# 一种基于与或图的语义Web服务自动组合方法研究

卢锦运 张为群
(西南大学计算机与信息科学学院 重庆 400715)
(重庆市智能软件与软件工程重点实验室 重庆 400715)

**摘 要** 单个Web服务提供的功能有限,服务组合成为Web服务应用的一个重要研究方向。提出了一种基于与或图的语义服务自动组合方法。该方法为Web服务引入语义,能将服务搜索空间受限于服务组合与或图中,并能从服务组合与或图中选出最佳组合图,从而达到优化服务组合的目的。仿真实验证明,该方法提高了Web服务组合的成功率和效率。

**关键词** 语义Web服务,服务组合,与或图

## Method of Automatic Semantic Web Services Composition Based on AND/OR Graph

LU Jin-yun ZHANG Wei-qun
(College of Computer and Information Science, Southwest University, Chongqing 400715, China)
(The Key Lab of Intelligence Software and Software Engineering, Chongqing 400715, China)

**Abstract** Single Web service just provides limited functionality, Web services composition has become an important research aspect in Web services application domain. This paper proposed an approach based on AND/OR graph to compose semantic Web services automatically. It adds the semantics to Web services and has a smaller search space limited to the service composition AND/OR graph and it finds the best composition graph in the service composition AND/OR graph to achieve the purpose of optimizing services composition. The results of the extensive experiments show that this method improves both the successful and the efficiency ratio of Web services composition.

**Keywords** Semantic Web services, Services composition, AND/OR graph

## 1 引言

Web服务是基于Web的、自描述的、模块化的一种分布式计算模型,具有高度的互操作性、跨平台性和松耦合性的特点。近年来,随着Web服务技术的快速发展和应用,单个Web服务提供的功能有限,很多时候都不能满足用户的需求,如何快速、准确、高效地组合已有的分布在Internet中的各类服务,实现服务之间的无缝集成,从而满足用户的需求,已经成为一个迫切需要解决的问题。

近年来,国内外提出的Web服务组合方法大多可以分为3种[9]:基于工作流的组合、基于人工智能的自动服务组合、基于图搜索的自动服务组合。文献[6]提出了一种基于与或图的服务组合方法,但该方法存在以下不足:1) Web服务的定义缺乏语义,服务组合的成功率不高。2)需要人工参与评估,服务组合效率较低。3)缺少对服务质量的考虑,无法组合出最佳服务。

基于以上分析,本文提出了一种基于与或图的语义Web服务自动组合方法。该方法将基于图搜索与基于语义Web的服务自动组合方法融为一体,利用本体来标注Web服务的语义信息,采用倒叙查找的方法构建服务组合与或图,并以服务质量为依据在服务组合与或图中选取最佳组合图,进而根据最佳组合图组合出具有较好服务质量且能够满足服务请求的服务。

## 2 基本概念

本体是基于语义的服务发现的基础,本体的表示和推理对基于语义的服务发现具有重要影响。OWL-S(Ontology Web Language for Services)作为Web服务描述本体语言,其优点是从服务发现、服务交互和服务通信等不同侧重点描述了Web服务[7]。因此,本文采用OWL-S来描述语义Web服务。

**定义1** 语义Web服务表示为一个四元组 $ws=(N,I,O,Q)$,其中:

(1)$N$表示服务名称,作为服务的唯一标识;

(2)$I=\{i_1,i_2,\cdots,i_k \mid k\in N\wedge k\geqslant 1\}$是服务的输入对象集合,输入对象元素 $i_j(1\leqslant j\leqslant k)$用本体标注;

(3)$O=\{o_1,o_2,\cdots,o_l \mid l\in N\wedge l\geqslant 1\}$是服务产生的输出集合,输出对象元素 $o_j(1\leqslant j\leqslant l)$用本体标注;

(4)$Q$表示服务质量值,$Q=F(q_1,q_2,q_3,q_4,q_5)$,其中 $q_1\in(0,1)$表示服务费用,$q_2\in(0,1)$表示服务性能,$q_3\in(0,1)$表

