QoS-Aware Ubiquitous Service Composition Based on Linear Logic Inference Rules

http://dx.doi.org/10.3991/ijoe.v9iS2.2570

H.B. Sun[1,2] and C.J. Zhou[2]
[1] Changchun Institute of Technology, Changchun, China
[2] Jilin University, Changchun, China

***Abstract*—Ubiquitous service modeling and composition is constrained and communication among nodes is error-prone and unreliable. Such a dynamic environment requires a continuous adaptation of the composition of services. The paper proposes a method for semantic message matching in automatic service composition. Since the service interface definition can be represented by ontology concepts, the internal representation language enables us to define some issues required by service composition formally, qualitative and quantitative constraints plus reasoning on concepts, and the service behavior can be represented using linear logic formulas, so the inference rules of linear logic can check the match-ability and satisfy-ability of service message. the mobile agent uses projection-join closure to capture such message situations. the optimization of the composition is translated into dynamic services selection with QoS global optimization. The simulation results show that the approach can significantly improve the Ubiquitous service performance. It adapts well to the changes of dynamic environments.**

***Index Terms*—Ubiquitous service Composition; QoS-Aware, Linear Logic; mobile agent.**

I. INTRODUCTION

A wireless sensor network is composed of a large number of sensor nodes that are densely deployed either inside the phenomenon or very close to it. The main objective of the wireless sensor networks is to observe an environment, collect information about the observed phenomena or events and deliver this information to the application.

The true potential of services can only be achieved if services are used to dynamically compose some new services that provide more sophisticated functionalities compared to existing ones. Service composition architectures should be able to utilize the spatial distribution of services to optimize service composition and execution. Fault management strategy has to take into consideration network level disconnection, service discovery failures, and service execution failures. Also, agent-based techniques [1] have been proved to be feasible to realize the automatic systems of services. Agents are envisioned for automatic discovery, execution, and integration of services [2]. However, no mature method has been proposed to manage birth, death, migration, stability, and communication processes of enormous agents. They cannot meet with autonomous management, evolution, and adaptation of the next-generation service [3].

The service composition is a highly complex task, and it is already beyond the human capability to deal with the whole process manually. Some methods for automatic composition and management of services have been proposed. They are conducted to fall into the realm of workflow composition or artificial intelligence (AI) planning methods. The workflow methods are mostly used in the situation where the request has already defined the process model. The AI planning methods are used when the requester has no process model but has a set of constraints and preferences.

In [4] use OWL-S language to describe the web services with their inputs and outputs. MARIO utilizes tags chosen by the user to provide possible composition schemes[5]. Service equivalence is presented to replace services in a mobile network where the connections between nodes are changing rapidly [6]. a dynamic web service composition method is proposed that considers quality of service (QoS) and network characteristics[7], Automatic Path Creation service is centralized and looks for a shortest path from the end-user to the primitive services. In sensor networks, few approaches have been proposed for service composition. A significant one is [8, 9] in which the authors provide a method based on logical programming through backward chaining for combining services. The method is used for automated inference in sensor networks. Another paper [10] tries to identify the service composition that is less likely to be invalid in the near future due to nodes going to sleep mode etc. The goal is to minimize the composition cost at a later time. In [11], the authors propose a dynamic flow control solution, applicable to sensor networks, which uses filters and wires between services. [12] proposes abstract task graphs that consist of abstract tasks and abstract channels. These are mapped to services (nodes) and possible connections (edges) in the service graph, In [13]. The authors present MiLAN, which is a middleware for sensor applications. It receives the application requirements and chooses a set of sensors that can provide this information according to certain quality of service requirements. In [14], A service composition selection method is presented based on sharing routing in wireless sensor networks . Rao et al. [15] proposed a method for automatic composition of Semantic Web services using Linear Logic (LL) [16] theorem proving. The services are presented by extralogical axioms and proofs in LL. A process calculus to present the process model of the composite service is used.

