise environment, multiple networks, performance optimization, genetic algorithm

INTRODUCTION

Today's competitive and global market forces enterprises to respond effectively and quickly. With Service Oriented Architecture (SOA) (Papazoglou, 2007), flexible business processes can be achieved by integrating internal systems, as well as coupling external business partners, so that enterprises could focus on their core business functions and outsourcing the others. These kinds of integration are called services composition. While SOA can be achieved by different technologies, Web Service (Curbera *et al*., 2002) is the de facto implementation due to its standardization efforts, massive tools and infrastructures. In this paper we will use the term Service and Web Service interchangeably.

Performance optimization problems of services composition are hot topics in recent years in both academic and industry area because of their importance and difficulty.

However, with the prevalence of web services, services orchestration evolves from intra-enterprise integration to cross-domain integration and involves multiple networks (Chafle *et al*., 2007). As proliferation of Web Services combinations between organizations, they will affect the network in numerous ways (Mullaley, 2005)

- Site-to-site traffic will grow exponentially due to the distributed services execution among multiple domains and the employment of the XML based protocol for inter-Web services communications
- Low latency will be an increasingly critical network characteristic. There will be a lot of time-sensitive inter-process communications traveling over the WAN rather than being executing on a single server or handled by high speed Enterprise System Bus (ESB)
- Since web services can be combined in many complex ways, the traffic patterns will be less predictable
- The priority of network packets identification will become increasingly difficult. Certain Network traffics, which are considered with low priority normally, can become high priority because it is in a critical path of high priority composite service

Consequently, in distributed SOA, network latency will directly affect the end user's experience; it is inevitable that the performance of these cross-organizational business processes at runtime becomes a major issue. In our previous work (Yin *et al*., 2010) we have proposed a QoS-aware services composition approach for multiple network environment

Corresponding Author: Shuai Zhang, Room 521, Dorm 25, Yuquan Campus, Zhejiang University, Hangzhou, Zhejiang, 310027, China Tel: +8613819125160 Fax: +8657128111775

which is based on centralized orchestration. However, this mode suffers from several problems like single point of failure and bottleneck of performance.

To solve these problems, we've proposed a performance optimization model for composite services in multiple network environments (Zhang *et al*., 2010). This study largely extends the work. First, we propose a performance optimization model for composite services in multiple networks environments. In this model, services are orchestrated in decentralized manner to avoid the single point of failure and performance bottleneck of centralized execution engine. Based on this model, we propose a concept called service interaction intensity, which is denoted by the data transmission strength between services and the importance of composition processes in the enterprise. With this concept, we can figure the priority of data transmissions between services.

Then we design an objective function for all composition services with overall minimum cross domains service interaction costs and thus reduce the response latency and increase the performance of composite services. At last we introduce a mixed repair genetic algorithm to achieve this objective function and implemented a simulation prototype.

MOTIVATIONS

Web service orchestration: Service composition can be orchestrated in centralized as well as decentralized manners (Patcas *et al*., 2005). In centralized service composition, component web services have to pass the data back to the execution engine and they do not interact with each other directly. The execution engine is responsible for manage all the control logic and distributing data to related web services. The advantage of this mode is simple and easy to manage.

Contrarily, in decentralized service composition, the control logic is distributed into several execution engines and the data are transmitted direct among component services in peer-to-peer mode, which is more efficient in most cases and is more robust.

Here we use a scenario to illustrate the difference between two orchestration modes. Figure 1 is a simple example for web service composition. It is a simplified risk evaluation service of portfolios for a Fund Investment Company. Service 1 is a historical stock pricing data web service from some partner data providers like Bloomberg; Service 2 is a fund accounting system. It produces the performance of portfolios like NAV, beta index, etc based on the market prices and stock transfer information of portfolios. Service 3 is an internal risk management system; it calculates the risk index based on confidential

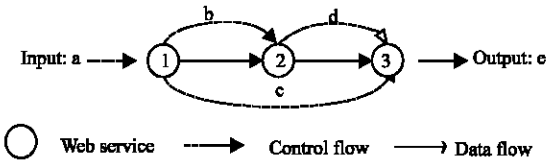

Fig. 1: Example of service composition

internal risk model with the market data from Service 1 and portfolio information from Services 2.