到稿日期:2009-08-09 返修日期:2009-11-30 本文受重庆市自然科学基金项目(CSTC,2006BA2003)资助。

卢锦运 男,硕士生,主要研究方向为软件工程、Web服务、面向服务的计算,E-mail:jinyunlu@swu.edu.cn;张为群 男,教授,主要研究方向为软件工程、形式语言、软件测试。

示服务可靠性，$q_4 \in (0,1)$表示服务可用性，$q_5 \in (0,1)$表示服务声誉，$F$为综合性能函数。根据以上5个非功能属性求出服务的具体QoS数值(本文假定QoS值越小表示对应服务的质量越好)，服务$ws_i$的服务质量表示为$Q_{ws_i}$。

**定义2** 服务请求是对用户请求接口的抽象描述，可以表示为二元组$SR=\{I^r,O^r\}$，其中：

(1)$I^r=\{i_1^r,i_2^r,\cdots,i_m^r \mid m\in N\wedge m\geqslant 1\}$是用户提供的输入对象集合，输入对象元素$i_j^r(1\leqslant j\leqslant m)$用本体标注；

(2)$O^r=\{O_1^r,O_2^r,\cdots,O_n^r \mid n\in N\wedge n\geqslant 1\}$是用户请求的目标输出对象集合，输出对象元素$o_j^r(1\leqslant j\leqslant n)$用本体标注。

**定义3** 服务节点是指与具体服务相对应的节点，表示为$v_s$，所有$v_s$构成的集合为服务节点集合，表示为$V_s$($V_s=\{v_{s_i} \mid i\in N\wedge i\geqslant 1\}$)。

**定义4** 数据节点是指与具体服务的输入/输出对象相对应的节点，表示为$v_d$，所有$v_d$构成的集合为数据节点集合，表示为$V_d$($V_d=\{v_{d_i} \mid i\in N\wedge i\geqslant 1\}$)。

**定义5** 目标节点是为了本文方法能够适应服务请求中存在一个和多个输出对象的情况而引进的节点对象，表示为$S_n$，$S_n$为与节点且$Q_{S_n}=0$，其父亲节点集合$P(S_n)=\{o_s \mid o_s\in SR.O^r\}$

**定义6** 服务组合与或图是指用来表示Web服务输入/输出对象与Web服务之间依赖关系的与或图，可以表示为二元组SCAOG=$(V,E)$。其中：

(1)$V=\{v_1,v_2,\cdots,v_n \mid n\in N\wedge n\geqslant 1\}$是节点的集合，$V_d\subseteq V$，$V_s\subseteq V$，$S_n\in V$。

(2)$E$是$k$-连接符的集合，$k$-连接符表示$k$个节点集合$V_k=\{v_i,\cdots,v_{i+k-1}\}\subseteq V$指向节点$v_k$($v_k\in V\wedge v_k\notin V_k$)，$E$可表示为[5]$E=\{e_j \mid e_j=(f,v_k,V_k)\}$，$1\leqslant j\leqslant m$，其中，$v_k=f(v_i,v_{i+1},\cdots,v_{i+k-1})$，$f\in\{and,or\}$表示节点与节点之间的逻辑运算操作，$m$为SCAOG中边的总条数。

如果$V_d\subseteq v_s.I$，那么$V_d$中的所有元素都是$v_s$的父节点且$V_d$中的元素之间存在“与”的逻辑关系，对应的$k$-连接符$e=(and,v_s,V_d)$。另一方面，如果对于$\forall v_s\in V_s$均有$v_d\in v_s.O$，那么$v_s$为$v_d$的父节点且$V_s$中的节点之间存在“或”的逻辑关系，对应的$k$-连接符$e=(or,v_d,V_s)$。