Our method can be regarded as an extension of the service composition method using Structural Synthesis of Programs proposed. The strength of the affinities indicates the usefulness of the relationship. The measure of affinity is based on the matching strength of agents, service quality score, and the trust. The simulation results show that our approach can well adapt to different scenario, including the changes of dynamic environments as well as partial failure of agents and nodes.

II. THE CONTEXT-AWARE METHOD OF UBIQUITOUS SERVICES COMPOSITION

A. *Linear Logic-Based Implementation of Service Composition*

As an atomic unit of Ubiquitous service composition, the aware agent (mobile agent) includes three modules: Attributes describe the characteristic of an agent itself. Function is designed to evaluate the matching ability of the message to the other mobile agents. Behavior contains interface operation, information issue, and energy transmission. The ideal model would place the platform on every device as a network node, such as mainframe, workstation, computer cluster, and PDA. [17, 18].

Service composition is the cooperation among service resources. Each agent should negotiate with others based on the capabilities that can be executed. We take advantage of full intuitionist LL. To use LL theorem proving as service composition negotiation is that LL is resource conscious logic. We can distinguish the information transformation and the state change produced by the service. Meanwhile, we can perform planning by using both qualitative and quantitative non-functional attributes. Because of soundness of the logic fragment, the correctness of composite services is guaranteed with respective to the initial specifications. Completeness of the logic fragment ensures that all compassable solutions can be found.

The service profile can be translated into LL axioms and LL sequences. In OWL-S, the information about Web services is presented by OWL-S classes and properties. They are translated into LL propositions referring to the specific classes and properties. The meaning of the propositions and the semantic relationships among the propositions are defined by the ontology relationships. After the service profile is translated into LL axioms and LL sequences, the next step is the negotiation among the agents in LL.

Negotiation is an interactive process involving partial deduction and LL theorem proving. Partial deduction is applied as a method of deducing sub problems. Generally, a request to a composite service (including functionalities and non-functional attributes) can be expressed by the following LL formula:

$$\Gamma_a, \Gamma_b;\ \Delta_c\ \ |-\ ((I \otimes P)\ -\!\circ\ (O \otimes F) \otimes E)) \otimes \Delta_n \qquad (2)$$

where, Γ_a and Γ_b are sets of extra logical axioms representing available value-added Web services and core services, respectively. Δ_c is a conjunction of non-functional constraints. Δ_n is a conjunction of non-functional results. $\otimes$ is multiplicative conjunction. For example, $A \otimes B$ denotes that the literals A and B are consumed or achieved simultaneously. $-\!\circ$ is linear implication. For example, $A -\!\circ B$ means that the goal B is achievable only when resource A is available. A LL sequence is divided into two parts by symbol $|-$. For example, $A\,|-B$ means that the goal B can be achieved by consuming the resource A. $(I \otimes P) -\!\circ (O \otimes F) \otimes E)$ is a functionality description of the required service. I represent a set of input parameters for the service and O represents a set of output parameters produced by the service. P and F are multiplicative conjunctions of preconditions and effects, respectively. E presents an exception. Intuitively, the formula can be explained as follows: Given a set of available services and non-functional attributes, we try to find a combination of services that computes O from I as well as that changes the world state from P to F.

Partial deduction steps as inference figures are defined in LL. While using these inference figures instead of basic LL rules, we can achieve more efficient proof search and higher efficiency. Partial deduction is known as one of optimization techniques in logic programming. Its basic idea is as follows: Given a specification, partial deduction derives a new (more specific) specification while preserving the meaning of the original one. We can use partial deduction to extract the maximum information from incomplete knowledge in the sense of the following specialization inference rule. Following is the corresponding functional specification of what a rule specialization process is. The extension to specialization of the agent's bases is straight forward formula (3).

$$S:\ B \to B$$

$$S(a) = \begin{cases} S((R - \{r\} + \{r'\}, P + \{p'\}) & (*)) \\ a & otherwise \end{cases} \qquad (3)$$

$$(*)\ \ if\ \ P \neq \phi\ \ and\ \ \exists p \in P\ \ and\ \ \exists r \in R$$

$$such\ \ that\ \ S_R(r, p) = (r', p')\ \ and\ \ r' \neq r.$$

where, B is a set of agents. We note agents as pairs a=(R, P), where R is a set of LL inference rules and P is a set of literals (agent states). In other words, the specialization of a agent's rule base consists of the exhaustive specialization of its rules. Rules that only have one condition appearing in the set of literals will be eliminated and a new literal will be added. This new literal will be used again to specialize the agent. The process will finish when the agent has no rule containing on its conditions.

