基于图的 Web 服务组合优化的研究*)

曹利培[1] 刘 静[2] 缪淮扣[1]

(上海大学计算机科学与工程学院 上海 200072)[1]

(华东师范大学 软件学院计算机理论研究所 上海 200062)[2]

摘 要 单个 Web 服务难以满足实际应用的需求,如何组合已有的服务,形成新的服务,已成为此领域的研究热点。现在的组合方法极少考虑服务质量 QoS(Quality of Service)。对于一些提供相似功能的 Web 服务,服务质量是判断是否选择此服务的关键因素,组合服务的质量必须满足用户的需求。本文基于 SOA 的服务开发思想,针对当前服务组合存在的问题,提出了一种基于 QoS 的服务组合方法,并给出了构建基于 QoS 的 Web 服务组合及选择最佳服务的策略,通过整合单个服务的质量以得到最终组合服务的整体最佳质量。在满足用户组合服务的功能需求的同时,也满足了用户对服务质量 QoS 的需求,实现了需求服务的优化。

关键词 Web 服务, OWL-S, WSDG, 带权依赖图

Study on Optimization of Web Services Composition Based on Graph

CAO Li-Pei[1] LIU Jing[2] MIAO Huai-Kou[1]

(School of Computer Technology and Science, Shanghai University, Shanghai 200072)[1]

(Software Engineering Institute, East China Normal University, Shanghai 200062)[2]

Abstract Single Web service just provides limited functionality, and can't meet the needs in practice. How to compose existing services to form new services has become an important research in Web service domain. Current composition method seldom considers Quality of Service(QoS). For some services of providing similar functionality, QoS is key factor in deciding whether a service should be selected. Composite service must meet requirements of consumer for QoS. To improve the situation, based on SOA, this paper presents a new composition strategy considering QoS, by introducing the property of quality of single service so as to obtain final optimization of whole service in quality. How to select optimal service and how to construct QoS-based composition is described in details in the paper. Such service can not only meet the requirement in functionality but also QoS property, and it realizes service optimization.

Keywords Web service, OWL-S, WSDG, Weighted dependency graph

1 引言

随着互联网技术应用的迅速发展,互联网的应用模式也从最初的页面 Web、应用 Web,发展到 Web 服务[14]。近年来,Web 服务作为一种炙手可热的技术,已经被应用到企业的 IT 系统和商业流程之中。现在越来越多的企业将自己的业务作为 Web 服务发布,Web 服务受到了越来越多的重视。但它还存在一些问题:首先,传统的搜索引擎不支持客户希望的语义查找;其次,单个的 Web 服务常常不能满足客户的需要,客户需要的服务通常是由一系列的不同服务提供者提供的服务组成。这样,帮助用户发现和组合他们所希望的服务,提供增值服务就变得日益重要起来[3]。另外,随着 Web 服务应用的深入,企业对服务的速度,服务质量等都提出了更高的要求,也就是说面对越来越多的 Web 服务,如何快速发现、部署较高质量的服务[7,8],以满足客户个性化的需求,也是亟待解决的问题。

Web 服务是一种构建面向服务架构(Service-Oriented Architecture,SOA)[10]的分布式计算技术。SOA 是一种粗粒度、松耦合的服务结构。SOA 是服务的集合,服务通过基于标准、精确定义的接口进行通信。SOA 是在计算环境下设计、开发、应用、管理分散的逻辑(服务)单元的一种规范[10]。在本质上,Web 服务是一种自描述、模块化、由 URI 标识的应用程序,它采用 XML 和 Internet 的开放标准,支持基于 XML 的接口定义、发布和发现[22]。在现有的开放环境下,Web 服务只有通过组合成为更大粒度的服务,才能充分发挥其潜力和作用。Web 服务组合就是通过多个小粒度的 Web 服务相互之间的通信和协作来实现大粒度的服务功能,通过有效地联合各种不同功能的 Web 服务,服务开发者可以借此解决较为复杂的问题,实现增值功能[12]。

