一种业务流程QoS有保障的动态服务组合方法

李盛恩[1] 洪晓光[2]
(山东建筑大学计算机科学与技术学院 济南 250101)[1] (山东大学计算机科学与技术学院 济南 250100)[2]

摘 要 为了使动态组合后形成的基于Web服务的业务流程不仅能够完成业务流程分配的任务,即满足局部约束,还能够与业务流程中完成其他任务的Web服务协作,使整个业务流程QoS有保障,达到全局最优,本文提出了一种基于遗传算法的方法来对问题进行求解。实验结果证明了文中方法的有效性。

关键词 Web服务组合,QoS,遗传算法

A Business Process QoS-Guaranteed Dynamic Service Composition Method

LI Sheng-En[1] HONG Xiao-Guang[2]
(School of Computer Science and Technology, Shandong Jianzhu University, Jinan 250101)[1]
(School of Computer Science and Technology, Shandong University, Jinan 250100)[2]

Abstract In order to make choice among the candidate Web services so that the selected ones can not only finish the assigned task, conform to the local restriction, but also cooperate with other Web services and optimize the QoS of the process, this paper proposes an approach based on the generic algorithm. The experimental results show our method is an effective one.

Keywords Web service composition, QoS, Generic Algorithm

1 引言

随着Web服务技术的普及与发展,一方面,越来越多的用户开始借助于通过组合Web服务来完成各类业务流程,这使得业务流程模式开始面向更广泛的普通用户;另一方面,大量具有相同或相似功能的Web服务的出现使得用户不再指定具体的Web服务作为流程中任务的执行者,而是由运行系统根据用户提出的需求和限制条件来自动选择Web服务。

在Web服务的选择中,除了功能性的属性之外,用户还可以通过对一些非功能性的属性设定约束来完成,即通过约束QoS(Quality of Service)属性来进行选择。QoS可以包含的属性有:价格、响应时间、可用性和可靠性等[2,8,9]。而Web服务的提供者也往往会对服务的QoS属性值给出一个范围,从而为潜在用户提供参考。

在QoS敏感的业务流程中,除了要求流程完成预先定义的任务之外,整个流程的QoS也是用户所关心的特征。因此,如何在备选的Web服务中进行有效的选择,使得被选中的Web服务不仅能够完成流程分配的任务、满足局部约束,还能够与流程中完成其他任务的Web服务协作,使流程的QoS达到全局最优,便成为亟待解决的问题。这类问题通常被称为QoS敏感的Web服务组合问题。例如,在制定外出旅行流程的过程中,旅游的组织者可以在每个旅游景点提供的多个Web服务中进行选择,然后将选择的Web服务组织在一起而得到一个行程。然而,如何在保证能到达所有景点的前提下,自动地在各个景点上的Web服务中进行有效的选择,以最大程度减少行程总的开销,进而组合成一个令人满意的旅游行程,便是一个典型的QoS敏感的Web服务组合问题。

针对上述问题,本文提出了一种基于遗传算法(Genetic Algorithms)的方法来有效地选择Web服务,并通过实验证明了该方法的有效性。与现有的方法相比,本文提出的方法拥有更好的普适性。

本文第2部分简要介绍了相关的研究工作;第3部分描述了业务流程QoS的计算规则;第4部分具体给出了如何通过遗传算法解决QoS敏感的Web服务组合问题;第5部分通过实验对本文提出的方法进行了分析和验证;最后给出总结。

2 相关研究

随着Web服务的流行,近年来在利用Web服务组合得到业务流程的领域,出现了大量的研究工作。M. Agarwal[3]、L. Aversano[4]等提出了利用已有Web服务进行动态组合得到业务流程的方法。

由于Internet环境固有的随机性与动态性,人们往往要求通过组合Web服务得到的业务流程能够符合一定的服务水平协议SLA(Service Level Agreement),即满足QoS约束。因此,对于如何解决QoS敏感的Web服务组合的问题,研究人员提出了很多的探讨。其中一类典型的范型是采用元过程和元服务的概念描述服务组合的结构;然后依据Web服务的QoS,由系统自动发现具体的Web服务来替换元过程中的元服务,从而得到一个具体的QoS最优的业务流程。简单地讲,一个元服务可以看作一组完成相似业务功能的Web服务的集合的代表。因此,对应同一个元服务的具体服务之间互相可以替换。元过程则是由元服务组成的一个抽象过程模型。例如,在前面提到的旅游流程制定过程中,元过程将会描述一个旅游过程需要哪些种类的服务以及包含哪些控制流。

