

动态的 QoS 感知 Web 服务选择和组合优化模型

蒋哲远 韩江洪 王 钊

(合肥工业大学计算机与信息学院 合肥 230009)

摘 要 Web 服务软件工程的实用化挑战之一是 QoS 感知的选择、组合和稍后的绑定,表现为允许在运行时绑定一组领域 Web 服务构成面向服务的系统.这些领域 Web 服务在提供要求的功能同时,还满足一些非功能约束,例如总的费用或响应时间,并且使之最优化.对此作者提出了一种把 Web 服务看作为首类构件对象的关系查询基础结构,它通过各种 Web 服务操作调用评估查询.鉴于个性化和效率在这种评估中的重要作用,提出了一个基于聚合不同 Web 服务的多属性 QoS 参数的查询优化模型.该模型通过用户设定的全局约束和偏好、一个动态的等级方案以及多级匹配来调整 QoS.等级提供了一个 Web 服务的行为评估,而多级匹配通过使用类似的和部分的答案对解决方案的空间进行扩展.进而给出了模型求解的遗传算法,并从适应度函数的静态惩罚、动态惩罚以及拉伸 3 个方面对优化性能进行了比较.文中最后介绍了一个从高端实现的服务查询引擎原型系统,用以展示该方法的适应性、可行性和有效性.

关键词 Web 服务;服务质量;服务选择;约束优化;遗传算法

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2009.01014

An Optimization Model for Dynamic QoS-Aware Web Services Selection and Composition

JIANG Zhe-Yuan HAN Jiang-Hong WANG Zhao

(School of Computer and Information, Hefei University of Technology, Hefei 230009)

Abstract One of the most promising opportunities from a Web services engineering perspective is the QoS-aware selection, composition and late-binding. This allows you to dynamically assemble a collection of domain-specific QoS-aware Web services providing the required features into a composition services that can meet some non-functional constraints, and optimize criteria such as the overall cost or response time. This paper presents a query infrastructure that considers Web services as first class component objects. It evaluates queries through the invocations of different Web services operations. Because personality and efficiency play a central role in such evaluations, the paper proposes a query optimization model based on aggregating the multi-attribute QoS parameters of different Web services. The model adjusts QoS through global constraints and preferences set by the user, a dynamic rating scheme, and multilevel matching. The rating gives an assessment of Web services behaviors. Multilevel matching provides the expansion of the solution space by enabling similar and partial answers. The paper describes a genetic algorithm for solving the model, and compares the optimization performance of the genetic algorithm using various fitness functions varying in terms of static penalty, dynamic penalty, and stretching. The

收稿日期:2006-08-08;最终修改稿收到日期:2009-04-10. 本课题得到国家“八六三”高技术研究发展计划项目基金(2002AA415280)、教育部博士点基金项目(20050359004)、教育部新世纪优秀人才计划项目(NCET-04-050562)、合肥工业大学科学研究发展基金项目(2007GDBJ012)资助. 蒋哲远,男,1966年生,博士,副研究员,主要研究方向为面向服务软件工程、软件体系结构和软件工程环境. E-mail: jzheyuan@mail.hf.ah.cn. 韩江洪,男,1954年生,教授,博士生导师,主要研究领域为智能控制技术和分布式系统. 王钊,男,1958年生,副教授,主要研究方向为电子商务和软件工程.

proposed approach has been applied to a service query engine system for SEIFCW, which consequently shows its applicability, feasibility and efficiency.

Keywords Web services; quality of service (QoS); service selection; constraint optimization; genetic algorithm

1 引言

面向 Internet 的 Web 服务应用, 比如信息门户、实时网格计算和电子商务应用等, 越来越关注提供给用户的服务质量(Quality of Service, QoS)^[1]. 从 Web 服务发现的角度看, 如何从大量的 Web 服务中有效地快速选择和集成最适合用户需求的、高可信性的、QoS 感知的 Web 服务成为面向服务软件工程的一个新的挑战.

面向服务软件工程代表了基于构件软件工程(Component-Based Software Engineering, CBSE)的自然演化. 在 CBSE 中, 一个构件集成者搜索可重用构件, 然后使用一些胶合代码(glue code)^[2]把它们集成到一个实际的新系统中. Web 服务由服务提供者在服务注册中心发布其功能, 并通过远程服务调用操作供服务请求者绑定使用, 因而没有必要在开发时就把服务集成到应用中. Web 服务的接口发布、发现和调用都是执行基于 XML 的一些标准, 例如 WSDL、UDDI 和 SOAP. 因此, 一个面向服务的系统是由一些服务调用构成, 并使用胶合代码或者一些具体的 Web 服务组合语言(如 BPEL4WS)来合成.