SCAOG如果无法组合出满足服务请求的服务，那么显然就不存在最佳服务组合策略，因此本文的研究是以服务库中存在能够满足服务请求的原子服务或组合服务为前提的。在此前提下，为了获取最佳组合策略，本文应对SCAOG中的节点赋值，并通过对具有最小权值节点的选取来获取最佳组合策略。节点$v(v\in V)$的权值用$g(v)$来表示，带权值的节点$v$则表示为$v(g(v))$。假设用户的服务请求$SR$需要$m$个服务$ws_1,ws_2,\cdots,ws_m$组合而成，那么节点$SR$的权值$g(SR)=g(ws_1)+g(ws_2)+\cdots+g(ws_m)=\sum_{i=1}^{m}g(ws_i)$。以下为节点$v$的权值赋值策略：

① $v\in SR.I^r$，$g(v)=0$；

② $v$为UNSOLVED或UNKNOWN节点时，$g(v)=\infty$；

③ $v$为与节点时，$g(v)=\sum_i g(c_i)+Q_v$，其中$c_i$为节点$v$的父节点。

④ $v$为或节点且$v\notin SR.I^r$时，$g(v)=\min_i(g(c_i))$，其中$c_i$为节点$v$的父节点。

**定义7** 服务组合图，是指能够组合出满足服务请求服务的与或图，它为SCAOG的子图，可以表示为二元组SCG=$(V'',E'')$，其中，$V''\subseteq V$，$E''\subseteq E$。在SCG中，$S_n\in V''$；若$\forall v_m\in V''$且$v_m$为或节点，则$v_m$有且仅有一个父亲节点在SCG中；若$\forall v_m\in V''$且$v_m$为与节点，则$v_m$所有的父亲节点都在SCG中；SCG中的每条路径都是起于$i^r(i^r\in SR.I^r)$，终于$S_n$；只有$S_n$的祖先节点在SCG中。最佳组合图OCG=$(V',E')$，为具有最小的综合性能总值的SCG，即OCG中的$S_n$权值不大于其它SCG中的$S_n$权值。

## 3 基于SCAOG的自动服务组合

给定一个服务库$WS=\{ws_1,ws_2,\cdots,ws_n\}$和一个服务请求$SR=\{I^r,O^r\}$，基于SCAOG的服务自动组合方法分两步来实现：1)SCAOG的自动生成；2)最佳组合解图(Optimal Composition Solution Graph，OCSG)的选取。

### 3.1 SCAOG的自动生成

基于黑盒视图的服务发现策略通常认为，判定一个服务能否满足服务请求，首先需要检验该服务能否提供服务请求所需的所有输出；如果能，则再检验服务请求能否提供调用该服务所需的所有输入。因此，从服务请求的所有输出对象出发以倒序查找的方法生成服务组合与或图。

给定一个服务库$WS=\{ws_1,ws_2,\cdots,ws_n\}$和一个服务请求$SR=\{I^r,O^r\}$，可通过如下GSCAOG算法自动创建与服务请求相对应的SCAOG。

**算法1** GSCAOG(Generation of Services Composition AND/OR Graph)

输入：语义服务库$WS=\{ws_1,ws_2,\cdots,ws_n\}$，服务请求$SR=\{I^r,O^r\}$

输出：SCAOG=$(V,E)$

Step1 初始化优先队列$K=K\cup\{o_i^r\in SR.O^r\}$，创建目标节点$S_n$并且$V=V\cup\{S_n\}$，创建$k$-连接符$e=(and,S_n,\{o_i^r \mid o_i^r\in SR.O^r\})$且$E=E\cup\{e\}$。

Step2 若$K=\emptyset$，则转Step5；否则，处理$n=POP(K)$。

Step3 将未被访问过的$ws_i(ws_i\in WS\wedge n\in ws_i.O)$装进$V$并标记$ws_i$为已访问，$V=V\cup\{n\}$，$e=(or,n,\{ws_i \mid ws_i\in WS\wedge n\in ws_i.O\})$。