近年来国内外学术界和工业界围绕着服务组合开展了大量的研究工作,并发布了一些用来描述 Web 服务组合的语言,主要有微软提出的 XLANG[4],IBM 提出的 WSFL[5],BEA 公司的 WSCI[17],IBM、微软和 BEA 公司联合提出的 BPEL4WS[6,11],乔治亚大学提出的 DAML-S[9,21]等。但是第一代组合语言 XLANG、WSFL 和 WSCI 是不兼容的,因此几家公司联合开发出了第二代语言 BPEL4WS,它综合了前面

*)国家重点基金研究发展规划(973)(2002CB312001,2005CB321904)资助、国家自然科学基金(60373032)资助、上海市自然科学(05ZR14052)资助。**曹利培** 硕士研究生,主要研究方向为面向服务的架构、Web 服务;**刘 静** 博士,副教授,主要研究领域为软件工程、面向服务的架构、MDA、Web 服务、构件技术与形式方法;**缪淮扣** 教授,博士生导师,主要研究领域为软件工程、形式方法。

三者的特点，但仍然缺乏对语义的支持。对于服务的组合、部署仍然缺乏过程层的标准[24]，也并没有组合提供形式化的建模与分析手段，不利于对组合 Web 服务的性能评价[1]。表 1 通过几个方面对已有 Web 服务组合技术进行了比较。

从表 1 中的比较可知，只有 DAML-S 对语义匹配提供了支持，解决了传统搜索引擎不支持语义查询这一个问题，因此本文使用了 Web 服务本体语言(Web Ontology Language for Services，OWL-S)。OWL-S(以前称为 DAML-S)是一种基于本体的高层服务描述语言，可以实现自动服务发现、自动服务调用、自动服务组合和互操作及自动服务执行监控[21]。它主要想解决 Web 服务组合中服务的匹配和自动发现问题，对于服务质量的支持力度并不大。

表 1　Web 服务组合技术比较

提出者	模型描述	语义匹配	动态复合	异常处理	QoS 支持
微软	XLANG	无	无	有	无
IBM	WSFL	无	无	有	无
BEA	WSCI	无	无	有	无
IBM、BEA 及微软	BPEL4WS	无	无	有	无
乔治亚大学	DAML-S	有	支持	无	部分

Web 服务的服务质量 QoS(Quality of Service)问题是一个非常具有挑战性的问题。由于 QoS 对于 Web 服务的成功应用非常关键，因此如何提供具有 QoS 保证的 Web 服务正在引起人们的关注[23]。如何从大量的 Web 服务中动态地选择出最能满足用户需求的服务是目前需要解决的问题。本文基于 SOA 的思想，把语义 Web 技术作为技术支撑，将研究的重点提升到了“如何得到最佳服务”的层次上，提出了一种考虑服务的非功能特性因素的直观的方法，发现、组合 Web 服务，以满足客户个性化需求。

第 2 节介绍了本文提出的 Web 服务的形式化概念及详细算法描述；第 3 节是实例验证；最后是全文总结及未来展望。

2　基于图的 Web 服务组合

基于 SOA 架构从服务集成的角度来设计应用软件及复用现有的服务的要求，针对当前 Web 服务组合技术存在的问题，本文提出了一种建立在图上的 Web 服务组合算法。也就是利用带权图作为 Web 服务组合的基础，实现得到的服务最佳化。

2.1　基于图的 Web 服务的形式化定义

一般，图可以提供一种自然直观的方式描述实体之间的复杂交互，因此本文采用图的表示方法来表示服务之间的关系。在文[19]中提到了依赖图的概念。本文在此基础上提出了更进一步考虑 QoS 的服务发现组合。首先对 Web 服务进行形式化定义。

