一种新的启发式Web服务组合算法

孙占志[1] 朱怡安[1,2] 车 鸣[1]
(西北工业大学计算机学院 西安 710129)[1] (西北工业大学软件与微电子学院 西安 710129)[2]

摘 要 作为推动SOA和Web服务向纵深化发展的重要支撑技术,Web服务组合一直在领域研究中占据着重要地位。提出了一种新的启发式Web服务组合算法——HASC算法,该算法分为遍历搜索和回溯组合两个过程。前者以Web服务输出集合基数为启发函数,确定到达每个目标本体所需调用的Web服务;在此基础上,后者采用输出集合与目标集合交集的基数为启发函数,逐步建立输出集合到输入集合的回溯路径,进而获取最优组合方案。最后,以公共测试集EEE05和ICEBE05为测试对象,对该算法的性能进行深入分析。实验结果表明,该算法在组合效率和寻求最短组合路径方面较同类算法有较大提升。

关键词 面向服务架构,Web服务组合,启发式搜索
中图法分类号 TP311 **文献标识码** A

New Heuristic Algorithm Based on Web Service Composition

SUN Zhan-zhi[1] ZHU Yi-an[1,2] CHE Ming[1]
(School of Computer Sciences and Engineering, Northwestern Polytechnical University, Xi'an 710129, China)[1]
(College of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710129, China)[2]

Abstract Web service composition is often considered to be one of the most important and vital building blocks for Service Oriented Architecture. Toward that, we presented a new heuristic algorithm named HASC. The algorithm obtains the solution through two steps which were traverse searching and regression. Both of the steps used heuristic method to select optimal Web services. In the process of traverse searching, the number of input parameters the Web service needed was considered as the heuristic function. In the process of regression, the heuristic function was the cardinality of the intersection generated by the output parameter set and the object ontology set. We evaluated the efficiency and effectiveness of HASC with two publicly available test sets—EEE05 and ICEBE05. Compared with other similar algorithms, HASC can provide higher efficiency and shorter solution path for the requests.

Keywords Service oriented architecture, Web service composition, Heuristic search

1 引言

作为SOA(Service Oriented Architecture)的主流实现形式,Web服务技术在解决异构分布式计算和代码与数据重用等问题方面表现突出,具有高度跨语言性、跨平台性和松耦合性等特点。近几年来,Web服务的种类和数量不断增长,提供相同或相似功能的Web服务越来越多,如何从海量Web服务中发现或组合一个合适的服务来完成用户请求,受到国内外越来越多研究机构和研究人员的关注[1]。部分学者致力于为Web服务添加语义信息,提供Web服务之间的可理解性,进而利用人工智能或程序验证的方法,结合用户请求直接生成新业务,最终实现Web服务动态组合。文献[2]以AI(Artificial Intelligence)规划为基础,提出WSPR算法,有效地提升了服务组合效率;文献[3]提出一种基于目标距离评估的算法,它根据已有Web服务的成功或失败经验动态地进行组合,能够很好地适应网络上Web服务不稳定的情况。相关研究工作还包括文献[4-8]等。这些算法的核心思想是将Web服务组合问题看作是一个状态空间的搜索问题,通过搜索Web服务库,寻找从初始状态到目标状态的最优或次优路径。从现有研究内容来看,主要工作集中于在尽量短的时间内找到合适的服务组合方案,而往往忽视了组合方案的质量,如使用该方案所需付出的代价等。

为了解决上述问题,本文提出一种新的启发式搜索算法——HASC(Heuristic Automatic Service Composition)算法。该算法首先根据服务请求遍历搜索Web服务库,计算出从初始集合到每个本体所需付出的代价;然后根据遍历搜索结果,以回溯方式找出初始集合与目标集合之间的最佳服务调用序列。在遍历及回溯过程当中,均采用启发式方法:遍历搜索过程以Web服务的输出集合基数作为启发函数,回溯过程以输出集合与目标集合交集的基数为启发函数。实验结果证明,该算法能够有效地完成状态空间搜索,并获取质量较优的服务组合方案。

到稿日期:2010-06-09 返修日期:2010-11-12
孙占志(1986—),男,硕士生,主要研究方向为高性能计算、Web服务计算,E-mail:george327@163.com;朱怡安(1961—),男,教授,博士生导师,主要研究方向为高性能计算、Web服务计算、网络软件应用技术;车 鸣(1981—),男,硕士生,主要研究方向为智能机器人应用、Web服务计算。