Correspondingly, Fig. 2 is an illustration for this scenario in both centralized and decentralized modes. It shows the difference in data flows intuitively. For example, In 2(a), all data exchange is based on the centralized composite engine as well as in 2(b), the market stock pricing data (represent by b and c) produced by service 1 are sending to service 2 and 3 directly.

At present, centralized composition is the mainstream manner. A typical example is the WS-BPEL (OASIS, 2007) which is the standard orchestration language in industry area. In our previous work (Yin *et al*., 2010) we have proposed a QoS-aware services composition approach for multiple network environment which is based on centralized orchestration.

However, this mode suffers from several issues as mentioned above. Moreover, in the multiple network environments, the performance of composite services like the response time is highly limited by network status between every component service and the composition engine; overall performance will be affect if any of these connections is suffered from network issues.

To solve these problems, many researchers from both academic and industry put their eyes on decentralized services orchestration. For example, IBM India Research Laboratory has done a series of researches. They proposed a technique to partition a composite web service written as BPEL program into an equivalent set of decentralized processes based on the dependence among component services (Nanda *et al*., 2004). Furthermore, they discussed the additional complexity of this decentralized manner like error recovery and fault handling (Chafle *et al*., 2004). Based on their works, Binder *et al*. (2006) proposed a trigger based decentralized orchestration model, in which a transparent trigger is deployed for each web service. The distributed invoke of services as well as error handling are all implemented by triggers.

Web services performance optimization: XML-based languages implement webs Services. All function invocations and message transmissions between services are throw SOAP messages. While the main goal of this

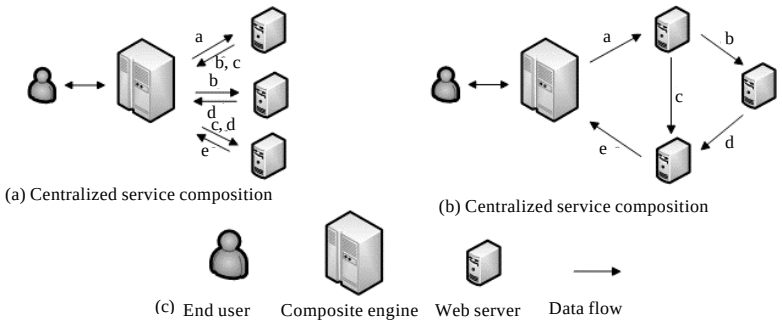

Fig. 2: Centralized vs. decentralized service composition

implementation is to hide the complexity of underlying systems and provide communication between different languages and platforms, it also introduced many performance issues. Hence, many performance optimizations of web services are focusing on data processing like XML messages processing and compression (Ericsson, 2007) and services requirements caching and rescheduling (Dyachuk and Deters, 2007), etc.

For composite web services in public Internet, as plenty of services are available, it is quite common that multiple web services have overlapping or even same functionality but have different QoS properties like response time, available rate, cost, etc. In such case, how to select candidate services for overall best QoS is the essential problem and has been discussed in vast majority of literatures (Menascé, 2002; Ran, 2003; Zeng *et al*., 2003).

However, in enterprise environments, candidates of component services are limited or even fixed and composite services are always crossing over many network domains due to business collaborations. Chafle *et al.* (2007) studied the performance optimization problem in such environments. The author proposed a system that assumes the location of component services is determined in advance and the data flow routes can be changed dynamically. Firstly it analysis the composite services and enumerate all the possible data flow topology structures. During runtime the system will monitor the availability of network bandwidth of each network domain and evaluate its effect on the performance of each topology using a performance evaluation model. When the current topology could not satisfy the SLA (Service Level Agreements) of composite services, the following service requirements will be changed into new topologies without affecting the running ones.

Different from the study Chafle *et al.* (2007) in this study we assume that the deployment location of component services are partially changeable e.g., in a cross-enterprise business process, the inner services can be deployed into any network domains of the enterprise, but the outer services provided by business partners is located in other domains and cannot be changed. In such a condition, the overall performance of composite services can be affected by different deployment policies of component services because the network costs among different network domains are different from each other.