Step4 令$E=E\cup\{e=(and,ws_i,\{i_s \mid i_s\in ws_i.I\})\}$，将未被访问的$i_s\in ws_i.I$装入$K$并标记其为已访问，转Step2。

Step5 返回SCAOG=$(V,E)$。

GSCAOG算法从服务请求的所有输出对象出发以递归方式自底向上地生成SCAOG。算法总是可以在有限递归次数后终止，因为：1)在服务库中的服务个数、输入对象集合、输出对象集合的元素个数均为有限多个；2)通过对已访问过的节点做已访问的标记，避免了与或图生成过程中出现某些路径的重复生成；3)$K$中的元素个数有限，通过有限次循环$K$最终会因为元素的移除而变为空。

假设利用该算法生成的SCAOG中包含有$k$个节点，服务库$WS$中有$n$个服务，服务库中的服务包含的输入对象个数最多为$q$，两个本体概念间的语义相似度计算的时间复杂度为O(1)，节点的创建及节点之间依赖创建的时间复杂度均为O(1)，则该算法的时间复杂度为$O(k*n*q)$。

### 3.2 最佳组合路径的选取

给定一个服务库，利用GSCAOG算法可以构造出与服务

请求相关联的服务组合与或图 SCAOG，但是生成的服务组合与或图仅仅是由服务库中所有与服务请求相关的服务以及这些服务的输入输出节点之间的依赖关系所构成的服务依赖图，它并不能说明是否可以满足服务请求以及如何组合出满足服务请求的最佳合成服务。因此需要通过最佳组合图搜索算法在 SCAOG 中获取具有最小综合性能总值的服务组合图。

根据服务请求 $SR=\{I^r,O^r\}$ 以及由算法 GSCAOG 生成的 SCAOG，可通过如下 SOCG 算法得到 OCG，如果无法获取 OCG，则说明已有的服务库无法提供满足该服务请求的服务。

**算法 2** SOCG(Selection of the Optimal Composition Graph)

输入：SCAOG=$(V,E)$，服务请求 $SR=\{I^r,O^r\}$

输出：OCG=$(V',E')$

Step1 初始化优先队列 $K$ 的值为 $i_1^r \in r^q.I^r$。

Step2 若 $K=\emptyset$，则转 Step5；否则从 $K$ 中取出具有最小权值的节点 $n$。若 $n=S_n$，则转 Step5；否则获取 $n$ 的子节点集合 $J$。

Step3 若 $J=\emptyset$，则转 Step2；否则处理对象 $j\in J$。若 $j\in V_d$，则转 Step4；否则若 $j$ 含有标记为 UNSOLVED 的父亲节点，则标记 $j$ 为 UNSOLVED。若 $j$ 所有的父亲节点均为 SOLVED 节点，那么 $g(j)=Q_j+\text{sum}(g(j'\text{s parents}))$，同时标记 $j$ 为 SOLVED 且 $K=K\cup\{j\}$，$V'=V'\cup\{j\}$，$E'=E'\cup\{e=(and,j,\{j'\text{s parents}\})\}$。若 $j$ 含有标记为 UNKNOWN 的父亲节点，则 $K=K\cup\{j\}\cup\{j'$UNKNOWN 的父亲节点$\}$，转 Step2。

Step4 若 $j$ 所有的父亲节点均为 UNSOLVED 节点，则将 $j$ 标记为 UNSOLVED。若 $j$ 所有的父亲节点均为 SOLVED 节点，则 $g(j)=g(sp)(sp\in j'\text{s parents}\wedge(g(sp)=smallest(g(j'\text{s parents}))))$，$V'=V'\cup\{j\}$，$E'=E'\cup\{e=(or,j,\{sp\})\}$，标记 $j$ 为 SOLVED 且 $K=K\cup\{j\}$，若 $j$ 含有标记为 UNKNOWN 的父亲节点，则 $K=K\cup\{j\}\cup\{j'$UNKNOWN 的父亲节点$\}$，转 Step2。