本文第2节介绍了Web服务组合问题，并对一些基本概念加以阐述；第3节详细介绍了启发式Web服务组合算法——HASC算法；第4节对算法的实现及性能进行了深入分析；最后对论文进行总结，并介绍下一步的工作。

2 问题描述与定义

Web服务组合就是通过组合简单Web服务来为用户提供增值服务的过程。该过程从用户需求出发，找到一个能达到用户目标的路径[4]。比如用户新到一个城市，需要找宾馆住宿：首先需要根据用户需求选择合适的宾馆，并根据日期预定房间，为了顺利到达，需要获取宾馆所在位置及行车路线。而用户也必须提供一些基本信息，比如他对宾馆价格、等级的要求，预定日期以及用户当前所处位置等。图1显示了Web服务需求与目标之间的关系。

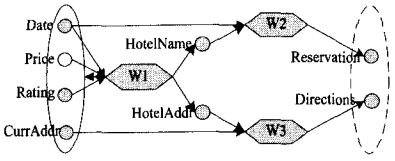

图1 Web服务组合示例

通常，Web服务组合问题可以表示为一个四元组：

$WSC=\langle S,R^{in},R^{out},W\rangle$

式中，S表示所有本体的集合，称为本体库。上例中的日期、宾馆地址、行车路线等都是本体，图1中用圆圈表示。R^{in}表示用户提供的初始本体集合，称为初始集合，$R^{in}\subseteq S$，如日期、宾馆价格、宾馆等级及当前地址，在图1中用实线椭圆框表示。R^{out}表示用户目标的本体集合，称为目标集合，$R^{out}\subseteq S$，如成功预定房间、行车路线，图1中用虚线椭圆框表示。W是由$S\rightarrow S$的映射构成的库，表示所有系统可用Web服务的集合，称为Web服务库。任意$w\in W$，都可表示为二元组$\langle w^{in},w^{out}\rangle$，其中$w^{in}\subseteq S,w^{out}\subseteq S$，前者表示该Web服务的输入参数，后者表示输出参数。图1中的六边形框表示在本例中用到的Web服务，$W=\{$"findHotel","reserveHotel","findDirection"$\}$。

基于这样一个四元组，Web服务组合方案可以看作是由一系列Web服务调用$\langle w_1,w_2,w_3,\cdots,w_n\rangle$构成，这里称之为服务链。每调用一个Web服务都会生成一个新的本体集合，所以Web服务组合的实现对应着一系列本体集合的获取过程$\langle S_0,S_1,S_2,\cdots,S_n\rangle$，且满足如下关系式：

$$S_0=R^{in},W_i^{in}\subseteq S_{i-1},S_{i+1}=S_i\cup w_i^{out},R^{out}\subseteq S_n \quad (1)$$

式中，$1\leqslant i\leqslant n-1$。

3 核心算法

在Web服务组合过程中，查找解决方案的最直接方式是遍历Web服务库。首先根据初始集合R^{in}建立已知集合Σ，逐个调用当前所有可行的Web服务w，并将其输出集合w^{out}加入到Σ中。重复上述过程，直到Σ包含了目标集合R^{out}中的所有元素。该算法在文献[9]中被称为SF(Stratified Flood)算法。以这种方式搜索Web服务组合方案效率很低，尤其是当服务数量较大时，算法的状态空间会随着Σ集合基数的增长呈指数增长。另外，通过该算法得到的解决方案有较多冗余项。为此，本文提出了HASC算法，其采用启发式方法，从而有效解决了上述问题。

HASC算法通过遍历搜索和回溯组合两个过程获取Web服务组合方案。在遍历搜索过程中，HASC算法假设在相同条件下，Web服务输出集合基数越大，最终服务组合方案的路径长度就越小，方案的服务质量也越好。在回溯组合过程中，HASC算法假设输出集合与目标集合交集的基数越大，最终服务组合方案的路径长度就越小。以上两点假设是HASC算法的理论基础。

步骤1 遍历搜索。计算从R^{in}到每个本体s所需付出的代价，计算过程可用如下的递归公式表示：

$$c(s)=\min_{w\in\Gamma(s)}[c(w)+\max_{s'\in w^{in}}c(s')] \quad (2)$$

式中，$c(s)$表示从初始集合R^{in}到本体s所需付出的代价，$\Gamma(s)=\{w\in W\mid s\in w^{out}\}$；$c(w)$表示调用Web服务$w$需要付出的代价，本文中取其值为1；$s'$表示已知本体。