PROBLEM DEFINITION

In this study we propose the preliminary definitions of the optimization problem under multiple networks environment. First we present the decentralized orchestration model. Then we define the performance optimization problem in this model including network costs and the interaction intensity between component services in composition.

Problem statement: Due to the geographical distribution, enterprises and their partners are located in different areas and thus in different network domains; one single enterprise may also have many different network domains. Both the enterprises and their partners implement all kinds of different standalone services based on business divisions. These services are deployed into different network domains, but need to be composited together and serve for clients from all over the world.

From the perspective of enterprises, services are divided into two different types: the inner services and the outer services. Inner services are fully controlled by the enterprises including the deployment location while outer services are controlled by business partners and can only be involved throw standard ways like WSDL.

The network conditions (like connection speed, bandwidth, economic cost, etc) between different network areas may distinct from each other, thus, the data transmission among component services will have

different costs while deploy them into different network domains. Network traffics between different domains in the wide area network are much higher than in local area network like the internal network in the same building. For example, put two services which have high data exchanging frequency into the same domain should have a better performance than into two domains those are far from each other. Besides, the costs between domains are usually different; it is determined by many reasons like geography distance, physical links and bandwidth, etc. Normally, the network cost from New York to Boston should be lower than to Beijing.

Service interaction is prevalent when a composite service is running due to service correlation caused by data or logic dependencies. When a composite service is deployed upon large domains of network, service interaction may require a large amount of network flow that cannot be ignored since it affects QoS of running business process. Moreover, enterprises have many different composite services for different business tasks; some component services such as authentication and authorization services are playing the role of common utilities and are shared by many different composite services. Thus, different deployment locations of component services, especially those common services, will highly affect the overall performance of composite services.

A MIXED DECENTRALIZED ORCHESTRATION MODEL

As discussed earlier both decentralized and centralized orchestrations have their own advantages and disadvantages. The former one has a better performance than the latter especially in multiple network environments; it will highly reduce site-to-site traffic and thus improve the QoS like latency and throughput of composite services while the latter one is much simple and easy to manage.

To utilize the benefits of both, our model combined these two methodologies together: The cross-domain service invokes and data transmissions are following the decentralized manner, while inner domain calls are following the centralized manner.

As shown in Fig. 3, the network structure in each domain is a traditional centralized orchestration model. Every domain mainly contains the Component Services, an Orchestration Engine and a high speed Enterprise Service Bus to which the former two are connected. Besides, there is a proxy in each domain. The proxy is a transparent agent for services invocations from both inner services and outer services.

Services are deployed into different network domains under the restriction of security policy, business division

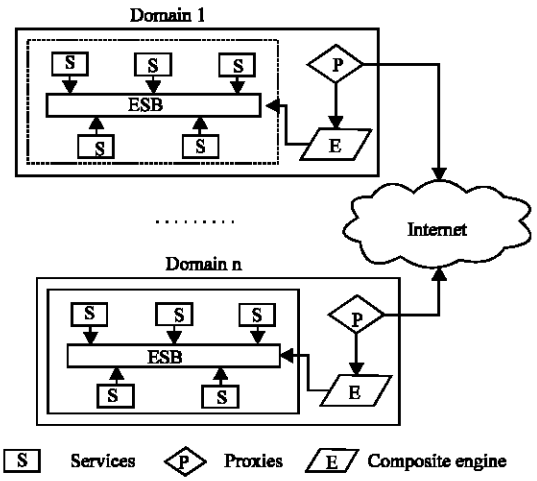

Fig. 3: A mixed decentralized orchestration structure

and the capabilities of the domains. All service invokes and data transmissions are throw the orchestration proxy, regardless of inner domain or cross domain, the proxy is transparent to component services.

A straightforward way to optimize the performance composite services in such a model is to put all component services into one single domain since the inner transmission costs could be ignored compare with cross domain costs. However, there are many constraints here:

- Services will require resources like CPU, memories, other hardware and software resource and so on during their runtime but the resources in a single domain are limited. Put all services in the same domain will have very serious performance issues under high request rate
- The locations of partner domains and outer services are fixed, inner services those have high data relevance with outer services should be deployed to domains those have lower cost to the partner domains

PERFORMANCE OPTIMIZATION OBJECTIVE

Formal definitions:

Definition 1: Component services: Define $s = \{s_1, s_2, s_3 \ldots s_n\}$ as the set of component services those are used in the enterprise; n is the total number of component services.