Step5 若 $S_n$ 已标记为 SOLVED，则找到最佳组合图，返回 OCG=$(V',E')$；否则，无满足的组合服务。

SOCG 算法首先将服务请求的所有输入对象装入优先队列 $K$ 中，通过 $K$ 来维持一组待扩展的节点。算法循环从 $K$ 中取出具有最小权值的节点 $n$，并通过扩展 $n$ 获取其儿子节点集合 $J$。根据 $sn_i(sn_i\in J)$ 的类别及其父亲节点的状态，可以判断 $sn_i$ 的状态同时可以计算出 $sn_i$ 的权值，通过对目标节点状态的判断可知是否能满足服务请求，若满足可则通过节点的选取得出满足服务请求的 OCG。

假设算法中的 $t^*$ 含有 $v$ 个节点，因为 $K$ 为优先队列，所以在最坏情况下从 $K$ 中取出具有最小权值的节点所需要的时间为 $O(\log v)$，设 $t^*$ 中一个节点最多有 $f$ 个父亲节点，有 $s$ 个儿子节点，那么该算法的时间复杂度为 $O(v*(\log v+f*s))$。

## 4 仿真实验与评估

仿真实验中使用 Protege 建立和模拟领域本体，其中包括类、子类和层次关系等信息，本体中的概念个数为 50。服务库中登记了大量用 OWL-S 描述的服务，服务个数为 100～2000，服务的输入输出参数个数为 2～5，同时对服务的功能描述、过程模型及 QoS 进行设定，并且将服务的输入输出参数随机映射到领域本体的概念定义上。随机生成 100 个服务请求，服务请求的输入输出参数也采用上述建立的本体进行标注。仿真实验在 CPU 为 Intel Pentium Dual E2140 1.60G，内存为 1.00GB，操作系统为 Windows XP 的环境上运行。

将生成的 100 个服务请求分别用本文提出的组合方法与文献[6]中提出的方法进行处理，从算法的执行时间效率和算法执行所满足的用户请求数两方面进行比较，实验结果如图 3 和图 4 所示。

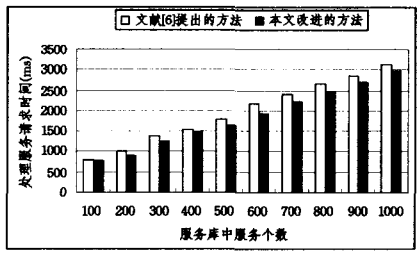


图 3 时间效率比较

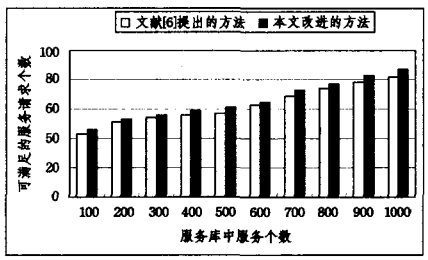


图 4 查找到可满足的用户请求数比较

仿真实验表明，在相同的实验条件下本文提出的方法与文献[6]的方法相比：1)服务组合的时间效率有所提高。从图 3 可知，在相同服务个数的前提下，本文方法处理服务请求的时间效率较文献[6]有所提高，这是因为该方法在生成服务组合与或图中采用倒叙查找的方法，在一定程度上提高了服务与或图的生成效率。2)服务组合的成功率有所提高。从图 4 可以看出，本文较文献[6]能够满足的服务请求个数有所增加，这是因为本文为 Web 服务引入语义，提高了服务的匹配效率，从而提高了服务组合的成功率。

**结束语** 本文提出了一种基于与或图的语义 Web 服务自动组合方法。该方法分两步完成服务自动组合过程，首先在已有服务库的基础上根据服务请求生成服务组合与或图，进而在服务组合与或图中选取最佳组合图。该方法为 Web 服务引入语义，实现了对象间的语义转化，从而提高了服务组合的成功率。此外，本文考虑了服务质量属性，通过对服务组合与或图中具有较好服务质量的服务节点的选取组合产生出具有最好 QoS 的合成服务。