算法1 遍历搜索过程

输入：初始集合R^{in}，服务库，代价最大值limit
输出：已知本体集合Σ

```
1  Σ=R^in;C=∅;cost=0;
2  for sin R^in do;
3     c(s)=0;w(s)=null;
4  while cost<limit and Σ⊉R^out do
5     Σ_t=∅;cost ++;
6     δ={w|w∈∅,w∉C,w^in⊆Σ};
7     for s in w^out(w∈δ)
8        if s∉Σ and s∉Σ_t then
9           Σ_t=Σ_t∪s;c(s)=cost;w(s)=w;
10       if s∉Σ and s∈Σ_t and |w^out|>|w(s)| then
11          w(s)=w;
12    Σ=Σ∪Σ_t;C=C∪δ
```

首先，将已知集合Σ初始化为初始集合R^{in}，并将cost值初始化为0(算法1的第1行)。对于初始集合中的所有本体，将$c(s)$的值初始化为0(算法1的2—3行)，其他本体的$c(s)$值则为∞。临时集合Σ_t用于保存新生成的本体。若已知集合中的元素能够满足某一Web服务的所有输入本体请求，且该Web服务尚未加入服务链C，则将其加入服务集合δ(算法1的第6行)。对该Web服务的所有输出本体做如下操作：若该本体在Σ中已存在，则不采取任何动作；若该本体不在Σ中但在Σ_t中，则表示该本体属于多个Web服务的输出集合，且得到该本体所需付出的代价相同，此时采用启发式方法选取Web服务。启发函数如下：

$$h(w)=|w^{out}| \quad (3)$$

式中，$h(w)$表示Web服务w的输出集合的基数。对原有生成该本体的Web服务$w(s)$与当前Web服务进行比较，取$h(w)$较大者赋予$w(s)$；若该本体不在临时集合Σ_t当中，则将其插入Σ_t，并设置其$c(s)$与$w(s)$。将Σ_t的所有元素并入Σ，并将δ的所有元素并入C。重复上述过程，直到Σ包含了目标集合R^{out}的所有元素，或者服务调用的代价超过了限定值limit。后者表示此算法没有找到合适的Web服务组合方案。

步骤2 回溯组合。根据步骤1的执行结果，以回溯方式找出R^{in}与R^{out}之间的最佳服务调用序列。使用临时目标集合Σ_t保存回溯过程中的目标本体，回溯开始时将其初始化为R^{out}(算法2的第1行)。定义Web服务集合δ，该集合中的元素与Σ_t的元素一一对应(算法2的3—4行)。然后采用启发式方法从δ中选取最优Web服务，启发函数如下：

$$h_{\Sigma_t}(w)=|w^{out}\cap\Sigma_t| \quad (4)$$

式中，$h_{\Sigma_t}(w)$表示 Web 服务输出集合与临时目标集合交集的基数。取其值最大的 Web 服务加入解决方案 soln，将该服务的输入集合 w_t^{in} 加入Σ_t，并将其输出集合 w_t^{out} 和初始集合 R^{in}中的所有元素移出Σ_t（算法 2 的 5—7 行）。重复上述过程，直到Σ_t 为空。最后，依次输出 soln 中的所有元素（算法 2 的 8—9 行），得到服务组合路径。

算法 2 回溯组合过程

```
输入：初始集合 R^in，目标集合 R^out，已知本体集合 Σ
输出：服务解决方案：w1→w2→…→wn
1 Σt=R^out,δ=∅,soln=∅;
2 while Σt≠∅ do
3     for s in Σt do
4         δ=δ∪w(s)
5     wt=arg max_{w∈δ} h_{Σt}(w)
6     soln=soln∪wt
7     Σt=(Σt∪wt^in)\(wt^out∪R^in)
8 for w in soln
9     print "→",w
```

4 实验分析

为了评估 HASC 算法的性能，本文将其和另外两种启发式 Web 服务组合算法——WSPR 算法[2]和 BF* 算法[9]进行了比较。实验环境部署在一台 CPU 为 Pentium4 3.00GHz，内存 1.0GB 的 PC 机上。IDE 环境使用 Eclipse3.2，程序运行平台为 JDK1.6，采用的测试集为 EEE05 和 ICEBE05[10]。实验分别选取 500，1000，1500，2000，2500，3000，3500，4000 个 Web 服务作为 Web 服务库，单个请求的执行时间和路径长度选取为本机上 50 个随机服务请求的算术平均值，并计算其标准偏差。