Define t as component service types. A service type can be inner or outer, representing by $s_i.t = \{inner, outer\}$. If $s_i.t = inner$, then s_i can be deployed in all inner domains of the enterprise; otherwise s_i can only be deployed in one pre-defined outer domain.

Definition 2: Composite service: Define S = {s_i, s_j, s_k,… $s_n \in s$} as a composite service. A composite service is represented as a set of selected component services those are combined together to achieve value-added functions.

Define CS = {S_1, S_2, S_3,… S_l} as the set of composite services those are using in the enterprise, m is the total number of composite services. Each component service could be shared by many different composite services.

Definition 3: Service interactions: Define matrix $IDs_k = [a_{ij}]_{n\times n}$ as service interaction dependency for the composite service S_k when it is running, in which a_{ij} represents service interaction times from component service s_i to service s_j. If there is no interaction between s_i and s_j in composite service S_i, $a_{ij} = 0$; moreover there are $a_{ii} = 0$, I={1,2,3...k}

$$ID = \sum_{i=1}^{1} p_i \times IDs_i \quad (1)$$

Expression (1) is the overall service interaction dependency matrix for all the component services those are in use, in which ID[i][j] = $p_1 \times S_1.aij + p_2 \times S_2.aij + p_3 \times S_l.aij$. p_i is the weight of the composite service S_i and $\sum_{i=1}^{1} p_i = 1$ It means the importance of this composite service in the enterprise and could be decided manually or by historical values.

Definition 4: Service resource requirement: Define $s_i.r = r_i$, i = {1, 2, 3… m} as service resource requirement for service s_i in a composite service. Service resource requirements refer to resource consumed by each component service when the composite service is running in network domains.

In fact, this notion is an abstraction of environment resources like CPUs, memories, hard disks, network, etc.

Definition 5: Network domains: Define ND = {D_1, D_2, D_3,… D_m} as the set of Network domains, D_k is the kth domain and m is the total number of domains.

$ND_{in} \subseteq ND$ is the set of domains inner the enterprise whose geography location maybe around the whole world.

$ND_{out} = ND - ND_{in}$ is the set of domains not managed by the enterprise, e.g. the network domains of business partners. It could be a null set.

Define $DS(s_i)$ as the deployable domain set of component service s_i, as defined in Definition 1:

$$DS(s_i) = \begin{Bmatrix} \forall D \in ND_{in}, & \text{if } s_i.t = \text{inner} \\ \exists D_j \in ND_{out}, & \text{if } s_i.t = \text{outer} \end{Bmatrix} \quad (2)$$

Definition 6: Domain capabilities: Define C as resource capability for each distributed network domain. $D_i.C = C_i$ i={1,2,3,… m}. Assume there are k deployed component services running in domain D_j, the total requirements of its deployed running services will not exceed its resource capabilities, that is:

$$\forall D \in ND, \sum_{i=1}^{k} s_i.r \le D_j.C \quad (3)$$

Definition 7: Network cost: Define matrix $P=[p_{ij}]_{m\times m}$ ($p_{ij}=p_{ji}$) as the unit cost for each cross domain service interaction when composite services are running, in which p_{ij} represents network cost between domain D_i and domain D_j per interaction. Assume the interaction cost caused by services in the same network domain can be neglected, there will be $p_{ii} = 0$, i = (1,2,… ,m);

Define 8: Service deployment result: Define n dimension Vector Dep = $[dep_i]_n$ is the final service deployment plan in distributed network environment for each inner and outer service, in which Dep[i] represents component service i is deployed in network domain ND[dep_i^{th}].

OPTIMIZING OBJECTIVES

The service deployment objectives is to minimize the service interaction costs between different network domains while meeting the capability-resource constraint of each domain; the optimizing result is represented by matrix Dep in Definition 8, the optimizing service deployment expressions are as followed:

$$Result = Dep$$

Where:

$$\text{Min} \sum_{j=i+1}^{n} \sum_{i=1}^{n-1} ID[i][j] \times p[dep_i][dep_j] \quad (4)$$

And match the limiting condition in expression (2) and (3)

This could be mapped to a multidimensional multiple choice knapsack problem which has been proved to be a NP hard problem (Martello and Toth, 1987). In this paper, we will use Genetic Algorithm to get approximate results.

