

一种基于 A * 算法的 Web Services 动态组合策略

蔡 虹, 胡 云, 李存华

(淮海工学院 计算机工程学院, 江苏 连云港 222005)

摘 要: 随着 Web Services 应用的发展, 服务的动态组合技术为解决信息共享和应用协作的问题带来了新的方案, 然而生成适应符合客户 QoS 需求的组合方案是耗时的优化问题. 文中通过构建 QoS 驱动的服务组合依赖图模型, 提出了基于 A * 算法的启发式的最佳路径组合方法. 该算法适用于存在多输入的 Web Services 动态组合情况并能够生成具有最小代价的简单调用序列. 实验表明, 该算法适用于多输入的 Web Services 动态组合情况, 并且具有较好的运行效率.

关键词: Web 服务组合; A * 算法; 依赖图

中图分类号: TP31; TP39

文献标识码: A

文章编号: 1000-7180(2011)09-0197-05

Dynamic Web Services Composition Strategy Based on A—Star Algorithm

CAI Hong, HU Yun, LI Cun-hua

(School of Computer Engineering, Huaihai Institute of Technology, Lianyungang 222005, China)

Abstract: With the development of web services, the dynamic composition technique of web service provides a new solution to solve the problem of information sharing and the application cooperating. However, generating the composition plan to meet QoS requirement of client is a time-consuming optimization problem. To solve the problem in a timely manner, this paper presents a heuristic method based on A—star algorithm by modeling the QoS-driven dependency graph. The algorithm can be applied to the situation with multi-inputs to generate a simpler calling sequence with lowest cost. The experiment shows that our method has good efficiency.

Key words: web services composition; A—star algorithm; dependency graph

1 引言

Web Services 技术为异构、自治和松散耦合的分布式系统应用提供了一个集成和交互机制. 但要想仅仅通过单一的、简单的 Web Services 交互实现真正跨企业边界的应用集成是显然不够的^[1]. 因此, 有必要对已有的、自治的、功能简单的 Web Services 根据应用需要进行动态、自动的组合, 从而提供新的功能更强的 Web Services.

目前对 Web Services 组合已有许多的研究, 有基于工作流的, 有基于语义的^[2], 但无论是哪种方法都要将 Web Services 自动或手工合成满足需要的

组合方案, 然后把生成的方案转换成可执行的代码. 在 Web Services 组合过程中, 利用已有的一些人工智能或者程序验证的方法, 从用户的需求直接生成新业务. 这种从需求直接自动生成 Web Services 组合业务的方式已成为目前研究的热点, 在这个过程中, 组合算法的选择至关重要, 关系到组合的效率.

文中在 Web Services 组合建模及本体构造的基础上, 利用执行代价评估方案将组合问题抽象为一个表示输入输出依赖关系的带权依赖图. 在此基础上提出一个基于启发式 A * 算法的 Web Services 动态组合策略, 这种算法的核心思想是将 Web Services 的组合问题看作是一个状态空间的搜索问

收稿日期: 2011-01-10; 修回日期: 2011-02-08

基金项目: 江苏省自然科学基金项目(BK2008190)

题. 该方法能够有效地利用已有的 Web Services 组合经验, 从而在大多数情况下都能够高效地完成状态空间的搜索, 实现对 Web Services 的高效组合, 并能有效解决服务组合过程中的多服务问题.

2 Web Services 组合模型表示

由于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, 因此本文采用带权依赖图的表示方法来表示服务之间的关系使得用户可以在考虑 QoS 的情况下进行 Web 服务的选择和组合.

定义 1 如果 v_x 和 v_y 分别是 Web Services w_i 的输入和输出信息, 那么就称 v_x 与 v_y 具有服务依赖关系, 表示为 $v_x \rightarrow v_y$. 所有具有这种服务依赖关系的消息所组成的关系集合称为服务依赖关系集.

定义 2 一个依赖图 $G = (V, E)$ 是一个有向图, 包括存储区中所有 Web Services 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服务依赖关系. 若节点 $v_x, v_y \in V$, 那么在图 G 中从节点 v_x 到 v_y 有边当且仅当服务依赖集中存在有服务依赖关系 $v_x \rightarrow v_y$. E 中的每条边都对应一个集合, 该集合包含所有那些在服务依赖关系集中包含有这种服务依赖关系 Web Services.

在服务组合过程中, 因为一个服务可能需要多个服务的输出作为输入才能得到期望的结果, 所以涉及多个顶点到一个顶点的一组路径, 这个问题称为多服务问题. 为了在依赖图中表示多服务问题, 我们给出如下定义.