定义 1　一个依赖图 $G=(V,E)$ 是一个有向图，包括存储区中所有 Web 服务信息。其中 V 是节点集合，表示出现的输入/输出动作或消息。E 是边的集合，表示输入/输出动作或消息之间的依赖关系。假设节点 $v_x, v_y \in V$，那么在图 G 中从节点 v_x 到 v_y 有边当且仅当依赖集合中存在有依赖关系 $v_x \rightarrow v_y$。E 中的每条边都对应一个集合，该集合包含所有那些在依赖关系集中包含有这种依赖关系的 Web 服务，简称为 Web 服务依赖图(Web Service Dependency Graph，WSDG)。

例如：假设 Web 服务 $w1$ 的输入用节点 $s1$ 表示，输出用 $s2$ 表示，那么对于 $w1$ 我们可以图形化表示为图 1。

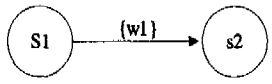

图 1　服务 $w1$ 的图形化表示

$\{w1\}$ 表示集合中只有一个服务满足这种输入、输出。如果有多个服务输入是 $s1$，输出是 $s2$，那么边上标注的集合中应该包含所有满足这种输入、输出要求的服务。

对于图中的每条边都有两个端点，因此我们可以把服务及其输入、输出形式化表示为一个序偶，其中的第一个元素表示服务的输入，第二个元素表示输出。例如服务 $w1$ 可以形式化表示为序偶 $\langle s1, s2\rangle$。

定义 2　设依赖图 $G=(V,E)$，如果有依赖图 $G'=(V',E')$，且 $E'\subseteq E, V'\subseteq V$，则称 G' 是 G 的依赖子图。

在 Web 服务组合中，根据用户的需求描述进行 Web 服务发现，把符合条件的单个服务组合在一起形成新的服务。在 UDDI 注册中心可能有很多 Web 服务提供相同的功能，但是却具有不同的非功能属性(如执行价格 price，执行时延 duration 等)。在 Web 服务组合时可以考虑作为 Web 服务质量的参考参数加以选择。

对于 Web 服务 ws，考虑 5 个非功能属性[7]：执行价格 price、执行时延 duration、可靠性 reliability、可用性 availability、信誉度 reputation，将它们作为服务质量的参考标准。

本文把 Web 服务的这些非功能属性看作是一个综合性能的函数 F(WS)＝F(price，duration，reliability，availability，reputation)。后面提到的图中，边的权值指的就是对应服务的性能函数值(假设综合性能值越小表示服务的 QoS 越好)。

定义 3　设有向图 $G=(V,\{E\})$，v、v' 是相邻节点，$E(v,v')$ 是连接 v 和 v' 的边，$W(v,v')$ 是该边的权重 weight，它是一个与服务性能属性(如执行 Web 服务的时间耗费、运行成本等)相关的函数，表示为 F(W)。这种边上带有权重的图称为带权 WSDG 图。

例如：假设 Web 服务 $w1$ 对应性能函数值是 2，那么就可以图形化表示为图 2。

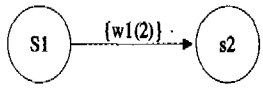

图 2　服务 w1 的带权图

形式化表示为 $\dfrac{\langle s1,s2\rangle}{w1(2)}$。

用户需求服务的性能函数值可以由用户根据情况指定，我们在发现组合服务过程中根据制定的条件选定组合所需要的 Web 服务。假设用户需求的服务需要 n 个服务 $w1$，$w2$，…，wn 组合而成，那么它的性能函数值就是 $F(W)=F(w1)+F(w2)+\cdots+F(wn)=\sum_{i=1}^{n}F(w_i)$。

根据依赖关系图及服务的形式化定义，我们可以从 OWL-S 描述的服务中抽取所需要的相关信息(如服务的输入输出和性能属性)，形成 UDDI 注册中心服务对应的 Web 服务依赖图。