The following LL inference rules, $R_b(L_i)$ and $R_f(L_i)$, are defined for partial deduction back (4) and forward (5) chaining steps, respectively.

$$\frac{S\ |-B \otimes C}{S\ |-A \otimes C} R_b(L_i(\underline{\lambda})) \qquad (4)$$

$$\frac{A \otimes C\ |-G}{B \otimes C\ |-G} R_f(L_i(\underline{\lambda})) \qquad (5)$$

where A, B, and C are LL formulae. L_i is a labeling of a particular LL axiom representing a agent's capability. $R_b(L_i)$ and $R_f(L_i)$ apply clause L_i to move the initial state towards the goal state or the other way around. In formulae (4), $A \otimes C$ and $B \otimes C$ denote goals G and G', respectively. It encodes that, if there is an extralogical axiom $|-B -\!\circ A$, then goal G can be changed to G'. In formulae (5), $B \otimes C$ and $A \otimes C$ denote state S and S', respectively. It encodes that, if there is an extralogical axiom $|-B -\!\circ A$, then initial state S can be changed to S'.

Additionally, we assume that $\underline{a} \overset{def}{=} a_1, a_2 \ldots\ldots$ is an ordered set of constants, $\underline{\lambda} \overset{def}{=} \lambda_1, \lambda_2 \ldots\ldots$ is an ordered set of variables, $[\underline{a}/\underline{\lambda}]$ denotes substitution, and $\lambda = \lambda'[\underline{a}/\underline{\lambda}]$. When substitution is applied, elements in $\underline{a}$ and $\underline{\lambda}$ are mapped in the order they appear in the ordered sets. These sets must have the same number of elements. The implementation of LL negotiation among the agents is based on Lygon. Lygon is a LL-based logic programming language, and can be viewed as Prolog extended with features derived from LL. These features include a clean declarative notion of state and the ability to express problems involving concurrency. As an abstracted language framework, Lygon can be implemented and expended in Java.

The application of Lygon to agent-oriented system is a new aspect. Since Lygon is suitable for concurrent programming, modeling actions, representing states, and searching, it is natural to use Lygon for working with the WSES. All LL inference rules can be implemented in Lygon. Lygon uses top-down computation, that is, a computation begins with a goal and seeks to prove it using the program. In our work, Lygon (version 0.7) written in fairly standard Prolog is used and should be easy to port to other Prolog systems. Lygon syntax is described in Table 1.

TABLE I.
GRAMMAR FOR THE LYGON.

G::=G $\otimes$ G| G $\oplus$ G| G & G G ⅋ G !G |negD|1|$\perp$|$\top$|A| negA
D::=[linear](A1 ⅋ A2 ⅋ … ⅋ An<-G)
Top 1 One
Bottom 0 Zero
& Provable in any context
$\otimes$ Provable only in empty context
& Cannot be proved, but can be weakened away
$\oplus$ Not provable

When an agent has to compose a new service, sub-services are generated. The sub-services are distributed among the partners and treated as offers to other agents. Semantic descriptions of the existing services are translated into extra logical axioms of LL, by applying partial deduction to find partial solutions. Ontology is used to reason over the Semantics of Web services' inputs and outputs. Partial solutions can be extended through our service emergent framework until a complete solution to be found

B. Service Quality Score and Affinity Strength

A threshold of service composition service quality score depends on three factors, Latency, Availability, and Cost. $Q_{bio-entity} = (Q_1(Latency), Q_2(Availability))$. Given a process in a service emergence, there is a set of candidate agents, which represent pervasive candidate services $S_{bio-entity} j = \{s_1 j, s_2 j \ldots\ldots s_n j\}$ that can be used to execute this task. By merging the quality vectors of all these services, a matrix $Q_{bio-entity} = (Q_{ij};\ 1 \le i \le n, 1 \le j \le 3)$ is built, in which each row Q_j corresponds to a service s_{ij} while each column corresponds to a quality dimension. A Simple Additive Weighting (SAW) [19] technique is used to evaluate service.

