基于动态描述逻辑的服务组合及质量模型

万长林[1,2],韩 旭[1,2],牛温佳[1,2],王文杰[2],史忠植[1]
(1.中国科学院计算技术研究所智能信息处理重点实验室,北京 100190;
2.中国科学院研究生院,北京 100049)

摘 要: 本文提出一个基于动态描述逻辑的 Web 服务自动组合框架.在该框架中,Web 服务自动组合被划分为逻辑层和实现层两部分,于是服务的自动组合问题在逻辑上归结为一个动作规划问题,在实现上归结为一个根据动作选择具体服务的服务选择问题.基于该框架,本文给出了一个支持非线性 QoS 聚合和显式数据流声明的 QoS 模型.得益于合理的服务组合框架和 QoS 模型,本文提出的服务选择方法可以将复杂服务分解为较小的服务,然后分别用递归的剪枝算法求解.实验显示该方法是有效的,不但可适应候选 Web 服务数量较大的情况,且相对于现有方法能更好地处理服务组合中的复杂结构.

关键词: 服务质量; 动态描述逻辑; 服务选择; 服务组合; 分治算法
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2010) 08-1923-06

Dynamic Description Logic Based Web Service Composition and QoS Model

WAN Chang-lin[1,2], HAN Xu[1,2], NIU Wen-jia[1,2], WANG Wen-jie[2], SHI Zhong-zhi[1]
(1. *Key Laboratory of IIP, ICT, CAS, Beijing* 100190, *China*;
2. *Graduate University of CAS, Beijing* 100049, *China*)

Abstract: This paper proposes a semantic model for web service based on Dynamic Description Logic (DDL). In this paper, service composition is divided into two stages: logic plan stage and grounding stage. Therefore, a service composition problem is reduced to an action planning problem at the planning stage and a QoS-aware service selection problem at the grounding stage. This paper mainly focuses on the QoS model and service selection. A composite service is divided to several smaller services, which are then solved separately by a recursive branch-and-bound algorithm. Experiments show that the proposed approach is efficient and scalable for runtime service selection, and better handles complicated structure of service composition than existing approaches.

Key words: QoS; dynamic description logic; service selection; service composition; divide-and-conquer algorithm

1 引言

向服务计算的体系架构(SOA)是一种软件系统的合理设计方法,服务作为 SOA 架构中可被发现的接口在网络中发布,通过这些接口同时为最终用户和其它应用程序提供服务[1].服务组合是 SOA 的重要特征,当前对自动服务组合的研究大概可归于两大类:一类基于工作流组合,代表性工作是 BEA 系统公司、IBM 和微软等提出的 BPEL4WS[2];另一类基于智能规划,代表性工作是斯坦福大学、马里兰大学和卡内基-梅隆大学等机构提出的 OWL-S[3].动态描述逻辑 DDL[4,5]是描述逻辑的一种动态扩展.基于动态描述逻辑的自动服务组合是中国科学院计算技术研究所提出的自动组合方法,它利用动态描述逻辑的描述和推理功能,可以将 Web 服务作为动态描述逻辑中的动作进行建模和推理.此外,由于具有相似功能的语义 Web 服务的数量不断增加,对于区分这些语义 Web 服务来说,它们非功能特性(比如 QoS)变得越来越重要,因而需要在运行时从数量庞大的、不断变化的服务提供商中根据它们的服务质量 QoS[6]选择具体服务.本文将重点关注质量驱动 Web 服务组合中的运行时服务选择问题.

2 相关工作

Zheng 等[7]首先提出并初步解决了运行时 QoS 驱动的最佳服务选择问题,采用状态图对组合服务的控制流和数据流进行建模,并分两个阶段计算全局 QoS:首先使用整数规划计算组合服务每条执行路径的 QoS,再合并计算全局 QoS.Yu 和 Lin[8]将最优服务选择问题归结

收稿日期:2009-06-01;修回日期:2010-04-26
基金项目:国家自然科学基金(No.60775035,No.60903141,No.60933004,No.60970088);国家 973 重点基础研究发展计划(No.2007CB311004);国家 863 高技术研究发展计划(No.2007AA01Z132);国家科技支撑计划(No.2006BAC08B06);中国科学院研究生院院长基金(No.085101JM03)