例如：设有部分 Web 服务的形式化表示如表 2。那么根据带权依赖关系图的定义我们就可以得到服务的图形化表示(如图 3)。

表 2　Web 服务及其性能表

Web 服务	F(WS)	Web 服务	F(WS)
⟨S1,S2⟩/W1	2	⟨S4,S6⟩/W5	6
⟨S4,S2⟩/W1			
⟨S1,S2⟩/W2	5	⟨S9,S6⟩/W6	6
⟨S1,S3⟩/W2		⟨S9,S8⟩/W6	
⟨S1,S5⟩/W2		⟨S9,S11⟩/W6	
⟨S3,S6⟩/W3	2	⟨S8,S6⟩/W7	1
⟨S5,S6⟩/W4	4	⟨S8,S10⟩/W7	
		⟨S8,S11⟩/W7	
⟨S5,S7⟩/W4		⟨S8,S10⟩/W8	3
		⟨S7,S10⟩/W8	

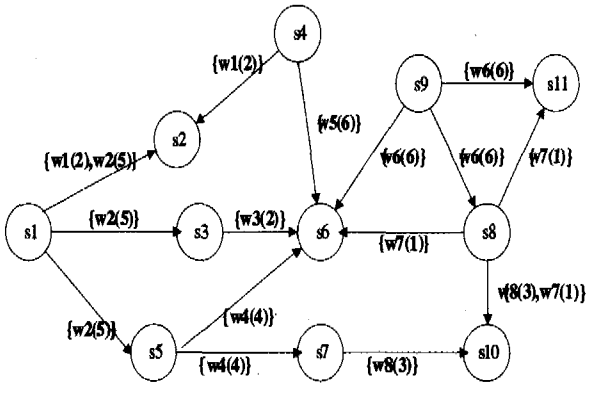

图 3　Web 服务带权依赖关系图

从图 3 中可以看出每个服务的输入输出。例如，服务 $w6$ 有一个输入 $s9$、3 个输出 $s6$、$s8$ 和 $s11$，服务的非功能函数值是 6。现在我们就可以利用已有的 Web 服务来组合得到满足用户需求的最佳服务。但是在这里还存在一个问题，就是是否在存储的服务关系图中存在一条从用户需求的输入到输出的路径。本文提供了一种判断方法，对输入进行检查，看是否可以产生需求的输出。

定义 4　设依赖图 $G=(V,E)$，如果有节点 $v1,v2\in V$ 并且从节点 $v1$ 到节点 $v2$ 至少存在一条路径，则称 $v1$ 可达 $v2$ 或 $v2$ 是从 $v1$ 可达的。从 $v1$ 可达的所有节点组成的集合称为 $v1$ 的可达集。

例如节点 $s4$ 可达 $s2$、$s6$，但是不可达 $s1$ 等节点。我们可以根据图的可达矩阵求解节点的可达性，也可以根据输入输出关系集合的传递闭包[13]来判断节点的可达性。例如 $R1=\{\langle s5,s7\rangle,\langle s5,s6\rangle,\langle s7,s10\rangle\}$，那么 $R1$ 的闭包 $R1^{+}=\{\langle s5,s7\rangle,\langle s5,s6\rangle,\langle s7,s10\rangle,\langle s5,s10\rangle\}$，也就可以得到 $s5$ 的可达节点集是$\{s6,s7,s10\}$。

这种节点的可达性分两种情况：如果输入可达输出的路径有多条，如何进行选择是我们下一步要解决的问题；如果只有一条就直接选择路径上的服务即可。但是假如输入不可达输出，那么也就说明在注册中的服务没有恰当的服务满足用户的组合要求，我们就需要重新查找或注册新的服务。本文仅关注可以在存储区的图中发现至少一条路径的情况。

例如：用户需求的输入集合是$\{s1\}$，输出集合是$\{s6\}$，输入输出依赖关系集合是 k_q

$k_q=\{s1\rightarrow s6\}$

并且要求得到的服务 QoS 较好。

从图 3 中我们可以很直观地看出有两条路径可以满足用户的需求，即

$$\frac{\langle s1,s3\rangle}{w2(5)}\rightarrow\frac{\langle s3,s6\rangle}{w3(2)} \tag{1}$$

$$\frac{\langle s1,s5\rangle}{w2(5)}\rightarrow\frac{\langle s5,s6\rangle}{w4(4)} \tag{2}$$