定义 3 若仅当两个或两个以上的 Web Services 输出信息同时具备才能满足后继 Web Services 输入信息需要, 则称这种情况为“Web Services 的与”, 用“ $\wedge$ ”表示. 如图 1 所示.

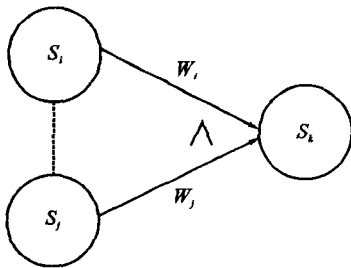


图 1 Web Services 的与

定义 4 若一个或多个 Web Services 输出信息中的任意一个具备, 就能满足后继 Web Services 输入信息的需要, 称这种情况为“Web Services 的或”, 用“ $\vee$ ”表示. 如图 2 所示.

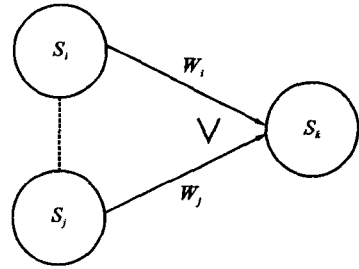


图 2 Web Services 的或

在 Web 环境下, 可能有多个 Web Services 都提供相同的功能, 但其执行代价、可靠性和可用性不一定相同. 在 QoS 驱动的 Web Services 组合模型中, 可用从多个方面进行衡量. 文中主要从以下 5 个方面进行代价评估, 并提出相应评估方案.

(1) 价格 Q_{price} : 是指服务请求者调用一次服务所要支付的费用, 一般由服务供应商确定.

(2) 持续期 $Q_{duration}$: 是指客户端从提交服务请求到获得服务响应预期所花费的时间, 由服务执行时间和传输时间两部分构成.

(3) 可靠性 $Q_{reliability}$: 表示服务请求在最大期望时限内被正确响应的概率, 可分为网络可靠性和执行可靠性. 服务的可靠性的值用以下方法进行计算: $Q_{reliability} = N/K$, 其中 N 是到目前为止 Web Services 在最大期望时限内成功完成的次数, K 是到目前为止 Web Services 被请求调用的总次数.

(4) 声誉 $Q_{reputation}$: 表示用户对特定 Web Service 满意程度, 服务的声誉可综合用户的评价, 以一定计算方式获得.

(5) 可用性 $Q_{availability}$: 表示 Web Services 可以被访问到的概率. 这个属性值在动态环境中可能会发生变化, 因此需要定期进行更新. 可用公式 $Q_{availability} = \text{服务可用时间} / \text{服务存在时间}$ 获得.

目前在 Web Services 组合中 QoS 的计算目前主要有两种形式: (1) QoS 各属性简单加权; (2) 对各方案的 QoS 应用归一化模型处理. 文中的重点是利用相同 QoS 计算标准来评估不同的 Web Services 动态组合策略, 因此选择了相对容易实现的第一种方法, Web Services 的 QoS 可以表示为:

$$Q(WS) = W_p * Q_{price}(WS) + W_d * Q_{duration}(WS) + W_{rel} * Q_{reliability}(WS) + W_{rep} * Q_{reputation}(WS) + W_a * Q_{availability}(WS)$$

其中, W_p 、 W_d 、 W_{rel} 、 W_{rep} 、 W_a 、分别代表五个质量标准在评估体系中所占的权重.

根据以上的 QoS 评估方案, 将一个 Web Services 性能属性 $Q(WS)$ 与依赖图中相关的边进行绑

定,用 $Q(WS)$ 的值表示该边的权值,就生成带权的 Web Services 组合依赖图.如图 3 所示.

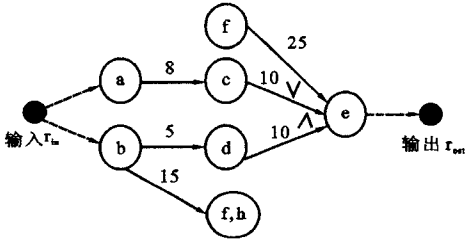


图 3 带权 Web Services 组合依赖图

3 基于 A * 算法的最佳组合路径选择

根据以上建立的模型,Web Services 组合实际上就是研究如何找到一组可以调用的 Web Services 操作,以及如何以串行、并行等方式组合调用这些操作来完成期望的功能.而这种调用的序列在依赖图中反映为有向图中的一条路径,由于这个调用序列可能并不唯一,而用户希望在一次组合调用中花费的代价最小,所以最终问题转化为求图中源点到目标点的一条最短路径问题.