为多选项背包问题(MCKP)和带约束的最短路径问题(CSPP),但是只限于处理 Sequence 控制流.Gao 等[9,10]将最优服务选择问题归结为带权的多层图中查找最长路径的问题并采用基于动态规划的算法对其进行求解,首次考虑了接口约束问题,但仅限于两个 Web 服务之间的接口.他们采用和 Zheng 等[7]相同的方式来处理复杂控制流.Canfora 等[11]提出一种基于遗传算法的解决方法,虽然执行时间通常比整数规划法要长,但是可以处理非线性的 QoS 聚合函数,并且支持多种控制流:Sequence、Switch、While、Flow 和 Pick.Cardellini 等[12]提出一种智能主体体系结构来优化所有收到的请求的端到端 QoS 聚合,采用线性规划技术来处理最优服务选择问题,并且对 Flow 结构做了比较详细的研究.Wan 等[13]通过对 BPEL4WS 和 OWL-S 的分析,提出一种复杂控制流和数据流的抽象模型和 QoS 模型,由此给出一种对复杂控制流和数据流支持更好的最优服务选择算法.马骞等[14]为 BPEL 组合服务引入一个运行时体系结构对象来解耦其与成员之间的绑定关系来实现 BPEL 组合服务的动态选择.叶世阳等[15]提出一个支持服务关联的 QoS 描述模型,基于该模型用整数规划和启发式搜索两种方法对最优服务选择问题进行求解.

3 基于动态描述逻辑的服务组合模型

3.1 原子 Web 服务建模

对于任一原子 Web 服务,本文通过为其关联一个动态描述逻辑[4]中的原子动作来进行语义建模.动态描述逻辑中的原子动作可定义如下:

$$a(x_1,\cdots,x_n)\equiv(P,E) \tag{1}$$

其中 a 为原子动作名,P 和 E 分别为动作的前提条件和执行结果;$x_1,\cdots,x_n$ 为在 P 和 E 中出现的所有个体名.

3.2 Web 服务组合模型

3.2.1 成员服务的控制流

复杂服务的控制流可由 DDL 中的动作构造符"?"、";"、"("以及"*"逐步构造得出.为了适应 QoS 驱动的服务选择,本文在文献[5]的基础上引入了五个复杂动作的宏定义:

$$While(s)::=((\varphi?\ s)^*;(\neg\varphi)?)\cup(((\neg\varphi)?;s)^*;\varphi')$$
$$Sequence(\{s,s'\})::=(s;s')\cup(s';s)$$
$$Switch(\{s,s'\})::=(\varphi?\ s)\cup((\neg\varphi)?;s')$$
$$Pick(\{s,s'\})::=s\cup s'$$
$$Flow(\{s,s'\})::=s\cap s'$$

其中 Flow 的构造符"∩"为本文提出的动作构造子,表示 s 和 s' 两个动作可以同时发生.基于 DDL 动作宏定义的组合服务可以被定义如下:

$$s\rightarrow t\mid While(s)\mid Sequence(S)\mid Switch(S)\mid Pick(S)\mid Flow(S) \tag{2}$$

其中 s 是服务的名称,t 表示基本服务任务,S 表示一组成员服务.图 1 刻画了一个涉及多个服务任务和多种控制流的"旅行计划者"场景.