There are two phases in applying SAW. Some of the criteria could be negative, i.e., the higher the value, the lower the quality. This includes criteria such as time and cost. Other criteria are positive, i.e., the higher the value, the higher the quality. For negative criteria, values are scaled according to (3). For positive criteria, values are scaled according to (4).

$$P_{ij} = \begin{cases} \dfrac{Q_j^{\max} - Q_{ij}}{Q_j^{\max} - Q_j^{\min}} & if\ \ Q_j^{\max} - Q_j^{\min} \ne 0 \\ 1 & if\ \ Q_j^{\max} - Q_j^{\min} = 0 \end{cases} \tag{3}$$

$$P_{ij} = \begin{cases} \dfrac{Q_{ij} - Q_j^{\max}}{Q_j^{\max} - Q_j^{\min}} & if\ \ Q_j^{\max} - Q_j^{\min} \ne 0 \\ 1 & if\ \ Q_j^{\max} - Q_j^{\min} = 0 \end{cases} \tag{4}$$

Where $Q_j^{\max} = Max(Q_{ij}), 1 \le i \le n.$ $Q_j^{\min} = Min(Q_{ij})$, $1 \le i \le n$. By applying these two equations on Q, we obtain a matrix $P = (P_{ij}; 1 \le i \le n, 1 \le j \le 3)$ in which each row P_j corresponds to a bio-entity s_{ij}, while each column corresponds to a quality dimension.

The following formula (5) is used to compute the overall quality score for each agent,

$$S_{Bio-entity}(s_i) = \sum_{j=1}^{3} (P_{ij} * W_j) \tag{5}$$

Where $W_j \in [0,1]$ and $\sum_{j=1}^{3} W_j = 1$, W_j represents the weight of criterion j.

Trust-based reconstruction of relationship makes the necessary preparations for a more efficient service emergence. The adjustment of trust indicated by a user received the service. Whenever a service is provided, agents adjust the trust with their interaction partners based on the level of user satisfaction or happiness. If the user is satisfied (not satisfied) with the service, the trust is strengthened (weakened).

A trust could be a reward or a penalty, which indicates that the degree of a user's preference to emergent service. This message is propagated through the same path where the discovery service has been originally forwarded. When an intermediate agent on the path receives message, it adjusts the trust value of the relationship that has been used to forward the original discovery request. Trust value is increased for a reward, and is decreased for a penalty.

Given a value $R \in [-\frac{1}{2} \le R \le 1]$ that contains in a defray message, the agent ΔS updates the trust value of $Trust_{ij}$ using the formula (6).

$$Trust_{ij} = Trust_{ij} + \Delta S \tag{6}$$

$$\Delta S = \begin{cases} (1 - Trust_{ij}) * R^2 & R \ge 0 \\ Trust_{ij}(1 - \dfrac{1}{1+R}) & R < 0 \end{cases}$$

$$R \in [-\frac{1}{2} \le R \le 1]$$

Where $Trust_{ij}$ is a number in [0,1] that represents relationship value. Correspondingly, ΔS is calculated based on the $Trust_{ij}$ and R. We have chosen the above formula with the purpose of remarking ratings rise slowly and fall quickly.

The affinity between two agents is calculated based on the matching strength LL^{ws}_{agent} , service quality score $S_{Bio-entity}$ and trust $Trust$ of bio-entities. It define as

$$Aff_{agnet} = (1-\alpha)LL^{ws}_{ASgent}(A_{s1}, A_{s2}) \cdot S_{Bio-entity}(s2) + \alpha \cdot Trust \qquad (7)$$

$\alpha \in [0,1]$ is a weight to affinity. The optimal local affinity can help bio-entity decide where to forward.