A * 算法作为启发式算法中很重要的一种,广泛应用在最优路径求解和一些策略设计的问题中,被认为是求最短路径最有效的方法之一^[3].但 A * 算法并不能直接应用在 Web Services 组合过程中,因为 A * 算法主要用于寻找点与点之间的最短路径.而在 Web Services 组合过程中一个服务可能需要多个服务的输出作为输入才能得到期望的结果,所以涉及多个顶点到一个顶点的一组路径,也就是多服务的问题.另外组合中一个顶点可能由多个元素组成,如图 3 所示. $\{a, b\}$ 可以作为一个顶点考虑用来产生 e,这同时需要考虑两个规则 $a \rightarrow c$ 和 $b \rightarrow d$,对于这些情况 A * 算法不能直接处理.若一个服务需要多个服务的输出作为输入,则多个服务被称为前趋服务,这时候最佳选择算法在使用 A * 算法计算出顶点对之间的最短路径后,把需要多个服务提供输出的服务和它的前趋服务一起记录下来,把前趋中每个服务的最短路径之和加上服务和前趋服务之间的开销来计算该服务的最短路径.

多输入问题则需要加入新的产生规则来解决.

从 s 到 v 的最短路径 $\delta(s, v)$ 表示 Web Services 组合中的最佳路径:

$$\delta(s, v) = \begin{cases} \delta(s, v) = \min(\text{path}: s \rightarrow v) & s \text{ 到 } v \text{ 有至少一条路径} \\ \infty & \text{否则} \end{cases}$$

设 $P(s, v)$ 为顶点 s 到顶点 v 的开销,对于图 3 可得:

$$\delta(\{a, b\}, e) = \min\{\delta(\{a, b\}, f) + p(f, e), \delta(\{a, b\}, c) + \delta(\{a, b\}, d) + p(\{c, d\}, e)\}$$

其中, $\delta(\{a, b\}, f)$ 表示由 a, b 共同产生 f 的最佳路径, $p(\{c, d\}, e)$ 表示为 $\{c, d\}$ 到顶点 e 的开销.一般情况下,若 v 有前驱顶点 m, n, u , m, n 共同产生 v , u 可以独立产生 v , $P(s, v)$ 为从顶点 s 到顶点 v 的开销,则从起点 s 到 v 的最短路径可以表示为:

$$\delta(s, v) = \min\{\delta(s, u) + p(u, v), \delta(s, m) + \delta(s, n) + p(\{m, n\}, v)\}$$

并可以得到不等式:

$$\delta(s, v) \leq \delta(s, u) + p(u, v)$$

以上述不等式为基础,利用 A * 算法计算出所有可能的输入输出对之间的最短组路径.这些组路径被保存下来供查询.对于多服务问题,如果不同产生规则中的多个顶点协同产生顶点,则该顶点的最短组路径为所有的前趋服务的代价之和加上它和前趋服务之间的开销,然后更新所有从该顶点可以产生的顶点的最短组路径.

A * 算法在人工智能中是一种典型的启发式搜索算法.启发式搜索就是在状态空间中的搜索对每一个搜索的位置进行评估,得到最好的位置,再从这个位置进行搜索直到目标.这样可以省略大量无畏的搜索路径,提高效率,A * 算法的核心部分在于估价函数的设计,公式表示为: $f(n) = g(n) + h(n)$.

A * 算法特别适合于静态网络搜索,算法效率高,通常选取的估计函数为当前顶点 n 到目标顶点的直线距离.但由于动态组合所生成的网络节点的不确定性,所以需要对目标节点 n 的估价函数做重新选取,同时在原有算法的基础上进行改进,其核心思想是:将当前临时标记节点到源点的最小费用、当前临时标记节点到目标节点的最小费用、当前临时标记节点前一节点到目标节点的最小费用三者之和作为此临时标记节点的一个属性值,这个属性值将作为从临时标记节点集合中选取永久标记节点的依据.修订后的目标节点 n 的估计函数为:

$$f^*(n) = g(n) + h^*(n) + h^*(n-1)$$

其中, $h^*(n)$ 为当前节点到目标节点的最短组合时间估计, $h^*(n-1)$ 是从临时标记节点前一点到目标节点的最短组合时间估计.