4.1 组合效率分析

算法从初始集合开始到找到正确组合方案所消耗的时间，反映了算法的组合效率。本文对 HASC，WSPR 以及 BF* 3 种算法的组合效率进行了比较，实验结果如图 2 所示。

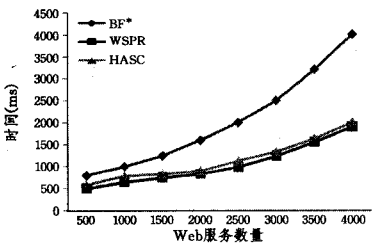

图 2 组合效率分析

由图 2 知，由于在回溯组合过程中采用了较优的启发函数，HASC 和 WSPR 较 BF* 在组合效率方面有较大提升，前两者组合效率接近。尤其是当 Web 服务库中的服务数量较大时，HASC 和 WSPR 的执行时间远短于 BF*。

4.2 组合路径长度分析

算法生成的组合方案中 Web 服务的个数反映了算法所提供的组合方案的质量。一般来讲，路径长度越小，调用这些服务所需付出的代价就越小，组合方案的质量也越好。

与 BF* 和 WSPR 相比，HASC 在遍历搜索阶段采用启发式方法，能够为每个目标本体选出该本体的最优 Web 服务，从而有效缩短了组合路径长度。由图 3 知，在服务数量相同的前提下，HASC 平均组合路径长度最短，WSPR 次之，BF* 最长。

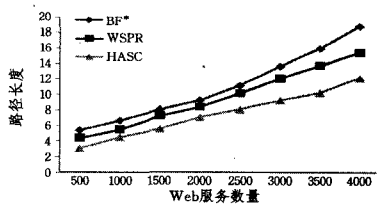

图 3 组合路径长度分析

由以上分析可见，HASC 算法在组合效率和寻求最短组合路径方面都能表现出较好的性能。

结束语 Web 服务自动组合是一种以用户为中心的业务模式，只需用户提供初始集合、目标集合和约束条件，就能计算出 Web 服务组合路径。基于启发式方法，本文提出了一种高效的服务组合算法。实验结果证明，该算法不仅可以自动得到满足请求的服务组合方案，而且能够有效地提高组合效率和组合方案质量。

为了进一步推进该领域的研究，本文后续工作主要包括：(1)深入探索并提升用户需求的语义描述和解析能力；(2)在服务组合算法中引入服务调用代价差异因子，进一步提高服务组合质量。

参 考 文 献

[1] 雷万保，朱怡安，迟文明. 基于可组合关联模型的 Web 服务排序算法[J]. 华中科技大学学报，2010，38(1)：105-108

[2] Oh S-C, Lee D, Kumara S R T. Effective Web Service Composition in Diverse and Large-scale Service Networks[J]. IEEE Transactions on Services Computing, 2008, 1(1): 15-32

[3] 温嘉佳，陈俊亮，彭泳. 基于目标距离评估的启发式 Web Services 组合算法[J]. 软件学报，2007，18(1)：85-93

[4] Chan K, Bishop J, Steyn J, et al. A Fault Taxonomy for Web Service Composition [J]. Lecture Notes in Computer Science, 2009: 363-375

[5] Lecue F, Silva E, Pires L. A framework for dynamic Web services composition [C] // CEUR Workshop Proceedings, 2nd ECOWS Workshop on Emerging Web Services Technology. Halle, Germany, 2007: 59-75

[6] Zeng Liangzhao, Ngu A H, Benatallah B, et al. Dynamic composition and optimization of Web services[J]. Distributed and Parallel Databases, 2008, 24(1-3): 45-72

[7] Sorin M, Paulo A, Maria E. Optimized dynamic semantic composition of services[C]// Proceedings of the ACM Symposium on Applied Computing. Hawaii, USA: ACM, 2008: 2286-2292

[8] Martin D, Burstein M, Hobbs J, et al. OWL-S: Semantic markup for web services [EB/OL]. http://www.daml.org/services/owl-s/1.1/overview/, 2004

[9] Oh S-C, Won B, Larson E J, et al. BF*: Web Services Discovery and Composition as Graph Search Problem[J]. IEEE Computer Society, 2005: 784-786

[10] Georgetown University. IEEE05 ICEBE05 [DB/OL]. http://ws-challenge.georgetown.edu/ws-challeng