在这类范型中,一些研究人员提出了基于整数规划的方法[1,10]。这种方法假设约束条件和目标函数都具有线性的特点。Zeng[10]的工作着重关注开销、响应时间、有效性和可

靠性等 QoS 属性；并在目标函数的计算过程中采用对数缩变。虽然，Zeng 认为他们提出的模型可以对其它相似的行为属性进行扩展。然而在现实中，流程需要满足的约束更为复杂，往往并不具备线性的特点，例如服务的依赖性和用户的偏好等。因此，Zeng 提出的模型并不具有较好的普适性。Aggarwal [1] 的工作虽然提到了这类非线性约束的需求，但并没有给出相应的解决方法。与这类基于整数规划的方法相比，本文中的方法基于遗传算法，可以处理更多形式的约束，拥有更好的普适性。

这类范型中另一类方法采用的是人工智能的查询策略。这类方法源自人工智能的约束逻辑 CLP (Constraint Logic Programming)，它的优势在于较强的建模能力；具体的研究在 Craenen [5] 和 Helm [7] 的文章中都有体现。由于 CLP 具有约束条件传递的特性，例如用于确定子域变量取值的约束会在整个 Web 服务的查找过程中不断地被用到；因此这类方法得到的查询更加高效，但其本身的开销却相对较大。在 QoS 敏感的 Web 服务组合过程中，由于约束一般数量较少而且均较为简单，因此我们认为采用开销相对较小遗传算法更加适合。

由于遗传算法本身并没有提供描述约束的机制，因此，不少研究工作提出了通过惩罚函数来实现的方法。针对 QoS 敏感的 Web 服务组合的问题，本文采用了文[6]中的一种基于距离的惩罚函数。文[6]中还提出了一些更为灵活的惩罚函数，但这类函数相对较为复杂；而在 Web 服务的选择过程中，约束数量较少，可以避免一些不必要的计算开销，因此，本文中并未采用这类函数。

3 业务流程的 QoS 计算

本文工作的目的是有效地确定流程中使用的 Web 服务集合，已在满足功能需求的前提下，使得整个流程的 QoS 达到最优。这个集合需要具备以下特点：1）满足所有的 QoS 约束，符合 SLA 协议。例如：Web 服务的用户的预算往往是有限的，因此流程的开销就是一种约束。通常，局部约束（如一个具体的操作的开销不能超过给定的限制）和全局约束（如流程总的响应时间作为约束条件）都需要被满足。2）对于指定的 QoS 参数达到最优化。例如：用户可能需要在保证开销低于一定限制的同时使得响应时间最小。

因此，为了根据 QoS 完成 Web 服务的选择，我们需要描述如何从单个 Web 服务的 QoS 属性值计算业务流程，即复合服务的 QoS。

对于一个给定具体 Web 服务的业务流程，表 1 给出了整个流程的 QoS 的计算规则。表 1 列出了各种常见的流程结构和常见的 QoS 属性对应的计算函数。同时，表 1 中的最后一行还给出了流程的用户自定义属性（如域依赖属性）的计算规则。在本文中，由于篇幅的限制，作者仅通过时间、费用、可用性以及可靠性等这四种常见的 QoS 属性来说明、验证文中提出的方法。

接下来我们对各种流程结构的 QoS 属性计算方法进行简要的解释。对于选择结构，每种选择分支都标有被选择的概率。例如：一个流程选择结构有两个分支，其开销分别为 C1，C2，被选择的概率分别为 p，$1-p$，则整个结构的开销计算如下：

$$pc_1+(1-p)c_2$$

概率的初始化值可由业务流程的设计者确定，然后通过监控流程执行过程获取的信息不断更新。

相应地，对具有 n 个分支的选择结构 1，…，n，每个分支的概率对应为 $\{p_1,\cdots,p_n\}$，其中 $\sum_{i=1}^{n} p_i=1$，通常这种选择结构的 QoS 为每个任务的属性值与其对应的分支概率的乘积然后求和。