在算法实现中,首先需要创建 OPEN 表和 CLOSE 表,OPEN 表保存所有已生成而未考察的节点,CLOSED 表中记录已访问过的节点.算法运

行过程中会根据估价函数的大小对 OPEN 表进行重新排序,这样做的目的是为了使每一步只要考虑 open 表中状态最好的顶点. 具体的算法描述如下:

```

1: Initialize OPEN set;
2: Initialize CLOSE set;
3: Create goal node named goal_node;
4: Create start node named start_node;
5: Add start_node to the OPEN set;
6: while (OPEN set is not empty)
7: { Get node  $n$  from the OPEN set with the lowest  $f^*(n)$ ;
8: Remove node  $n$  from the OPEN list;
9: Add  $n$  to the CLOSE set;
10: if ( $n$  is the same as goal_node)
11: return Solution( $n$ );
12: else
13: {
14: Generate each successor node  $n_s$  of  $n$ ;
15: for (each successor node  $n_s$  of  $n$ )
16: { Set the parent of  $n_s$  to  $n$ ;
17:  $h^*(n_s) \leftarrow$  the estimate value from  $n_s$  to goal_node;
18:  $h^*(n_s - 1) \leftarrow$  the estimate value from preceding
node of  $n_s$  to goal_node;
19:  $g(n_s) \leftarrow g(n) +$  the cost to get to  $n_s$  from  $n$ ;
20:  $f^*(n_s) \leftarrow g(n_s) + h^*(n_s) + h^*(n_s - 1)$ 
21: if ( $n_s$  in OPEN set && the existing one is better)
22: discard  $n_s$  and continue;
23: if ( $n_s$  in CLOSE set && the existing one is better)
24: discard  $n_s$  and continue;
25: Remove occurrences of  $n_s$  from OPEN and
CLOSE set;
26: Add  $n_s$  to the OPEN set;
27: }
28: }
29: }
```

4 实验分析

为了验证 A* 算法在动态 Web Services 组合环境中的可行性和效率,实验使用随机生成的 Web Services 模型库从固定用户输入和固定规则数目两个方面进行模拟效率测试,从时间和性能两个方面进行分析. 实验中设定的 QoS 属性的权重为 $W_p = 0.1$ 、 $W_d = 0.1$ 、 $W_{rel} = 0.2$ 、 $W_{rep} = 0.1$ 、 $W_a = 0.5$,在实际应用中用户也可根据不同的需要调整 QoS 各属性的权重.

实验首先在规则数目固定的情况下改变用户输入的参数进行模拟实验比较本算法与文献[8]中提

到的最优算法,实验采用随机生成的 Web Services 模型库,规则数为 100. 在时间方面,考虑用户输出对运行时间的影响,如图 4 所示. 在文献[8]中用户的输入越多,穷举花费的时间就会呈指数的增长. 而在本算法中只是呈一个较为平缓的增长. 显然本算法在输入较多的情况下时间上是明显优于文献[8]的. 另外,当用户输入个数固定时,组合选择策略中 A* 算法随着规则数目的增长而基本呈指数增长,而规则数目固定的情况下算法的执行时间随着用户输入的增多近似呈线性.

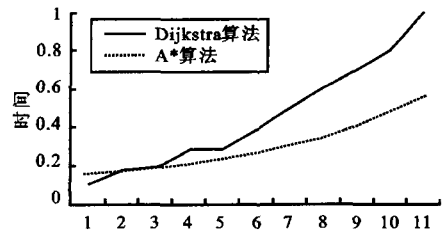


图4 算法效率图

性能方面,由于引入文献[9]中的多标准综合评价方法,最优性能的理论证明比较复杂,因此通过实验的方法进行评价. 对不同的输入输出进行了 300 次的实验,本算法得到的结果为最优的方案次数为 89 次,得到的结果排在前三的次数为 253 次. 并且排在第二和第三的结果值非常接近最优值,而 Dijkstra 算法分别为 62 次和 204 次(见图 5). 此外,通过观察发现,未能得到最优解的主要原因由于在路径选择初期,对中间路径的计算时,由于最高代价和最小代价路径是全局值,可能直接舍弃最优方案的某条中间路径. 总体结果表明该算法能在绝大多数情况下得到用户较为满意的 QoS 驱动的组合方案. 实验表明该算法适合 Web Services 动态组合过程,也具有较好的效率.