If request of service composition is CS_{req} , according the rule of optimal local affinity, a user service is composed a set of agents, such as $CS_{s1}, CS_{s2}, CS_{s3} \ldots CS_{sn}$, Aff_{Agent} is the satisfaction value of service composition.

$$Aff_{Agent} = Aff_{Agent}(CS_{req}, CS_{S1}) \cdot \prod_{i=1}^{n-1} Aff_{Agent}(CS_{Si}, CS_{Si+1}) \cdot Aff_{Agent}(CS_{Sn}, CS_{req}) \qquad (8)$$

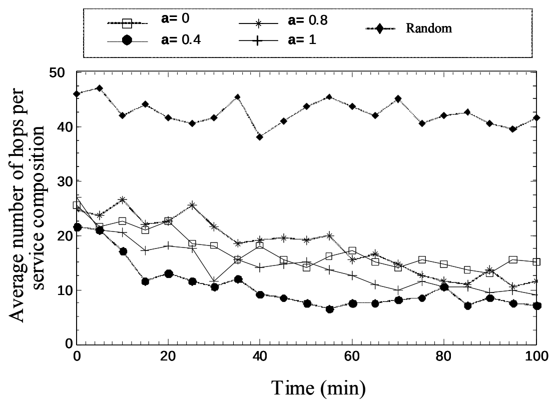
A threshold of service composition θ will be specified ($0<\theta<1$). If $Aff_{Agent} > \theta$, service composition is successful.

III. SIMULATION AND DISCUSSION

In our pervious work, we have implemented the prototype service simulation platform, including software, general objects and simulators in java. It supports pluggable functions and provides a generic easy-to-use programming API. The platform is initialized on all network hosts. The frequency of user request is 10 times per second, and simulation time begins from 0 to 100 minutes. Because there is no standard testing data sets, the interface matching and QoS data of service are randomly generated, which are assigned to the agents for testing the characteristics of service composition. We make a set of common service resources (which contains 1000 different resource vectors), and use 100, 200, 300, 400, 500 random services to evaluate the performance. We repeat the experiments multiple times. The results given out are the averaged values of measurements.

First experiment on performance is shown with threshold θ=0.8, the number of agents is 200 (Fig.1). The simulation evaluates the adaptation and evolution from two aspects: response time per service composition, average number of hops per service composition. Response time represents the efficiency of Service composition and average number of hops represents cost. We also give out the preference measurement of a random agent without operation compared with LL method.

In all 100 minutes, response time keep constant for random agent. It has a poor performance. In LL method, affinity relationships are random, and matching performs poorly at first, response time is 550ms. Obviously, much response time need to be visited to hit the target agent. Then, agents utilize adjustment of the affinity network, and meet desirable requirements and dynamic management of service. Thereby the response time decrease dramatically. We have a further discussion on the performance distinctions affected by different values of α (i.e. α=0, 0. 4, 0.8, 1), response time of α=0 is smaller than the value of α=0.4 during 20 minutes. With time passing by, the response time of α=0.4 is about 240 ms, which achieves the minimum value. Average number of hops represents the cost of service composition. Fig.1 also shows that the random method has the highest cost. Its average number of hops is 47 hops in 100 min, which is about 3 times of that as α=0.4 in the same experiment environment. The comparison results have proved that our approach can form clusters and improve the performance and adaptation.