表 1 工作流结构和 QoS 属性的聚合函数

QoS 属性	顺序(Sequence)	选择(Switch)	并行(Flow)	循环(Loop)
Time(T)	$\sum_{i=1}^{m} T(t_i)$	$\sum_{i=1}^{m} P_{ai} * T(t_i)$	$Max\{T(t_i)_{i\in\{1\cdots p\}}\}$	$k * T(t)$
Cost(C)	$\sum_{i=1}^{m} C(t_i)$	$\sum_{i=1}^{m} P_{ai} * C(t_i)$	$\sum_{i=1}^{p} C(t_i)$	$k * C(t)$
Availability(A)	$\prod_{i=1}^{m} A(t_i)$	$\sum_{i=1}^{m} P_{ai} * A(t_i)$	$\prod_{i=1}^{p} A(t_i)$	$A(t)^k$
Reliability(R)	$\prod_{i=1}^{m} R(t_i)$	$\sum_{i=1}^{m} P_{ai} * R(t_i)$	$\prod_{i=1}^{p} R(t_i)$	$R(t)^k$
Custom Attr. (F)	$f_s(F(t_i))i\in\{1\cdots m\}$	$f_B((P_{ai},F(t_i)))i\in\{1\cdots n\}$	$f_F(F(t_i))i\in\{1\cdots p\}$	$f_L(k,F(t))$

对于循环结构 QoS 的计算，我们采用估计的循环次数 k 的方式。例如，如果一次循环之行开销为 Cl，则估计的循环结构的总开销为 k Cl。与展开循环的方法相比，这种处理循环的方法有可以更加快速、准确地计算整个流程的 QoS。

除了时间属性为{t1，t2，…，tp}的最大值之外，并行结构(Flow)的 QoS 计算规则与顺序结构基本上类似。

4 基于遗传算法的解决方案

遗传算法是基于自然选择和自然遗传机制的搜索算法，是一种有效地解决最优化问题的方法。用遗传算法解最优化问题，首先需要在可行域中挑选一些个体作为染色体进行编码，组成第一代群体，并计算每个染色体的适应度。接下来，算法利用选择机制从群体中随机挑选染色体作为繁殖过程前的样本。一个染色体被选中的概率与其适应度成正比。在接下来的繁殖过程中，遗传算法通过了交叉和变异两种算子对挑选后的染色体进行繁殖处理。交叉算子随机挑选两个染色体并在某些位上进行交换。变异算子则直接在一个染色体中的随机挑选某一位进行突变。这样便产生了下一代群体。算法重复上述选择和繁殖过程，直到结束条件得到满足为止。进化过程的最后一代中便是用遗传算法解最优化问题所得到的最终结果。

与文中提出的线性规划方法等其他方法不同，采用遗传算法不要求 QoS 组合操作（也包括目标函数和约束）具备线性的特征。这就使得我们的方法适合更多的（甚至是用户自定义的）QoS 属性，而不需要进行线性化处理。

为了利用遗传算法来解决 QoS 敏感的 Web 服务组合问

题，我们需要针对具体问题定义如何对染色体进行编码，如何计算染色体的适应度，如何执行交叉和变异算子以及算法的终止条件。为此，首先需要采用合适的染色体对问题进行编码。在 QoS 敏感的 Web 服务组合问题中，染色体描述为一个整型数组 $S=\{S_1, S_2, \cdots, S_n\}$。该数组中的每个元素 S_i 对应于业务流程中的一个子任务。每个元素中包含一个索引，索引指向该元素在一组具有相同或相似功能的 Web 服务中的所选中的 Web 服务 CS_{ij}。图 1 描述了染色体的结构。

相应地，交叉算子中的位交叉是交换两个染色体中某个元素对应的 Web 服务。而变异算子的变异操作则是在某个元素对应的 Web 服务中随机选择新的 Web 服务替换当前的 Web 服务。从这里可以看出，只有一个 Web 服务相对应的染色体元素在遗传进化中不会起到作用，可以不予考虑。