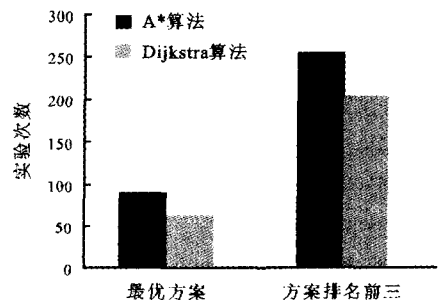


图5 性能比较图

5 结束语

文中根据 Web Services 组合的特点,应用图论中相关知识建立了 QoS 驱动的服务组合的模型,在

此基础上提出了基于A*算法的动态组合方法,同时解决了A*算法在应用于Web Services组合时出现的多服务问题.通过实验分析确认该算法适用于多输入的Web Service动态合成情况,并且具有较好的算法处理时间.下一步的研究是进一步改善算法估价函数,将该算法生成的组合方案转换为可执行的代码并提高转换和执行效率.

参考文献:

- [1] Yuri Babich. Composition of web services and Qos aspects[Z]. seminar;Data Communication and Distributed Systems 2003/2004.
- [2] Sheshagiri M, DesJardins M, Finin T. A planner for composing services described in DAML-S[C]// Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering. Baltimore:[s. n.], 2003: 267-279.
- [3] Lloyd Smith B, Kist A. A shortest paths in stochastic networks[J]. IEEE, 2004(4):492-496.
- [4] Seyyed Valhid Hasemian, Farhad Mavaddat. A Graph-based approach to web services composition[C]//Proc of the Symp on Applications and the Internet. 2005:183-189.
- [5] Liang Wen Yau, Huang ChunChe, Chuang HorngFu.

The design with object (DWO) Approach to Web services composition[J]. Computer Standards&Interfaces 2007,29(1):54-68.

- [6] 滕星全,李增智. 基于OWL-S的Web服务组合的验证研究[J]. 微电子学与计算机, 2007,24(12):62-69.
- [7] Juntao Cui, Jiamao Liu, Yujin Wu. An ontology modeling method in semantic composition of web services[C]// Proceedings of 2004 International Conference on e-Commerce Technology for Dynamic e_business. Washington, DC, USA; IEEE Computer Society, 2004: 102-109.
- [8] Liu Jiamao, fan Chenhui, Gu Ning. Web services automatic composition with minimal execution price[C]// Proceedings of the IEEE International Conference on Web Services (ICWS'05). Washington, DC, USA; IEEE Computer Society, 2005:302-309.
- [9] Cui Juntao, Zhu Yiwen, Gu Ning. An algorithm with best selection policy in semantic composition of web service[J]. Lecture Notes in Computer Science, 2005: 816-821.

作者简介:

蔡虹女,(1977-),硕士.研究方向为计算机软件及应用、数据库技术.

(上接第196页)

关系,所以接下来亟待解决的问题是如何确定识别完整度峰值时对应的各参数相应取值.

参考文献:

- [1] Weber J L, Myers E W. Human whole-genome shotgun sequencing[M]. New York: Cold spring Harbor Laboratory Press, 1997.
- [2] Beleza S, Alves C, Gonzales-Neira A. Extending STR markers in Y chromosome haplotypes[J]. International Journal of legal Medicine, 2003, 117(1):27-33.
- [3] Sonnhanner E L, Durbin R. A dot-matrix programme with dynamic threshold control suited for genomic DNA and protein sequence analysis[J]. Gene, 1995, 167(1):1-10.
- [4] 李冬冬,王正志,倪青山. 一种有效的重复序列识别算法[J]. 生物信息学, 2005, 63(4):163-166.
- [5] Gu Wanjun, Todd A Castoe, Dale J Hedges, et al. Identification of repeat structure in large genomes using repeat probability clouds[J]. Analytical Biochemistry, 2008, 380(1):77-83.

- [6] Buhler J. Search algorithms for biosequences using random projection [D]. Washington, DC, USA; Ph D thesis University of Washington, 2001.

- [7] Morrison D R Patricia. Practical algorithm to retrieve information coded in alphanumeric[J]. Journal of the Association for computing Machinery, 1968.
- [8] Ukkonen E. A space-economical suffix tree construction algorithm[J]. Journal of ACM, 1976, 23(2):262-272.
- [9] Weiner P. Linear Pattern Matching Algorithms[C]// Proc 14 th IEEE Annual Symp on Switching and Automata Theory. Washington, DC, USA; IEEE Computer Society, 1973:1-11.
- [10] Han J, Wang W, Prins J. Efficient mining of frequent subgraphs in the presence of isomorphism[C]// Proceedings of the third IEEE International Conference on Data Mining. Washington, DC, USA; IEEE Society, 2003.

作者简介:

周溜溜男,(1987-),硕士.研究方向为数据挖掘、机器学习.