GENETIC ALGORITHM AND REPAIR GENETIC ALGORITHM

GAs (Renders and Flasse, 1996) are search techniques which could be used for global optimization problem. They perform a search by evolving a population of candidate solutions through the use of nondeterministic operators and by improving incrementally the individuals forming the population by mechanisms inspired from

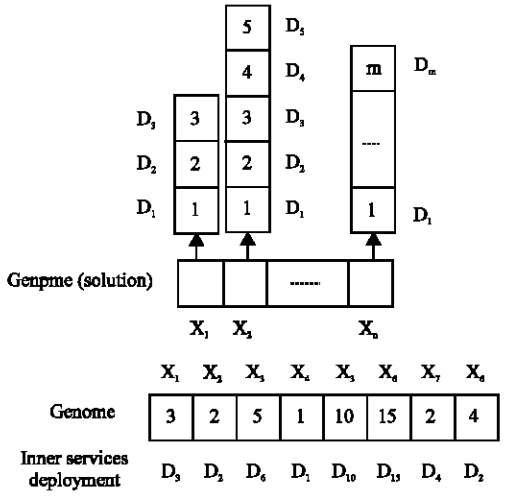

Fig. 4: Encoding and decoding

those of genetics (e.g., crossover and mutation). (1) Create an initial population consisting of randomly generated solutions; (2) generate new offspring by applying genetic operators, namely selection, crossover and mutation, one after the other; (3) evaluate the fitness value of each individual in the population; (4) repeat step 2 and 3 until the algorithm converges. There are many solutions to improve the efficiency of genetic algorithm while avoiding local optimum, e.g., Gene Space Balance Strategy (GSBS) (Hu *et al*., 2007) Information Entropy based selection strategy (Cui and Lin, 2004) roulette-wheel selection, linear rank selection, etc, to keep diversity of each generation and increase evolutionary efficiency.

For constrained optimization, two different constraints handling mechanisms could be incorporated into GAs. One is penalty mechanism, that is, to give penalty to infeasible individuals (the individuals who violate the constraints) when evaluating the fitness of the reproduced individuals. The GA with penalty mechanism is called penalty GA. When the constraint density is high, the fitness function will be complicated and should be carefully designed to handle constraints of different kinds, since it affects the effectiveness of the algorithm, what worse, a feasible individual cannot be generated after many generations in the penalty GA. Another approach is to use domain specific knowledge to fix up those infeasible individuals in the population such that all the individuals in the population are always feasible. The GA with such a repairing mechanism is called repair GA. In this study, we will incorporate an improved mixed heuristics GA to solve the service deployment-optimizing problem.

MIXED HEURISTIC REPAIR-GA

Genome encoding: The genome is encoded by an integer array with a number of items being equal to the number of atomic shared service for deploying. Each gene i range from 1 to m, where m is the number of inner services for deploying. The method is illustrated in Fig. 4. The bottom part of the Fig. 4 is an illustrative example basing on the services deployment result for inner services.

Genome operators: Once the initial population is achieved, rank-based selection mechanism is used to select individuals to participate in crossover or mutation, in which the individuals with higher rank (better fitness value) will be more likely to be selected. For the crossover operator, a couple of individuals will generate two offspring using the one point crossover operator. The mutation operator is conducted by randomly selecting a gene in the genome and replacing it with another gene whose state is available. However, the solutions of the offspring will be, in general, infeasible. In this case, the repair operator (mixed heuristic repairing operator) is applied to enforce feasibility. Besides, the elitist operator in the GA preserves the best solutions found by maintaining a group of them in the next generation. In our GA, the best two distinct individuals are kept alive across the current generations.

Fitness function: A fitness function is used to measure the quality of the individuals in the population according to maximize the optimization objective. In our approach, we use change the objective in expression (4) to $\eta - \sum_{j=1+1}^{n}\sum_{i=1}^{n-1} t_{ij}$ as the fitness function where η is big enough to ensure the positive quantity of the result.