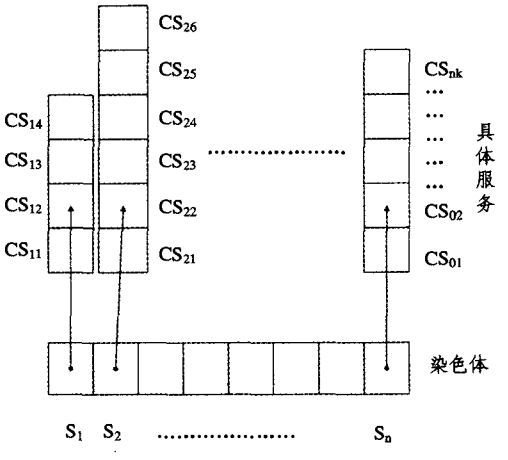

图 1　染色体编码

接下来，我们需要定义适应度函数。根据 QoS 敏感的 Web 服务组合问题的特点，染色体的适应度函数要求使部分 QoS 属性值达到最大，同时使其他部分 QoS 属性值最小。同样，如果用户定义了特定领域的 QoS 属性，那么适应度函数还要根据情况做出进一步的调整。

另外，适应度函数应当能够抑制不满足约束的染色体以引导它们向满足约束的方向进化。我们可以对 Web 服务需要满足的 n 条 QoS 属性的约束给与不同的违反惩罚权重，这里表示为：

$$cl_i(g) \leqslant 0, i=1, \cdots, n$$

这里的 $cl_i(g)$ 表示了违反第 i 条约束对染色体的适应度带来的惩罚值。进一步地，我们将当前染色体违反各类约束所造成的影响 D(g) 定义如下：

$$D(g)=\sum_{i=1}^{n} cl_i(g) * y_i$$

其中如果染色体违反了约束，则 y_i 为 1，否则 y_i 为 0。

根据表 1 中常见的 QoS 属性，我们可以初步地给出染色体 g 的适应度函数的定义：

$$F(g)=\frac{w_1 \text{Availability}(g)+w_2 \text{reliability}(g)}{w_3 \text{Cost}(g)+w_4 \text{ResponseTime}(g)+} W_5 D(g)$$

在上述 $F(g)$ 的计算公式中，各种 QoS 属性（如 Availability，Reliability 等）的计算结果均被正规化为 [0，1) 之间的实数，公式中的 w1，…，w5 均为正实数，作为参数。其中 w1，…，w4 代表用户为具体的 QoS 属性设定的权值，w5 表示惩罚违反约束的整体权值。

上面定义的适应度计算函数包含了对违反约束的染色体所给出的静态惩罚值。换句话说，对于每一代染色体，惩罚值都是一样的。而通常情况下，如果一个惩罚因素的权重很高，进化过程就可能使得接近最优解决方案的染色体在进化的早期因违反约束而被抛弃。

针对这个问题，本文提出了动态惩罚的策略。惩罚权重随着后代数目的增大而增大。这就允许在进化早期，保留一些违反约束但接近最优解决方案的染色体，通过几代进化后，整个群体会满足约束，进化将只改进适应度函数余下的部分。

因此，我们给出的动态适应度函数定义如下：

$$F(g)=\frac{w_1 \text{Availability}(g)+w_2 \text{Relialbility}(g)}{w_3 \text{Cost}(g)+w_4 \text{ResponseTime}(g)}+ w_5 D(g)^* \frac{\text{gen}}{\text{maxen}}$$

其中，gen 为当前后代数，maxgen 为最大后代数。

最后我们还需要定义遗传算法的终止条件。在终止条件的设定上，规定最大的进化次数是一种效率较高的方法。因此，在 QoS 敏感的 Web 服务组合问题中，我们采用如下的终止条件：

设定最大后代数目 MaxGenConstrain，循环直到约束（D(g)=0）被满足终止。如果达到最大后代数目 MaxGenConstrain，约束仍未满足，则算法认为没有相应的解决方案。

一旦 D(g)=0，那么循环将继续进行。此处可将循环的次数可以预定为 MaxGenConstrain 的百分数；也可以循环直至最优化适应染色体在给定的后代数目内不再发生变化为止。