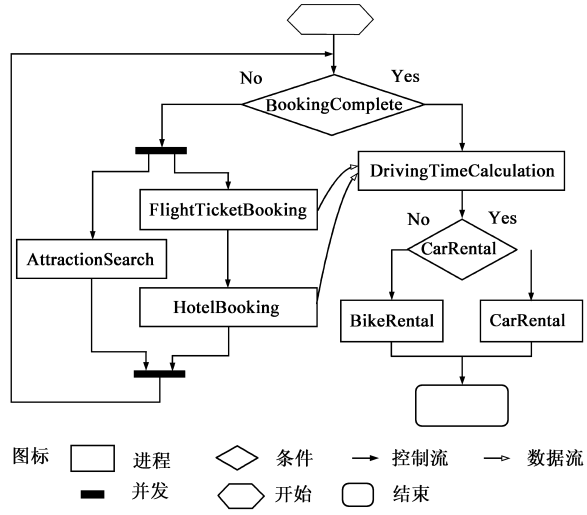

图1 组合服务"旅行计划者"中控制流和数据流的依赖关系

3.2.2 组合服务的数据流

OWL-S 使用绑定类来描述进程之间的数据流,BPEL4WS 通过对活动之间的交互状态建模来处理数据流,它们都不对复杂的数据流进行显式声明,本文提出数据流的声明应引入 *AND*,*OR* 两个结构进行扩展.

考虑一个包含 n 个基本服务任务的组合服务 s,可以定义服务 s 的数据流如下所示:

$$f\rightarrow<t_i,t_j>\mid AND(F)\mid OR(F) \tag{3}$$

其中 $1\leqslant i,j\leqslant n$,$f$ 是数据流的名称,t_i 和 t_j 是基本服务任务的名称,F 表示一组数据流.一个元组 $<t_i,t_j>$,或简记为 f_{ij},表示一个从起点服务任务 t_i 到目标服务任务 t_j 的简单数据流.

4 Web 服务的质量模型

4.1 QoS 的语义描述

根据动态描述逻辑的相关概念,可对 QoS 属性定义如下:

$$\begin{aligned}QoSAttribute\equiv&(=1\,hasName.AttributeName)\sqcap\\&(=1\,hasType.AttributeType)\sqcap\\&(=1\,hasUnit.AttributeUnit)\sqcap\\&(\leqslant1\,hasEnumConstraint.\\&\quad QoSEnumConstraint)\sqcap\\&(\leqslant1\,hasRangeConstraint.\\&\quad QoSRangeConstraint)\end{aligned}$$
$$QoSPrimitiveAttribute\equiv QoSAttribute\sqcap(\forall\ hasRule.PrimitiveRule)$$
$$\begin{aligned}QoSCompositeAttribute\equiv&\ QoSAttribute\sqcap\\&(=1\,hasRule.CompositeRule)\sqcap\\&\exists\ linkAttribute.QoSAttribute\end{aligned}$$

本文将 QoS 属性分为两大类:一类是基本 QoS 属

性,它不依赖于其它 QoS 属性;另一类是复杂 QoS 属性,它的度量依赖于其它 QoS 属性.这两类 QoS 属性都由属性名、属性类型、属性单位和对属性值的约束条件来定义.其中概念 QoSEnumConstraint 约束 QoS 属性的度量值必须在指定的几个值之内,而概念 QoSRangeConstraint 约束 QoS 属性的度量值必须在指定的值域之内.概念 PrimitiveRule 和 CompositeRule 分别用来定义基本 QoS 属性和组合 QoS 属性的度量值的计算规则/函数.

4.2 基本服务的质量属性

在本论文中,为便于叙述,本文考虑决定服务质量的四个典型属性[7]:

(1)执行价格 $Q_p(s)$:使用 Web 服务一次所需支付的资金数目.

(2)执行时间 $Q_d(s)$:发送请求到服务 s 和从服务收到响应之间的时间.

(3)可靠性 $Q_r(s)$:服务 s 未能完成其例程向请求者返回错误提示的几率.

(4)可用性 $Q_v(s)$:服务 s 不能被服务请求者访问的几率.

基于这四种质量属性,服务质量的比较就存在一种基本的偏序关系:对于具有相同功能的服务 s_1 和 s_2,如果:$Q_p(s_1)\leqslant Q_p(s_2)$, $Q_d(s_1)\leqslant Q_d(s_2)$, $Q_v(s_1)\geqslant Q_v(s_2)$, $Q_r(s_1)\geqslant Q_r(s_2)$, $O_1\supseteq O_2$,那么 s_1 的服务质量就优于 s_2 的服务质量,表示为 $s_1\succ s_2$.

4.3 组合服务的质量属性

本文假定在相同条件下不同的执行路径有相等的可能性被选择执行(等可能性假设),于是组合服务 s 的 QoS 聚合可由表 2 中列出的函数计算.