对应的依赖关系子图如图 4。那么就需要进行选择了。

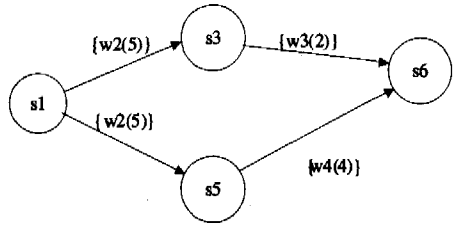

图 4　依赖关系子图

显然，QoS 较好的应该是性能函数总值较小的服务路径(1)满足要求，性能函数总值为 5+2<(2)的性能函数值 5+4。

这种在图中搜索最短路径的方法如果用手工完成，在服务数量较大的情况下就很难实现。因此，如何实现自动的搜索功能，就成为亟待解决的问题了。

2.2　算法步骤描述

本文采用图论搜索算法中的 Dijkstra 算法解决上面提到的问题，对得到的服务子图进行搜索，得到所有路径中的最短路径。

用户要将原始的 Web 服务组合成新的服务，利用 Web 服务依赖图(Web Service Dependency Graph，WSDG)发现组合服务，得到性能较好的需求服务，需要经过以下步骤：

1)根据依赖关系图的定义，从 OWL-S 描述的服务抽取相关信息，形成对应 Web 服务的 WSDG。

2)根据 UDDI 注册中心提供的的 Web 服务性能函数值把 WSDG 转化为带权的 WSDG。

3)根据用户的需求，在 WSDG 中标注与需求输入相匹配的节点集合。

4)从依赖图中标注的每个输入节点出发，根据可达性定义判断是否可达对应输出节点。如果不可达，则需要建立新的 Web 服务。

5)如果可达，并且只有一条路径，则直接选取路径上的服务；否则，如果从输入节点有多条路径可达输出节点，则使用 Dijkstra 算法搜索依赖关系右边的输出节点，得到输入节点到输出节点的最短路径。

6)重复 4)，直到 3)中标注的节点集合中的所有节点都遍历完。

7)对所得到的服务组合得到最终的需求服务。

设 service(e)是边 e 上所标识的服务集合；tag(si)表示服务 si 在 WSDG 中是否标记的布尔值，设初始值均为 false；min(集合)表示求得集合中的最小值。如果用户需求中包括

n 个输入输出依赖关系，即 $k_q=\{s1^1\rightarrow s1^2,\ s2^1\rightarrow s2^2,\ s3^1\rightarrow s3^2,\cdots,\ sn^1\rightarrow sn^2\}$，那么用户需求的输入集合就是 $s1=\{s1^1,\ s2^1,\ s3^1,\cdots,\ sn^1\}$；输出集合是 $s2=\{s1^2,\ s2^2,\ s3^2,\cdots,\ sn^2\}$。

执行过程描述：

```
//为每个服务添加性能函数值；
if(E≠∅)
{   for (each e in E)
  {   for( each service in service(e))
      {if(service-added==false)
//false 表示服务的性能函数值还没有添加；
       {
         character-value=F(service);
           //添加性能函数值；
         service-added=true;
      }
     }}}
//在图中标注所有输入节点；
for(each si1 in s1)
  {if(tag(si1)==false)
       {在 WSDG 中标注节点 si1；
  tag(si1)=true;
   }
  else continue;
  }
  //求出满足用户需求的所有最佳路径；
for( each si1→si2 in kq)
  { if(si1 不可达 si2)
       //没有满足这种要求的服务；
        building new service;
    else { if(only one path satisfied si1→si2 and min(service(e)))
//有一条路径满足要求并求出路径上所有边标识服务集合中服务性能最佳的服务
        select services on this path;
    else
     利用 dijkstra 算法求出所有 min(service(e))中的最短路径
       }
  }
```