Mixed heuristic repair operator: When a random individual is generated for the initial population or an offspring is acquired after crossover or mutation operation, it may be infeasible in most cases for violating the capability-resource constraints in expression (3). In order to enforce feasibility, we propose a mixed heuristic repairing operator to quickly repair the infeasible individuals or offspring while keeping diversity of individuals in each generation. The mixed heuristic repair process is a mixture of a heuristic repair process and a normal repair process, each of which with a separated probability, the algorithm of repair process for an infeasible offspring (representing service deployment result matrix: Dep) is shown in Table 1.

The heuristic repair process is like a hill climbing optimizing process, where each time an infeasible individual repairs interactively but making its best improvement for its overall objective. It is to increase the individual evaluation speed across different generations.

Table 1: Pseudo code of repair process for an infeasible offspring

*Input: R, C, T and Dep (one individual in one generation) for repair; */
/*Output: Dep after repair;*/
Define repair_max_time=max;
Randomly decide whether to perform a heuristic repair process (HRP) or a Normal RepairProcess (NRP), with probability of μ and 1-μ, ($0 \le \mu \le 1$)
repair_time = 0;
while (repair_time <= max)
  if (Dep meets constraints in (3)) then
    return Dep;
    end if
  if (HRP) then // heuristic repair process
    Choose si where si.t = inner:

and $\sum_{k=1,dep_k=dep_i}^{n} s_k.r > D_{dep_j}.C$

and $\forall s_j.t = inner, \sum_{k=1}^{n} p_{ik} \ge \sum_{k=1}^{n} p_{jk}$

Chose Di ∈ Ndin:

and $R[i'] + \sum_{k=1,dep_k=i'}^{n} s_k.r \le D_{dep_{i'}}.C$

and $\forall s_{j'} = inner, \sum_{k=1}^{n} p_{i'k} \le \sum_{k=1}^{n} p_{j'k}$

else // normal repair process
 Randomly choose Si where Si.T=shared:

and $\sum_{k=1,dep_k=dep_i}^{m} C[k] > R[dep_i]$

Randomly choose Di '∈ Ndin:

and $R[i'] + \sum_{k=1,dep_k=i'}^{m} C[k] \le R[i']$

end if

if (Di ' doesn't exist) then
repair fails and exit;
else

$dep_i = i'$ and repair_time++;

end if
end while

And the normal repair process is mixed with the heuristic repair process to shorten calculation time for each repairing process and helping keeping diversity of individuals to avoid local optimum.

Maintaining diversity: To ensure global convergence of the genetic algorithm and avoid local optimum while increasing convergence rate, it is important to maintain diversity of individuals in each generation. We use theory of Information Entropy (Cui and Lin, 2004) to maintain diversity of individuals for each generation, the definition is as followed: the joint entropy of individuals is defined as , $H(M) = \frac{1}{M}\sum_{j=1}^{M} H_j(N)$ in which M is genome number of each generation, $H_j(M)$ is the entropy of information for the jth genome. $H_j(M) = -\sum_{i=1}^{N} P_{ij} Log_2 P_{ij}$ in which P_{ij} represents the probability of the ith symbol appearing at the jth gene location. Population similarity $A(M) = \frac{1}{1+H(M)}$, when the population similarity exceeds a threshold, population renew procedure begins as followed: 1) if population similarity exceeds threshold A, generate P new individuals, otherwise renew procedure is finished; 2) select M district individuals with highest fitness from (M+P) individuals; 3) if there are only N (N<M) district individuals selected in step 2, generate (M-N) new individuals, otherwise, skip to 1).

Table 2: Optimize effect

	TS	ID	IC	OC	Rate (%)	Time
1	30	5	373.22	294.91	21	3000
2	30	10	402.94	265.01	34.20	4344
3	50	15	1146.21	786.94	31.30	14890
4	100	20	4762.95	3878.12	18.60	221468

EVALUATION

The purpose of the evaluation is to validate the optimization efficiency of GA-based services deployment approach proposed in this paper in the multiple network environments as well as the performance of the algorithm itself.