表 1　复杂控制流的 QoS 聚合函数

	$Sequence(s_1,\cdots,s_n)$	$Switch(s_1,\cdots,s_n)$	$While(s_1)$	$Flow(s_1,\cdots,s_n)$	$Pick(s_1,\cdots,s_n)$
Price $Q_p(s)$	$\sum_{i=1}^{n} Q_p(s_i)$	$\sum_{i=1}^{n} Q_p(s_i)/n$	$2\cdot Q_p(s_1)$	$\sum_{i=1}^{n} Q_p(s_i)$	$Q_p(s_k)$
Duration $Q_d(s)$	$\sum_{i=1}^{n} Q_d(s_i)$	$\sum_{i=1}^{n} Q_d(s_i)/n$	$2\cdot Q_d(s_1)$	$\max\{Q_d(s_i)\}, n\geqslant i\geqslant 1$	$Q_p(s_k)$
Reliability $Q_r(s)$	$\sum_{i=1}^{n} Q_r(s_i)$	$\sum_{i=1}^{n} Q_r(s_i)/n$	$[Q_r(s_1)]^2$	$\prod_{i=1}^{n} Q_r(s_i)$	$Q_p(s_k)$
Availability $Q_v(s)$	$\sum_{i=1}^{n} Q_v(s_i)$	$\sum_{i=1}^{n} Q_v(s_i)/n$	$[Q_v(s_1)]^2$	$\prod_{i=1}^{n} Q_v(s_i)$	$Q_p(s_k)$

5 服务选择模型

5.1 优化问题

对于任一基本服务任务 t,它的候选服务域 D 被定义为一组基本服务的集合,这些服务都能实现任务 t 所需功能.最佳服务选择问题可以被描述为:

给出一个组合服务 S,它包含 n 个基本服务任务及其候选服务域,寻找一个选择方案 $X(X:d_i\rightarrow t_i, d_i\in D_i, 1\leqslant i\leqslant n)$使得效用函数最大化并满足相关约束条件.

5.1.1 效用函数

由于 Web 服务组合的质量标准不一,很难确定一种指派(或一个服务)比另一个更好,除非在统一的标准下比较它们,本文基于四个基本质量属性定义了一个效用函数:

$$F(s)=\frac{w_1\cdot Q_r'(s)+w_2\cdot Q_v'(s)}{w_3\cdot Q_p'(s)+w_4\cdot Q_d'(s)} \tag{4}$$

其中 w_1,w_2,w_3,w_4 是预定义的权值.$Q_p'(s)$,$Q_d'(s)$,$Q_r'(s)$,$Q_v'(s)$是使 QoS 度量值规范化(文献[6]提供了一个详细的规范化过程)到值域[0,1]的函数.

5.1.2 服务接口约束

对于最佳服务选择问题,所有选定的服务的兼容性是一个必要的约束.由于接口约束可以通过数据流的声明来定义和检验.给定一个组合服务 S,如果它的成员服务 d_k 所有输入参数都能被初始输入 I_0 或它的之前执行的服务 $d_j(j<k)$的参数 O_j 匹配则认为服务 d_k 是满足接口约束的.

5.2 穷举搜索算法

最优服务选择问题可以通过穷举所有可能选择方案,计算其 QoS 值,然后选择满足所有约束条件的最佳方案来解决.如果有最佳方案存在,则此穷举搜索算法必定能找到该方案,但它运算量过大,在候选服务较多时就变得不可行.考虑 n 个服务任务 t_i 和候选服务域 D_i,可能的工作分配的总数最多是 $\prod_{i=1}^{n}|D_i|$,穷举搜索算法的时间复杂性是 $O\left(\prod_{i=1}^{n}|D_i|\right)$.