## 参 考 文 献


[1] Oh Seog-Chan, On Byung-Won, Larson E J, et al. BF*: Web Services Discovery and Composition as Graph Search Problem [C]// IEEE International Conference on e-Technology, e-Commerce and e-Service. 2005:784-786

[2] Menascé D A. QoS issues in Web service [J]. Internet Computing, 2002, 6(6):72-75

[3] Majithia S, Shields M, Taylor I, et al. Triana: A Graphical Web Service Composition and Execution Toolkit[C]// IEEE International Conference on Web Services. 2004:514-524


(下转第 261 页)

[15] Ei-Moursy A, Albonesi D H. Front-End Policies for Improved Issue Efficiency in SMT Processors[C]// Proc. of the 9th International Symposium on High-Performance Computer Architecture. Anaheim, California, USA, February: 31-40

[16] Cazorla F J, Ramirez A, Valetor M, et al. Dcache Warn: an I-Fetch Policy to Increase SMT Efficiency[C]// Proc. of the 18th International Symposium on Parallel and Distributed Processing. Santa Fe, New Mexico, USA, April 2004: 74-83

[17] 孙彩霞，张民选. Dwarn+：一种改进的同时多线程取指策略[J]. 小型微型计算机系统，2007，28(7)：1720-1723

[18] Suh G E, Devadas S, Rudolph L. A New Memory Monitoring Scheme for Memory-Aware Scheduling and Partitioning[C]// Proc. of the 8th International Symposium on High-Performance Computer Architecture. Boston, Massachusetts, USA, February 2002: 117-128

[19] Ramirez T, Pajuelo A, Santana O J, et al. Runahead Threads: Reducing Resource Contention in SMT Processors[C]// 16th International Conference on Parallel Architecture and Compilation Techniques. Brasov, Romania, September 2007: 423-423

[20] Eyerman S, Eeckhout L. A Memory-Level Parallelism Aware Fetch Policy for SMT Processors[C]// Proc. of the IEEE 13th International Symposium on High Performance Computer Architecture. Phoenix, Arizona, February 2007: 240-249

[21] Shin C, Lee Seong-Won. Dynamic Scheduling Issue in SMT Architectures[C]// Proc. of the 17th International Parallel and Distributed Processing Symposium. Nice, France, April 2003

[22] Raasch S E, Reinhardt S K. Applications of Thread Prioritization in SMT Processors[C]// Proc. of the Workshop on Multithreaded Execution and Compilation. Orlando, Florida, January 1999

[23] Yamasaki N, Magaki I, Itou T. Prioritized SMT Architecture with IPC Control Method for Real-Time Processing[C]// Proc. of 13th IEEE Real Time and Embedded Technology and Applications Symposium. Bellevue, Washington, USA, April 2007: 12-21

[24] 何立强，刘志勇. 一种具有 QoS 特性的同时多线程处理器取指策略[J]. 计算机研究与发展，2006，43(11)：1980-1984

[25] Ma Pengyong, Chen Shuming, Hu xiao. Two methods to enhance the master thread's performance in SMT Chip[C]// 2007 IFIP International Conference on Network and Parallel Computing. Dalian, China, September 2007: 578-583

[26] Aamodt T M, Chow P, Hammarlund P, et al. Hardware Support for Prescient Instruction Prefetch[M]. Madrid, Spain, February 2004: 84-95

[27] Luo Kun, Ju J G, Franklin M. Balancing Throughput and Fairness in SMT Processors[C]// ISPASS. 2001

[28] 张盛兵，王晶. 同时多线程结构的线程预构[J]. 西北工业大学学报，2007，25(2)：159-163

[29] Wang Jing, Zhang Shengbing, Zhang Meng, et al. A Modified Instruction Fetch Control Mechanism for SMT Architecture[C]// 2007 IEEE Region 10 Conference. Taipei, Taiwan, October-November 2007: 1-4