3 实例

本文以学校教学管理系统为例来说明该优化组合策略。假设学校教学管理系统包含 6 部分，每一部分都作为一个 Web 服务提供，分别为学生管理服务 SI(对学生的有关信息提供服务，如可以提供学生信息查询等)；教师管理服务 TI(提供与教师相关的信息，如任课情况等)；排课服务 LI(提供课程的相关信息，如任课教师，所用书等)；学校信息服务 UI(提供学校的资源情况，如教师、班级等)、班级管理服务 CI(提供每个班级的概况，如任课教师、学生等)及教材订购服务 PI(提供教材信息，包括出版社信息等)等，对应模型如图 5 所示。

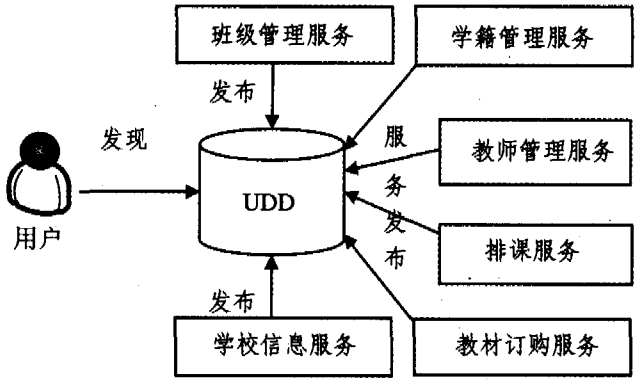

图 5 服务模型图

对应性能函数值分别为 F(SI)=0.3，F(UI)=0.2，F(CI)=0.5，F(LI)=0.2，F(TI)=0.1，F(PI)=0.4。那么就可以从服务对应的 OWL-S 描述中抽取出相应的输入输出等必要信息，依据前一部分的方法图形化这部分服务依赖关系(简化起见，本文根据服务的性能函数值直接给出了带权的 WSDG)，如图 6。

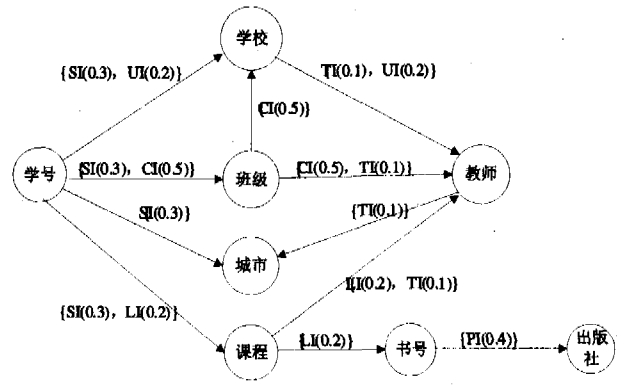

图 6 部分服务带权的 WSDG

图中节点表示 Web 服务的输入输出，边表示对应输入输出的 Web 服务。边的起点表示 Web 服务的输入，终点表示 Web 服务的输出。图 6 中我们把可以完成这种输入输出依赖关系的服务集合标示在边上。如果某个边标识的集合中含有多个元素，也就表示有多个 Web 服务满足此输入输出依赖关系。如果没有其他约束条件，那么当组合中用到这种依赖关系的 Web 服务时，就可以选集合中的任意一个 Web 服务进行组合。但是如果有 QoS 方面的要求，就需要从集合中挑选服务质量较好的服务进行组合。

例如用户的服务请求如下：

输入集合是{学号}，输出集合是={教师，出版社}，输入输出依赖关系集合是 k_q，其中：k_q={学号→教师，学号→出版社}要求得到服务属性最佳的满足要求的服务。

显然，用户输入节点集合为{学号}，我们可以在图中标注；然后从标注的每个节点出发判断可达性，从图 6 可以直接看出从“学号”到“教师”和“出版社”都存在路径，也就是说从输入到输出都是可达的。如果图形比较复杂不能直接判断是否可达时，可以利用求闭包进行判断。例如，从“学号”到“教师”，在形式化表示中存在序偶〈学号，学校〉、〈学校，教师〉，那么根据传递性可以得到〈学号，教师〉，也就表明从“学号”到“教师”是可达的。当然还存在其他的路径使得这两点间可达，这里我们不再一一表达出来。到此为止，已经知道这两个依赖关系都可以得到满足。可以得到带权依赖子图如图 7。