面向服务系统的最大期许是使用稍后的绑定机制. 事实上, 一些关键的业务应用场景(如面向服务 ERP 协同集成框架 SEIFCW^[3]中的组装领域服务构件)是给出某个领域服务的所需功能单元的具体特性描述(这个特色描述将据此作为一个抽象服务), 且在组装实现时有多个符合该特性描述的服务(称为具体服务)可利用. 一个抽象服务和对应的所有具体服务在功能上是等价的, 因而可相互取代. 它们之间的选择由非功能属性来确定, 例如 QoS 属性. 一个用户可决定挑选最便宜的和最快的服务, 或者是两者之间的折中. 参照文献[4], QoS 的属性包括费用、响应时间、可用性、声誉. 此外, 还可能有一些领域特定的 QoS 属性, 例如一个运输服务的 QoS 属性可能有温度或湿度. 通常, 一个用户可能对某些 QoS 属性给出限制值, 例如费用不能大于某个给定值, 进而影响服务的选择. 另一方面, 服务提供者

可以把 QoS 属性值的估计范围作为和潜在用户之间契约的一部分, 例如服务等级协定(Service Level Agreement, SLA). SLA 是用户与服务提供者之间达成的有关服务内容、服务质量等方面的合约, 它规定了服务提供者必须为用户提供的具体服务参数. SLA 评价的最大优点是赋予用户获知系统性能的权利, SLA 评价也为系统提供者提供了一种竞争依据.

服务查询引擎(service query engine)是 SEIFCW 的核心模块之一, 它的主要功能包括领域服务查询请求提交、服务检索、服务选择、服务组合并生成基于 XML 的服务组合描述语言等. 由于 QoS 对于 Web 服务在领域的成功应用非常关键, 因此, 如何使查询的组合服务提供预期的服务质量正成为服务查询引擎设计面临的重要挑战. 其中, 一个有关的问题是如何使抽象服务与具体服务快速绑定, 且使实际的 Web 服务满足 SLA 规定的 QoS 约束, 并使服务集成者所选择的一些适应度标准最优化(例如, 费用最小). 另外, 求解 QoS 感知组合问题的解是一个 NP 难题^[5]. 一些方法, 主要是基于线性规划, 已经在一些文献中提出^[6]. 为此, 本文提出了一种新颖的服务查询和优化模型用于保障 SEIFCW 应用的 QoS 感知的服务组合, 并用遗传算法(Genetic Algorithm, GA)来解决 QoS 感知的组合问题. 所提模型的主要特点是: (1) 使用关系表达查询请求; (2) 领域服务的抽象化; (3) 可扩展的 Web 服务 QoS 度量; (4) 基于多度量的组合服务 QoS 优化.

本文第 2 节介绍一种基于关系和抽象操作的 QoS 感知的 Web 服务多层查询体系结构; 第 3 节建立 QoS 感知的 Web 服务查询优化数学模型, 并使用遗传算法对其目标进行求解; 第 4 节给出模型在 SEIFCW 中的实现, 并报告和讨论了仿真的结果; 第 5 节介绍一些相关工作, 并和相关研究作比较; 第 6 节是结论, 同时对未来的工作进行展望.

2 Web 服务的查询模式

Web 服务的应用主要由包括发送和接收消息的一组调用操作组成. 尽管这可满足简单应用的需

求,但对访问千变万化的 Web 服务的复杂应用,需要使用一种集成方式去操作与交付 Web 服务的功能.此外,随着 Web 服务的广泛应用,提供相同功能的服务越来越多.但由于竞争,Web 服务提供的这些功能(例如,要求不同的输入和输出参数)和使用条件(QoS 的等级不同)的方式是有区别的.另外,满足用户的需求有时并不需要返回精确的答案.事实上,用户可能对可供选择的部分答案满意.

针对上述挑战,需要提出一个在操作调用期间查询 Web 服务和信息流的新方法,以便决定各种 Web 服务操作的调用方式,返回相应输出结果的整合.该方法强调查询决策过程的优化,同时应允许部分的答案.