[30] He Liqiang, Liu Zhiyong. An Effective Instruction Fetch Policy for Simultaneous Multithreaded Processors[C]// Proc. of the 7th International Conference on High Performance and Grid in Asia Pacific Region. Japan, July 2004: 162-168

[31] Robatmili B, Yazdani N, Sardashti S, et al. Thread-Sensitive Instruction Issue for SMT Processors [J]. IEEE Computer Architecture Letters, 2004, 3: 5

[32] Sharkey J J, Ponomarev D V. Efficient Instruction Schedulers for SMT Processors[C]// Proc. of the 12th International Symposium on High-Performance Computer Architecture. Austin, Texas, Feb. 2006: 288-298

[33] Sharkey J J, Ponomarev D V. Balancing ILP and TLP in SMT Architectures through Out-of-Order Instruction Dispatch[C]// Proc. of the 2006 International Conference on Parallel Processing. Columbus, Ohio, August 2006: 329-336

[34] Raasch S E, Reinhardt S K. The Impact of Resource Partitioning on SMT Processors[C]// Proc. of the 12th International Conference on Parallel Architecture and Compilation Techniques. New Orleans, Louisiana, Sept. - Oct. 2003: 15-25

[35] Cazorla F J, Ramirez A, Valero M, et al. Dynamically Controlled Resource Allocation in SMT Processors[C]// Proc. of the 37th International Symposium on Microarchitecture. Portland, December 2004: 171-182

[36] Choi S, Yeung D. Learning-Based SMT Processor Resource Distributing via Hill-Climbing[C]// Proc. of the 33rd International Symposium on Computer Architecture. Boston, MA, USA, June: 239-251

[37] Yang Hua, Cui Gang, Yang Xiaozong. Eliminating Inter-Thread Interference in Register File for SMT Processors[C]// Proc. of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies. Dalian, China, Dec. 2005: 40-45

[38] 杨华，崔刚，刘宏伟，等. 基于线程感知寄存器重命名的 SMT 处理器资源分配[J]. 2008，31(5)：845-857

（上接第 190 页）

[4] Hashemian S V, Mavaddat F. A Graph-Based Approach to Web Services Composition[C]// IEEE/IPSJ International Symposium on Applications and the Internet. 2005: 183-189

[5] Xie X Q, Chen K Y, Li J Z. A Composition Oriented and Graph-Based Service Search Method[C]// Asian Semantic Web Conference. 2006: 530-536

[6] Liang Qianhui, Stanley Y W S. AND/OR Graph and Search Algorithm for Discovering Composite Web Services [J]. International Journal of Web Services Research, 2005: 48-68

[7] Paolucci M, Wagner M. Grounding OWL-S in WSDL-S [C]// Proceedings of IEEE International Conference on Web Services. 2006

[8] Zeng L, Benatallah B, Dumas M, et al. Quality driven Web services composition[C]// Proc. of the World Wide Web Conference. 2003: 411-421

[9] 邓水光，吴健，李莹，等. 基于回溯树的 Web 服务自动组合[J]. 软件学报，2007，18(8)：1896-1910

[10] 邝砾，邓水光，李莹，等. 使用倒排索引优化面向组合的语义服务发现[J]. 软件学报，2007，18(8)：1911-1921

[11] 岳昆，王晓玲，周傲英. Web 服务核心支撑技术：研究综述[J]. 软件学报，2004，15(3)：428-442

[12] 石静，丁长明，赵泽宇，等. Web 服务合成研究综述[J]. 计算机科学，2004，31(6)：54-58

[13] 高亚春，张为群. 基于 QoS 本体的 Web 服务描述和选择机制[J]. 计算机科学，2008，35(12)：273-276

[14] 孙亮，任小康. 基于本体的图像语义检索模型[J]. 重庆工学院学报：自然科学版，2009，23(1)：127-131