这样，我们便可以借助于染色体来描述 QoS 敏感的 Web 服务组合问题实例；并通过对染色体反复执行选择、交叉和变异操作，来不断对结果进行优化；直至算法的终止条件为真，便可以得到 QoS 敏感的 Web 服务组合问题的最优解。

5　性能分析

本节中将通过实验来分析、验证文中所提出方法的有效性，并通过分析遗传算法产生的后代来观察进化过程中染色体适应度的变化；从而比较静态和动态的适应度函数。

我们采用精英遗传算法，其中每次进化只保留 2 个最优染色体。交叉概率为 0.7，变异概率为 0.01，群体规模为 100，选择机制采用轮盘转法。适应度函数开始为不同的 QoS 属性预置相同的权值。在某些情况下，需要放弃某些属性。

每次实验执行 50 次遗传算法并记录平均结果，与平均值的标准偏移低于 5%。为了观察不同的适应度函数，我们规定当达到规定的进化次数（本实验规定为 100）后代没有改进则中止算法。实验中采用的业务流程包含 25 个服务调用，其中有 16 个不同的任务。对每个任务，分别考察对应有 5、10、15、20、25 个可选的 Web 服务的情况。

图 2 显示了各个 QoS 属性在不同的适应度函数下的进化情况。如图所示，约束要求 Cost 和 ResponseTime 两个属性达到最优，图中显示遗传算法可以找到满足约束的解决方案，同时优化其他的适应属性（即在保持最低开销和最快响应时间的前提下使有效性最大）。在试验中，动态适应度函数并没有表现出明显的优势，均得到相同的结果；仅在有效性属性上动态适应度函数收敛地略快。

图 3 给出了遗传算法与整数规划算法使用的 CPU 时间的比较。通过图 4 可以看出，当可选的 Web 服务较少时，整数规划算法拥有比遗传算法更好的性能。但当可选的 Web 服务较多时，整数规划算法所需的时间会会急剧增加，而遗传