We designed a simulation program to generate test cases. This program takes 3 main parameters: the total number of Component Services (CS), Outer Services number (OS), Inner Domains number (ID). Each outer service is located in its individual out domain thus the total domain number is OS+ID.

The program will generate other parameters with reasonable random distributions:

- The Service Interactions matrix is generated with Gaussian distributions in a given upper and lower bond and ensures that about 50~70% services have interactions
- Resource requirement for each component service and the capability of each domain is generated with reasonable random distributions in a given range

All experiments were conducted on a Lenovo Workstation with Intel Core 2 Duo 2.33GHz processor, 2GB RAM. The software environment is Windows XP with SP3 and JDK 1.6 update 17.

OPTIMIZATION EFFECT

Series of experiments in this section is aim at testing the optimization effect of our algorithm. We designed 4 set s different input parameters as 4 scenarios, which are listed in Table 2.

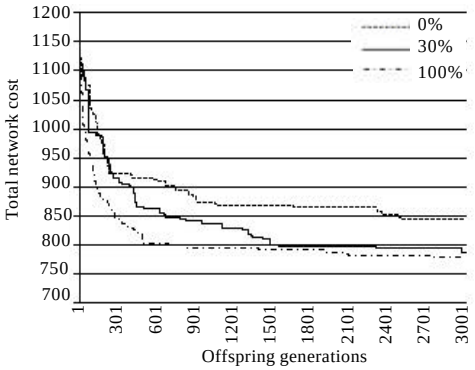

Fig. 5: Heuristic rate and optimization efficiency

In this series of experiments, the heuristic parameter is set to 30%, which means about 30% of offspring repair will be heuristic and 70% will be normal repair. Termination condition of these experiments is that 1000 sequential generations have the same fitness function number.

Algorithm will generate 100 random deployment plans as the initial genome encoding. In these results the one with best fitness function number (Initial Network Cost, IC) is selected to compare with final result (Optimized Network Cost, OC). Optimize rate (IC-OC)/IC and the total runtime (in millisecond) of the algorithm are also listed in Table 2.

Table 2 shows that the optimization algorithm has very good effect in all scenarios. Especially in scenario 2 and 3 there are more than 30% reduce of total network cost. This is because when services number is not too large they will located much tight in a few domains and thus reduce the cross network costs. This can also be proved by scenario 4 when services number grows to a larger number, the optimize rate becomes much lower.

Scenario 1 have a lower optimization effect than 2 and 3 is because both the services number and domain number are quite small; the best initial network cost in 100 random deployments could be very close to the optimized result.

The last column shows that running time of the algorithm will be significantly increased with the rising of component services number as well as domain numbers.

However, since this is a static optimization algorithm and runs only for one time, it will not affect the dynamic performance of composite services and the performance should be acceptable.

HEURISTIC REPAIR VS NORMAL REPAIR

This part evaluates the influence of different heuristic rate to the optimization efficiency. We take Scenario 3 as a base line and change the heuristic parameter to 0% (no heuristic) and 100%(full heuristic) accordingly.

Axis X is the offspring generations and axis Y is the total network cost for each generation; it compares initial 3000 generation for 3 different heuristic rate parameters. Figure 5 show that bigger heuristic rate will make the algorithm approach to the best solution faster. Moreover, in all 3 cases, the algorithm has a more notable effective in the incipient 600 generations.

However, since heuristic repair has a higher algorithm complexity than normal repair, it will cost more time to reach final result. Total time cost for 3 cases are: 10453, 14890 and 19750 ms correspondingly.

CONCLUSION

In a cross domain multiple networks environment deploying component services into different domains will have different overall network cost and thus affect the performance of composite web services significantly because of the network condition between network domains are different from each other.

In this study, we proposed a composite services performance optimization problem under multiple networks environment. To solve this problem, we presented a mixed decentralized orchestration model which takes advantages from both centralized and decentralized orchestration. Then we proposed the concepts of services interaction and cross domain network cost. Finally we designed a mixed repair genetic algorithm to find the optimum deployment plan for all component services which has the lowest cross domain network costs.

The evaluation program shows that our algorithm has very good optimization efficiency and the algorithm itself has an acceptable performance.