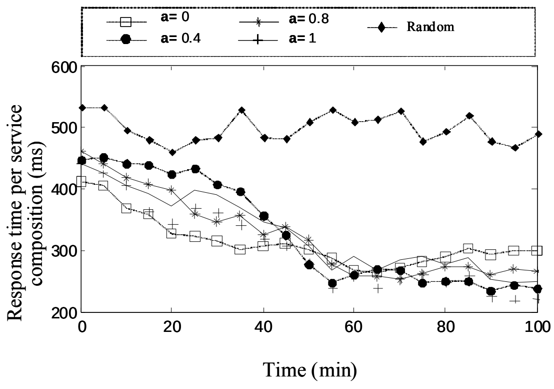


Figure 1. Response time and hops of service composition

TABLE II.
RESPONSE TIME DIFFERENCE IN UNRELIABLE ENVIRONMENT

Numbers of agents	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0.8$	$\alpha = 1$
100	6.2	6.4	6.4	6.3
200	5.3	5.6	5.3	5.2
300	5.1	5.4	5.5	5.2
400	4.9	4.7	4.4	4.5
500	4.1	3.9	3.9	3.7

We experiment on the adaptability of the approach in the unreliable condition. We adopt the same simulation environment as Fig. 2 to conduct a comparison. We chose 1% of agents and 1% of topological nodes randomly in the simulation and set them unavailable. A measure of the adaptability is measured as formula (9)

$$AD = \frac{T_{res-time}^{unreliable} - T_{res-time}^{reliable}}{T_{res-time}^{reliable}} \qquad (9)$$

Where $T_{res-time}^{unreliable}$ and $T_{res-time}^{reliable}$ are respectively the average response time per service composition for a certain scenario. The average result is given with the number of agents respectively as 100, 200, 300, 400 and 500. The experiment on adaptability to unreliable network shows the performance of our approach in the dynamic scenario. With the number of service increasing from 100 to 500, the response time appears small gradually, which suggests that our approach could be survivable in the environment with unreliable network.

IV. CONCLUSIONS

Sensor networks present a challenging programming environment because of their limited resources, heterogeneity and highly dynamic nature. Such a dynamic environment requires a continuous adaptation of the composition of services. The method of QoS-Aware Ubiquitous service Composition is presented based on Linear Logic Inference Rules, qualitative and quantitative constraints plus reasoning on concepts, and the service behavior can be represented using linear logic formulas, so the inference rules of linear logic can check the match-ability and satisfy-ability of service message. the mobile agent uses projection-join closure to capture such message situations. The optimization of the composition is translated into dynamic services selection with QoS global optimization.

The next work is on issue about improving the efficiency of the Linear Logic matching. Usually, the amount of the available services and the size of ontology models are huge. Therefore, it is necessary to reduce the search space during problem solving. In addition, more experiments will be designed to evaluate the method in service composition and management.

ACKNOWLEDGMENT

This work was supported in part by the National High Technology Research and Development Program of China (863Program, No.2012AA040104), A Project Supported by Scientific Research Fund of Jilin Provincial Education (20120268). A Project Supported by Scientific and Technological Planning Project of Jilin Province (20120332). A Project Supported by Scientific and Technological Planning Project of Jilin Province (20100565).

REFERENCES

[1] N. R Jennings. “An agent-based approach for building complex software systems”, *Communications of the ACM*, Vol.44, No.4, 2001, pp.35-41. http://dx.doi.org/10.1145/367211.367250

[2] M.J.Grady and G. M. P Hare. “Mobile devices and intelligent agents—towards a new generation of applications and services”, *Information Sciences*.Vol.171,No.4,2005,pp. 335-353. http://dx.doi.org/10.1016/j.ins.2004.09.009

[3] H.Zhuge. “The Future interconnection environment”, *IEEE Computer*, Vol.38,No.4, 2010,pp. 27-33. http://dx.doi.org/10.1109/MC.2005.142

[4] E.Sirin, B.Parsia, J.Hendler. “Composition-driven Filtering and Selection of Semantic Web Services”, *In AAAI Spring Symposium on Semantic Web Services*, 2004.

[5] A.V.Riabov, E.Bouillet, M.D.Feblowitz. “Wishful Search: Interactive Composition of Data Mashups”, *in WWW Conference,* Beijing, China, 2008. pp. 775-784,

[6] D.V Thanh, I.Jorstad. “A Service-Oriented Architecture Framework for Mobile Services”, *Proceedings of IEEE Telecommunications*. 2005, pp. 65-70.