算法的性能则能基本保持不变。因此，与整数规划算法相比，遗传算法还具有更好的可扩展性。

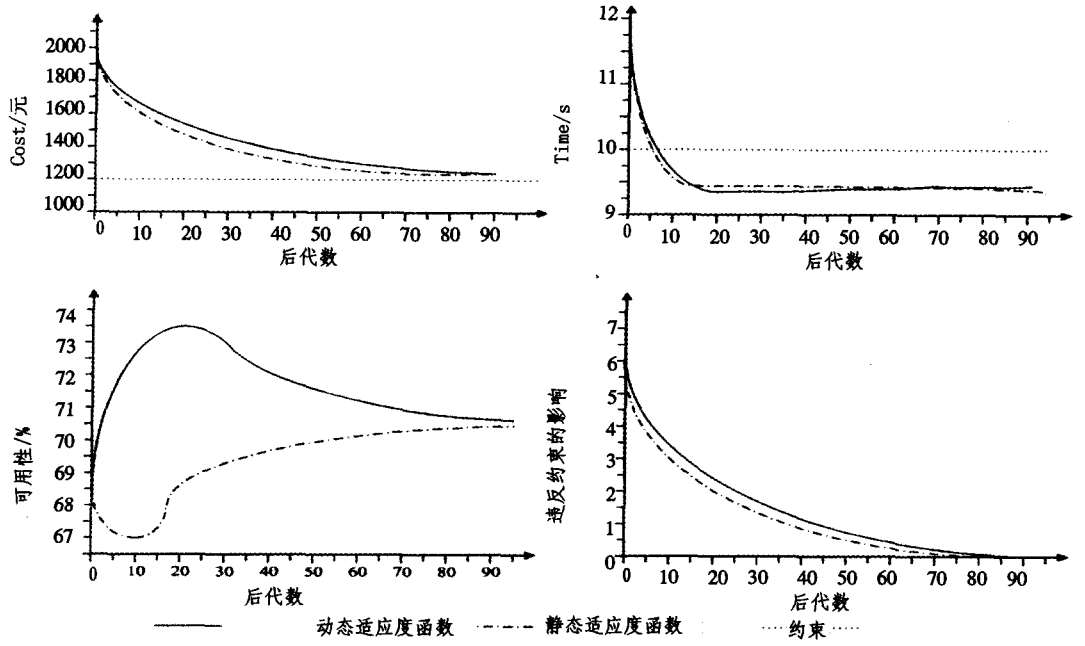

图 2　各参数的不同适应度函数的比较

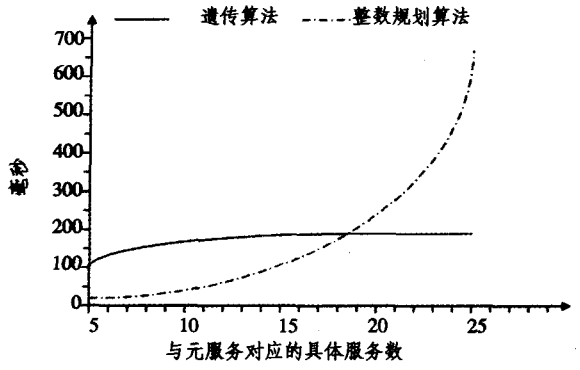

图 3　遗传算法与整数规划算法使用 CPU 时间的比较

总结与展望　本文提出了一种基于遗传算法来的方法来解决 QoS 敏感的 Web 服务组合的问题，即通过遗传算法来为流程中的任务选择合适的 Web 服务，并进行绑定，从而得到一个既能满足约束条件还能使指定 QoS 属性达到最优的业务流程。实验数据证明了本文方法的有效性。

尽管现有的工作如整数规划方法可以解决这个问题，但由于用于评价 Web 服务的各种 QoS 属性一般不具备现有的解决方法所要求的线性特征；因此需要一种更为通用的方法来解决复杂属性的 Web 服务的组合问题。而遗传算法并不受线性关系的约束，具有更好的普适性。另外，现有的解决方法会使问题的规模随着具体服务的增加而急剧增大。而如果采用遗传算法，可选 Web 服务的增加仅会增大算法的搜索范围，对问题规模的增加并不大。所以，采用遗传算法解决 QoS 敏感的服务组合问题具有更好的可扩展性。

文中同时提出用动态适应度函数在进化早期减少约束条件对群体中优良染色体的影响，增加优良染色体的竞争力。尽管在实验中表现得并不明显，但还是可以看出动态适应度函数能够提高服务组合的适应度。

在将来的工作中，我们将进一步对适应度函数进行优化；并考虑通过遗传算法来解决 Web 服务组合过程中元过程的自动生成问题。

参 考 文 献

1　Aggarwal R, Verma K, Miller J, Milnor W. Constraint driven Web service composition in METEOR-S. In: Proceeding of the 2004 IEEE International Cnference on Services Computing, (SCC), 2004. 23～30

2　Cardoso J. Quality of service and semantic composition of workflows:[PhD thesis]. University of Georgia, 2002

3　Agarwal M, Parashar M. Enabling autonomic compositions in grid environments. In: Proceedings of the 4th International Workshop on Grid Computing (GRID), 2003. 34～41

4　Aversano L, Canfora G, Ciampi A. An algorithm for Web service discover through their composition. In: Proceedings of the 2004 IEEE International Conference on Web services (ICWS), 2004. 332～339

5　Craenen B, Eiben A, van Hemert J. Comparing evolutionary algorithms on binary constraint satisfaction problems. IEEE Transactions on Evolutionary Computation, 2003, 7(5):281～308

6　Fang H. Genetic algorithms in time abling and scheduling:[PhD thesis]. University of Edimburg, 1994

7　Helm T, Painter S, Oakes W. A comparison of three optimization methods for scheduling maintenance of high cost, long-lived capital assets. In: Proceedings of the 34th conference on Winter Simulation: exploring new frontiers, 2002. 1880～1884

8　ISO. UNE-EN-ISO, ISO 8402 (Part of the ISO 9000 2002): Quality Vocabulary

9　ITU Recommendation. E. 800 Quality of service and dependability vocabulary

10　Zeng L, Benatallah B, Ngu A, Dumas M, Kalagnanam J, Chang H. QoS-aware middleware for Web services composition. IEEE Transactions on Software Engineering, 2004,30(5):311～327