The system presented in this study has some limitations as well. For example, composite services are not invariable in the enterprise. New business processes will be added and old ones might be removed. Component services could also be added, updated, removed or change to other alternative ones. Thus, the services interactions are changing dynamically and original deployment plan may not be the optimal choice. A basic way to solve this problem is to set a threshold value; if the overall cost exceeds this value, the optimize algorithm

will be retriggered and deployment location of services will be adjusted accordingly.

The next step of our work is to design an algorithm to find minimum changing plans when optimize algorithm is retriggered since cost of redeployment of services is very high.

REFERENCES

Binder, W., I. Constantinescu and B. Faltings, 2006. Decentralized orchestration of composite web services. Proceedings of 2006 IEEE International Conference on Web Services, Sept. 18-22, Chicago, Illinois, USA., pp: 869-876.

Chafle, G.B., S. Chandra, V. Mann and M.G. Nanda, 2004. Decentralized orchestration of composite web services. Proceedings of the 13th International Conference World Wide Web (WWW), May 17-20, ACM, New York, USA., pp: 134-143.

Chafle, G., S. Chandra, N. Karnik, V. Mann and M.G. Nanda, 2007. Improving performance of compositeweb services over a wide area network. Proceedign of the IEEE Congress on Services, July 9-13, Salt Lake City, pp: 292-299.

Cui, X.X. and C. Lin, 2004. Multicast QoS routing optimization based multi-objective genetic algorithm. J. Comput. Res. Dev., 41: 1144-1150.

Curbera, F., M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, 2002. Unraveling the web services web: An introduction to SOAP, WSDL and UDDI. IEEE Internet Comput., 6: 86-93.

Dyachuk, G.D. and R. Deters, 2007. Optimizing performance of web service providers. Proceeding of the 21st International Conference on Advanced Networking and Applications, May 21-23, Niagara Falls, pp: 46-53.

Ericsson, M., 2007. The effects of XML compression on soap performance. World Wide Web, 10: 279-307.

Hu, J.J., C.J. Tang, L. Duan, J. Zuo, J. Peng and C.A. Yuan, 2007. The strategy for diversifying initial population of gene expression programming. Chinese J. Comput., 30: 305-310.

Martello, S. and P. Toth, 1987. Algorithms for knapsack problems. Ann. Discrete Math., 31: 70-79.

Menasce, D.A., 2002. QoS issues in web services. IEEE Internet Comput., 6: 72-75.

Mullaley, M., 2005. The Impact of SOA and Web Services on Wide-Area Networks. http:// au.sys-con.com /node/164551.

Nanda, M.G., S. Chandra and V. Sarkar, 2004. Decentralizing execution of composite web services. ACM Sigplan Notices, 39: 170-187.

OASIS Standard, 2007. Web Services Business Process Execution Language Version 2. OASIS Standard. 11 April 2007. http://docs.oasis-open.org /wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

Papazoglou, M.P., 2007. Service oriented architectures: Approaches, technologies and research issues. Vldb J., 16: 389-415.

Patcas, L.M., J. Murphy and G.M. Muntean, 2005. Middleware support for data-flow distribution in web services composition. The PhDOOS Workshop and Doctoral Symposium. http://citeseerx.ist.psu.edu /viewdoc/ summary?doi=10.1.1.104.4015.

Ran, S., 2003. A model for web services discovery with QoS. ACM SIGecom Exchanges, 4: 1-10.

Renders, J.M and S.P. Flasse, 1996. Hybrid methods using genetic algorithms for global optimization. IEEE Transactions Syst. Man Cybernetics, Part B, 26: 243-258.

Yin, K., B. Zhou, S. Zhang, H. Jiang and J. Cristoforo, 2010. Optimizing services composition in multi-network environment. Inform. Technol. J., 9: 399-411.

Zeng, L., B. Boualem and D. Marlon, 2003. Quality driven web services composition. Proceedings of the 12th International Conference on World Wide Web, May 20-24, Hungary, pp: 411-421.

Zhang, S., J. Sun, D. Lu, Y. Shen and J. Aleksander, 2010. Performance optimization for composite services in multiple networks environment. Proceedign of 6th World Congress on Services, July 5-10, Miami, Florida, pp: 183-184.