5.3 分治策略算法

本节提出一种最优服务选择算法 DCSS,其基本思想是:如果一个服务没有数据流输入或输出到其它服务,则该服务被认为是独立的.如果一个服务的所有成员服务都是相互独立的,则该服务是可分解的.对于不可分解服务,则使用递归的剪枝算法 RBAB 来搜索最优选择方案.在这个分枝扩展的过程中,有两种边界条

件:(1)通过检查当前的选择方案是否符合接口约束,不符合的分支可以被剪掉;(2)对于同一服务任务的两个具体服务 s_i 和 s_j,如果 s_i 优于 s_j 则分枝 s_j 可以被剪掉.DCSS 算法和 RBAB 算法分别描述如算法 1 和算法 2 所示,其中 ComputeQoS 和 IFSatisfy 分别为计算全局 QoS 和判断服务是否满足接口约束的算法(详见文[13]).

算法 1 基于分治策略的服务选择算法

服务选择算法 DCSS(s,D)

Input: service s, service domain D.
Output: best assignment Z (globally accessed by RBAB).
1: if s is elementary or $\exists f_{ij} \mid t_i \in s_i, t_j \in s_j, i \neq j$ then
2: for each service task $t_i \in s$ do
3: if t_i is interface free then
4: select the optimal service from domain D_i;
5: else
6: prune any services d_k if $d > d_k$;
7: end if
8: end for
9: $X \leftarrow$ *NULL* // current assignment X (globally accessed by RBAB)
10: $F_{\min} \leftarrow$ *MAX*// $F_{\min}$ denotes the minimal value of $F(s)$, *MAX* is a constant max number.
11: call RBAB(s,1);
12: else
13: for each component services $s_i \in s$ do
14: call DCSS(s_i, D);
15: end for
16: end if

算法 2 递归的剪枝算法

递归剪枝算法 RBAB(s,k)

Input: service s, current task number k.
Output: best assignment Z for service s.
1: for each service $d \in D_k$ for task t_k do
2: if IFSatisfy(t_k, $I_k - I_0$, f, X) = *False* then
3: try next service in D_k;
4: else
5: $X \leftarrow t_k : d$; // assign d to t_k in X.
6: if $k = n$ (n is the number of tasks in s) then
7: $F(s:X) \leftarrow$ ComputeQoS(s, X);
8: if $F_{\min} > F(s:X)$ then
9: $Z \leftarrow X$; $F_{\min} \leftarrow F(s:X)$;
10: end if
11: else
12: call RBAB($k+1$);
13: end if
14: end if
15: end for

6 实验评估

我们在一台配置 1.5GHz P4 CPU、512MB 内存、J2SDK 6 和 WINXP (SP 2)的工作站中对本文提出的最优服务选择算法进行实验评估.试验场景则使用图 1 所示的旅行计划者服务,其中任务 *DrivingTimeCalculation* 接收来自 *FlightTicketBooking* 和 *HotelBooking* 的数据流,且假设 *AirPortAddress* 和 *HotelAddress* 是 *DrivingTimeCalculation* 必需参数.每项任务 t_i 的服务 d_i 根据下面的规则随机生成:

(1)价格、持续时间、可用性和可靠性的值从(0.0,1.0)中随机选取;

(2)接口无关任务的输入/输出参数随机生成;

(3) *DrivingTimeCalculation*、*FlightTicketBooking* 和 *HotelBooking* 的参数的个数被设为其必需参数的两倍,每个参数都从必需参数集和随机生成的参数集中选择,必需参数集和随机参数集的选择概率分别为 80% 和 20%.

为了评估本文提出算法在服务任务的域大小(候选服务数量)改变时的效果,我们比较了该算法几种不同优化程度的变形:算法 A1 是纯穷举搜索算法;算法 A2 是对接口无关的服务任务事先选择最佳服务的穷举搜索算法;算法 A3 是一种没有事先处理接口无关服务任务的递归剪枝算法;算法 A4 为 5.3 节中所述的 DCSS 算法.每个算法在不同域大小时执行 100 次并计算其平均执行时间,实验结果如表 3 所示.

表 3 候选域大小改变时不同算法的执行时间比较

Domain Size	Running time (ms)			
	A1	A2	A3	A4 DCSS
10	10305	10	1375	8
20	*	100	8562	22
50	*	1438	*	155
100	*	11302	*	639
200	*	*	*	2783
500	*	*	*	16499

从表 3 中可以看到,对于包含复杂控制流和数据流的组合服务,纯穷举搜索运算 A1 开销太大.A2 通过预先处理接口无关的服务任务性能得到很大的改善,但仍然很难处理候选服务数量超过 100 的问题.递归剪枝算法 A3 采用了接口约束来修剪分支提高搜索性能,但仍然不足以处理较大规模的服务选择问题.DCSS 算法性能最佳,更重要的是,试验结果显示 DCSS 的运行时间几乎是随候选服务数量线性增加.