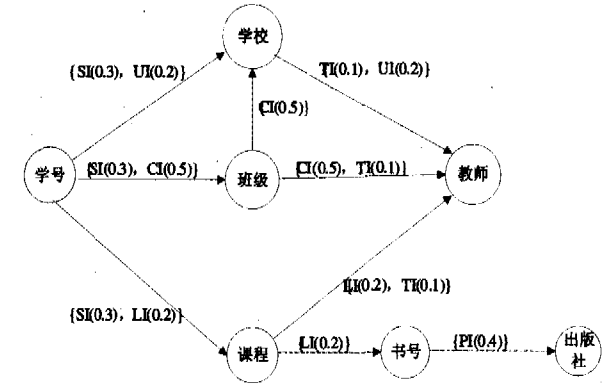

图 7 带权依赖子图

从图 7 中可以看出，从“学号”到“教师”存在多条路径，这时需要考虑用户对于 QoS 的要求，也就需要从这 4 条路径中寻求函数性能值最小的路径。本文利用 Dijkstra 算法求得最后满足要求的路径如下：

$$\frac{\langle 学号,学校\rangle}{UI(0.2)} \rightarrow \frac{\langle 学校,教师\rangle}{TI(0.1)} \quad (Ⅰ)$$

$$\frac{\langle 学号,课程\rangle}{LI(0.2)} \rightarrow \frac{\langle 课程,教师\rangle}{TI(0.1)} \quad (Ⅱ)$$

如果没有其他要求，在组合用户需求的服务时就可以从(I)、(II)两条路径中任选一条，或者使用并行运行的方式组合。而用户需求的另一个输入输出依赖关系：学号→出版社，只有一条路径满足，所以直接选择路径上的服务组合即可。综合所有输入输出依赖关系得到的最短路径也就得到了满足要求的最优依赖子图(如图 8)。

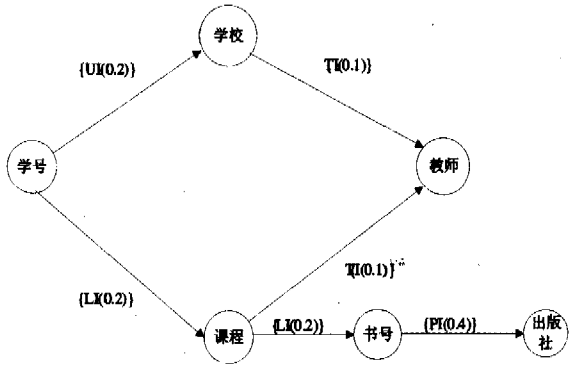

图 8　服务请求优化依赖关系子图

查找到组合需要的服务后，最后一步是对所得到的服务进行组合，最终得到满足用户要求的最佳服务。本文假设两个服务的顺序执行形式化表示为 wi · wj；选择分支形式化表示为 wi⊕wj ；并行分支表示为 wi ‖ wj，则最后得到的 Web 服务可以形式化表示为 WS=((UI · TI) ‖ (LI · TI))⊕(LI · LI · PI)或 WS=(UI · TI)⊕(LI · LI · PI)和 WS=(LI · TI)⊕(LI · LI · PI)。最后的组合结果如何选择，也是我们下一步要考虑的问题。

结论　Web 服务作为 Internet 中的新兴技术，使应用程序的集成比以前更快、更容易，而且更便宜。Web 服务是可以组合的，这是 Web 服务的一个重要特征，利用组合 Web 服务可以增加服务的附加值。本文在综合分析了已经出现的几种模型系统，指出了其中的不足(如表 1)；提出了自己的改进方法，即考虑服务性能的 Web 服务的发现及组合，使得组合得到的 Web 服务性能上更好，并举例验证了组合方法的有效性。与已有的其它组合方法相比，本文考虑了服务的质量属性，在服务发现过程中对 Web 服务进行约束，以组合产生具有较好 QoS 的服务。