[7] Z. M.Mao, R. H.Katz, E.A.Brewer .“Fault-tolerant, Scalable, Wide-Area Internet Service Composition”, *Technical Report UCB/CSD-01-1129, EECS Department, University of California, Berkeley, California,* USA, 2001.

[8] L. Mottola and G. P. Picco. “Programming wireless sensor networks: Fundamental concepts and state of the art,” *ACMComput. Surveys,* Vol. 43, 2011, pp.1−51 http://dx.doi.org/10.1145/1922649.1922656

[9] K.Whitehouse, F.Zhao,J.,Liu.“ Semantic Streams: a Framework for Composable Semantic Interpretation of Sensor Data”, *EWSN,* 2006, pp.5-20.

[10] X.Wang,J.Wang, Z.Zheng, Y.Xu, M.Yang.“Service Composition in Service-Oriented Wireless Sensor Networks with Persistent Queries”, *Consumer Communications and Networking Conference, Las Vegas,* NV ,2009, pp. 1-5.

[11] A.Bamis, N.Singh, A.Savvides.“An Architecture for Dynamic Reconfiguration of Data Flows in Sensor Networks”, *Technical Report, ENALAB, Yale University*, 2007.

[12] A.Bakshi,V. K Prasanna, J.Reich , D.Larner.“The Abstract Task Graph: A methodology for architectureindependent programming of networked sensor systems”, *in Proc. Workshop on End-to-end, sense-and-respond systems, applications and services*, 2005.

[13] W.Heinzelman, A.Murphy, H.Carvalho, M.Perillo. “Middleware to Support Sensor Network Applications”, *IEEE Network Magazine* , Jan. 2004

[14] S. Liang. “A service composition selection method based on sharing routing in wireless sensor networks”, *International Conference on Advanced Intelligence and Awareness Internet,* 2011, pp. 46 – 49 http://dx.doi.org/10.1049/cp.2011.1425

[15] J.Rao, P.Kungas, and M. Matskin. Composition of Semantic Web services using Linear Logic theorem proving, *Information Systems.*Vol 31,No4, 2011,pp. 340-360. http://dx.doi.org/10.1016/j.is.2005.02.005

[16] J. S. Hodas and D. Miller. “Logic programming in a fragment of intuitionistic Linear Logic”, *Information and Computation* .Vol.110, No.2, 1994, pp. 327-365. http://dx.doi.org/10.1006/inco.1994.1036

[17] L. Gao, Y. S. Ding, H. Ying. “An adaptive social network-inspired approach to resource discovery for the complex grid systems”, *International Journal of General Systems,* Vol.35.2006.pp.347-360

[18] Y. S.Ding, H. B Sun, K. R. Hao. “A bio-inspired emergent system for intelligent Web service composition and management”, *Knowledge-Based Systems*. Vol.20. No.5. pp.457-465. http://dx.doi.org/10.1016/j.knosys.2007.01.007

AUTHORS

H.B. Sun obtained his B.S. and M.S. in Electrical Engineering from Huabei Electric University, China in 1991, 1997, respectively. He obtained his Ph.D. in Electrical Engineering from Donghua University, Shanghai, China in 2007. His current research interests include complex network systems, nature-inspired technologies, Web-services technologies and multi-agent systems (e-mail: win_shb@163.com).

C.J. Zhou is a Professor at the College of Information Sciences and Technology, Jilin University, China. He obtained his B.S., M.S. and Ph.D. degrees in Electrical Engineering from Jilin University, Shanghai, China in 1994, 1999 and 2003, respectively. His scientific interests include computational intelligence, network intelligence, intelligent decision-making. (e-mail: zhangfangyu11@163.com).

This article is an extended and modified version of a paper presented at the International Conference on Mechanical Engineering, Automation and Material Science (MEAMS2012), held 22-23 December 2012, Wuhan, China. Received 14 February 2013. Published as resubmitted by the authors 25 March 2013.