为了评估算法在服务本身大小(即涉及的基本服务任务数量)改变时的效果,本文按如下方法随机生成组合服务 S:随机生成组合服务 S 的控制流以及它的成员服务,而包含 6 个基本服务任务的旅行计划服务在这里作为组合服务的"基本"服务,即将随机生成的组合服务 S 中基本服务任务替换为包含 6 个基本服务任务的旅行计划服务,于是一个大小为 n 的组合服务在替换后大小变为 $n \times 6$.在实验中,服务 S 的大小分别被设置为 1×6、2×6、3×6、4×6、5×6 和 6×6,域的大小分别被设置为 10、20、40、50 和 80.对于不同的服务大小

和域的大小 DCSS 都要执行 100 次,并计算其平均执行时间.实验结果如图 2 所示.

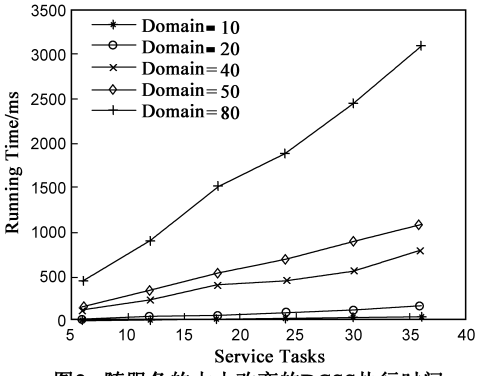

图2　随服务的大小改变的DCSS执行时间

从图 2 可以看到,得益于 DCSS 采用的分治策略,它的运行时间几乎是随服务大小线性增加.由于 DCSS 算法在实验中显示的对候选服务数量和服务本身大小具有良好的可伸缩性,使得它在实际应用中具有良好的适应性.此外,由于预选规则并未考虑控制流依赖性,这将导致所得到的最优服务选择方案的准确性有所下降.但是,经过我们额外的实验表明其准确性的损失是有限的.比较在 DCSS 中采用和不采用预选规则的两种情况,发现由此丧失的准确性始终小于 10%.

7　结论

本文基于动态描述逻辑提出了一种服务组合模型和 QoS 模型以及一种有效的基于分治策略的方法用于解决最优服务选择问题.得益于合理的服务建模,该方法不仅对于候选服务的数量而且对于组合服务的大小都是可扩展的.此外,与通常的处理复杂执行路径的两个阶段的方法[7~10,12]比较而言,本文采用的递归方法更适合于处理复杂的执行路径,它用一种集成化的方法处理多条执行路径,并且支持非线性 QoS 聚合函数.本文用计算实验对算法的有效性和效率进行了评估.实验结果显示本文提出的算法可以有效地解决问题且同时能适应候选服务数量较多和服务本身结构较复杂的情况.

参考文献:

[1] M P Papazoglou. Service-oriented computing: concepts, characteristics and directions[A]. In Proc of the 4th Int'l Conf. on Web Information Systems Engineering[C]. Rome: IEEE, 2003. 3 – 12.

[2] T Andrews, F Curbera, H Dholakia, et al. Business process execution language for Web services version 1.1[OL]. http://www. ibm. com/developer works/library/specification/ws-bpel/, 2002-07-30.

[3] D Martin, M Burstein, J Hobbs, et al. OWL-S: semantic markup for Web services[OL]. http://www. w3. org/Submission/2004/SUBM-OWL-S-20041122/, 2004-11-22.