本文我们只考虑了单输入的情况，下一步考虑对于有多输入的 Web 服务的情况。其次，需要研究这种组合方法对于死锁等的处理；研究如何实现自动从 OWL-S 中抽取输入输出及其关系。另外，如何使这种方法支持前置条件及后置条件，也是未来研究的一个方面。对于需要多个输入的 Web 服务，也是我们需要考虑的方面。

参 考 文 献

1 Nikola M, et al. Current Solutions for Web Service Composition. In: IEEE Computer Society, Dec. 2004

2 Oh S C, On B H, Larson E J, et al. BF*: Web Services Discovery and Composition as Graph Search Problem, e-Technology, e-Commerce and e-Service. In: 2005. EEE '05. Proceedings. The 2005 IEEE International Conference on, 29 March-1 April 2005. 784～786

3 Rao J. Semantic Web Service Composition via Logic-based Program Synthesis: [Dr Ing Thesis]. Norwegian University of Science and Technology, 2004. 121

4 Thatte S. XLANG : Web Services for Business Process Design. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/, 2001

5 Leymann F. Web Service Flow Language(WSFL1.0). http://www-4.ibm.com/software/solutions/Webservices/pdf/WSFL.pdf, 2001

6 Curbera F, Goland Y, et al. Business Process Execution Language for Web Service(BPEL4WS). http://www-106.ibm.com/developerworks/library/ws-bpel/, 2002

7 Zeng L, Benatallah B, et al. Quality Driven Web Services Composition. www2003, Budapest, Hungary, 2003

8 Fung C K, Hung P C K, Wang G, et al. A Study of Service Composition with QoS Management. In: Proceedings of the IEEE International Conference on Web Services (ICWS'05), 0-7695-2409-5/05

9 Ankolekar A, et al. DAML-S: Web Service Description for the Semantic Web. In: the Proceedings of the 1st Semantic Web Conference(ISWC), Sardinia, Italy, Jun. 2002

10 SOA:构建下一代 Web 服务的技术架构. http://www.softhouse.com.cn/special/sub_special.jsp? id=116

11 Andrews T, et al. Business Process Execution Language for Web Services(Version 1.1). http://www-106.ibm.com/developerworks/library/ws-bpel, May 2003

12 Booth D, et al. Web Services Architecture. http://www.w3.org/TR/ws-arch, W3C Working Group Note, February 2004

13 缪淮扣，李刚，朱关铭. 软件工程语言 Z. 上海：上海科技文献出版社，1999

14 Ganesarajah D. Web service workflow. http://www.doc.ic.ac.uk/~dgl97/project, 2001-06.

15 Hongbing W. Web services: problems and future directions. Web Semantics: Science, Services and Agentson the World Wide Web, 2004(1):309～320

16 Berardi D, Calvanese D, De Giacomo G, et al. Automatic Composition of e-Services. In: Proceedings of the First International-Conference on Service-Oriented Computing (ICSOC), 2003. 43～58

17 W3C. Web Service Choreography Interface (WSCI) 1.0. W3C Note 8 August 2002. http://www.w3.org/TR/2002/NOTE-wsci-20020808/

18 Joseph B, Karli W, Beginning. NET Web Services Using Visual Basic. NET

19 Ashemian V H S. A Graph-Based Approach to Web Services Composition. In: Proceedings of the 2005 Symposium on Applications and the Internet

20 Hamadi R, Benatallah B. A Petri Net-based Model for Web Service Composition. In: Proceedings of the Fourteenth Australasian Database Conference on Database Technologies 2003, 2003. 191～200

21 Martin D. OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owls/1.0/owl-s.html, November 2003

22 柴晓路，梁宇奇. Web Service 技术、架构和应用. 北京：电子工业出版社，2003

23 Menasce D A. QoS issues in Web service. Internet Computing, 2002(6)

24 W3C. Web Services Activity. www.w3.org/2002/ws