[4] 史忠植,董明楷,蒋运承,张海俊.语义 Web 的逻辑基础[J].中国科学(E辑),2004,34(10):1123 – 1138.
Shi Zhongzhi, Dong Mingkai, Jiang Yuncheng, et al. The logical fundation for semantic web[J]. Science in China Series E: Information Sciences, 2004, 34(10): 1123 – 1138. (in Chinese)

[5] 史忠植,常亮.基于动态描述逻辑的语义 Web 服务推理[J].计算机学报,2008,31(9):1599 – 1611.
Shi Zhongzhi, Chang Liang. Reasoning about semantic web services with an approach based on dynamic description logics [J]. Chinese Journal of Computers, 2008, 31(9): 1599 – 1611. (in Chinese)

[6] Y Liu, A H Ngu, L Z Zeng. QoS computation and policing in dynamic web service selection[A]. In Proc of the 13th Int'l Conf on World Wide Web[C]. New York: ACM, 2004. 66 – 73.

[7] L Zeng, B Benatallah, M Dumas, et al. Quality driven web services composition[A]. In Proc of the 12th Int'l Conf. on World Wide Web[C]. Budapest: ACM, 2003. 411 – 421.

[8] T Yu and K J Lin. Service selection algorithms for web services with end-to-end qos constraints[A]. In Proc of the IEEE Int'l Conf. on E-Commerce Technology[C]. 2004: IEEE. 129 – 136.

[9] Y Gao, B Zhang, J Na, et al. Optimal selection of web services for composition based on interface-matching and weighted multistage graph[A]. In Proc of the 6th Int'l Conf. on Parallel and Distributed Computing Applications and Technologies[C]. Dalian: IEEE, 2005. 336 – 338.

[10] Y Gao, J Na, B Zhang, et al. Optimal web services selection using dynamic programming[A]. In Proc of the 11th IEEE Symposium on Computers and Communications[C]. Sardinia: IEEE, 2006. 365 – 370.

[11] G Canfora, M D Penta, R Esposito, et al. An approach for qos-aware service composition based on genetic algorithms[A]. In Proc of the Genetic and Evolutionary Computation Conference [C]. Washington DC: ACM, 2005. 1069 – 1075.

[12] V Cardellini, E Casalicchio, V Grassi, et al. Flow-based service selection for web service composition supporting multiple qos classes[A]. In Proc of IEEE Int'l Conf on Web Service [C]. Salt Lake City: ACM, 2007. 743 – 750.

[13] C L Wan, C Ullrich, L M Chen, et al. On solving qos-aware service selection problem with service composition[A]. In Proc of the 7th Int'l Conf on Grid and Cooperative Computing [C]. Shenzhen: IEEE, 2008. 467 – 474.

[14] 马骞,虞建杰,马晓星,等.一种基于运行时体系结构的 BPEL 支撑环境[J].电子学报,2006.34(12A):2360 – 2365.

Ma Qian, Yu Jianjie, Ma Xiaoxing, et al. A runtime software architecture-enhanced bpel supporting system[J]. Acta Electronica Sinica, 2006. 34(12A): 2360 – 2365. (in Chinese)

[15] 叶世阳,魏峻,李磊,等.支持服务关联的组合服务选择方法研究[J].计算机学报,2008,31(8):1383 – 1397.
Ye Shiyang, Wei Jun, Li Lei, et al. Service-correlation aware service selection for composite service[J]. Chinese Journal of Computers, 2008, 31(8): 1383 – 1397. (in Chinese)

作者简介:

万长林 男,1978 年生于江西安福.博士研究生,研究方向为人工智能、语义 Web 等.

韩 旭 女,1984 年生于河南安阳.博士研究生,研究方向为分布式人工智能、服务计算、故障诊断.

牛温佳 男,1982 年生于宁夏银川.博士研究生,研究方向为服务计算.

王文杰 男,1997 年於中科院计算所获博士学位,现为中科院研究生院副教授.研究方向为人工智能、语义 Web、Web 服务等.

史忠植(通信作者) 男,1941 年生于江苏宜兴.研究员、博士生导师,主要研究方向为智能科学、人工智能、多主体系统、数据挖掘、机器学习、知识工程等.1979 年、1998 年、2001 年均获中国科学院科技进步二等奖,1994 年获中国科学院科技进步特等奖,2002 年获国家科技进步二等奖.发表学术论文 400 余篇,出版专著 14 部.
E-mail: shizz@ics.ict.ac.cn