2.1 Web 服务查询模式的体系结构

为了方便用户和 Web 服务的交互,需要使用一个通用的方法表达给定应用领域的 Web 服务空间.为此,我们提出了图 1 所示的 3 层查询模式,并定义了一组称为抽象操作的特定领域操作,它是对特定领域 Web 服务应用共用操作的抽象化(虚拟化),不属于任何实际的 Web 服务.

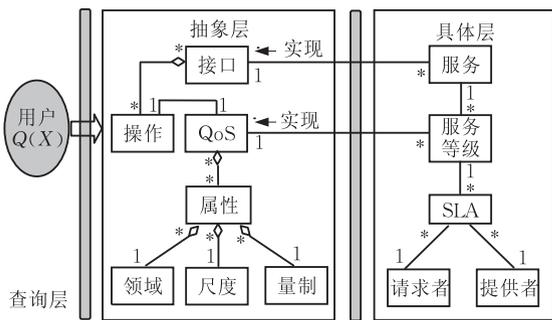


图 1 QoS 感知服务查询的三层体系结构

(1) 查询层(query layer). 它由提供给用户一个表达和提交查询语句到 Web 服务接口的关系集组成,可使用一种从 XML 文档中抽取数据的查询语言规范 XQuery 来构造.用户从查询层提交关系联合查询(关系和条件的联合),其通用形式如下:

$$Q(X) := \bigwedge_i R_i(X_i), \bigwedge_k C_k,$$

其中,符号 $:=$ 表示映射, R_i 是来自查询层的关系, X 和 X_i 是变量元组; C_k 代表查询中出现的变量条件.条件的形式是 $C_k = x \text{ op } c$, 式中 x 是出现在任何一个 X_i 中的输入或输出变量,或者是 QoS 规格的元素(参见第 3 节), c 是一个常量, $\text{op} \in \{=, \neq, <, >, \geq, \leq\}$. 查询模型根据各种映射规则把每个关系映射到对应的抽象操作中.

(2) 抽象层(abstract layer). 它由一个特定领域

典型提供的类 Web 服务操作组成,可采用基于 XML 的 WSDL 对包含其操作的服务接口进行描述.标准 WSDL 不提供对 QoS 的支持,需要对 WSDL 进行扩展,以便能使 WSDL 恰当地提供 QoS 信息.例如,通过扩展 WSDL 提供一个服务提供者和请求者之间的 Web 服务 SLA 文档引用,可使 WSDL 的每个操作和一个由 Web 服务 SLA 定义的 QoS 相关联,其格式为

```
<operation name="opName" WSLA="WSLAOfferedURL">
  <input message="tns:opName"/>
  <output message="tns:opNameResponse"/>
</operation>
```

QoS 是一个属性集,它的每一个属性,例如领域(domain)、尺度(scale)和量制(metrics)等需要根据具体应用场景来精确定义.这些 QoS 属性的抽象集都需要绑定到某个接口操作,以便规定它的应用属性集.

(3) 具体层(concrete layer). 其代表 Web 上提供的 Web 服务空间,即潜在的目标查询候选者,包括元服务与组合服务,且与服务等级(service level)有关.当一个服务提供者在注册中心的数据库保存他的服务和等级时,其应用范围相应地被定义.一个提供者和一个请求者能协商更精确的属性,并创建一个 SLA 作为契约.由于 Web 服务开始是未知的,查询模型需要发现它们,并与出现在查询中的抽象操作相匹配.

使用该查询模式的优点之一是容易使用 UML 的模式驱动方法(model-driven methodology)建模和开发实现^[7].下面是抽象层到具体层的转换定义.

定义 1. 假设抽象服务接口 I 的 QoS 集定义为如下的一个 P_i 属性集: $Q_I = \{P_1, P_2, \dots, P_n\}$. 其中,对每个 $P_i \in Q_I (1 \leq i \leq n)$ 都有一个领域 $\text{dom}(P_i)$ 和尺度 $\text{scale}(P_i)$ 的度量值.当然,属性被度量或赋值的前提条件是首先定义好每个属性的量制 $\text{metric}(P_i)$. 此外,假设 $\text{dom}(Q_I) = \bigcup_{P \in Q_I} \text{dom}(P)$. 那么,一个接口 I 到一个 QoS 等级为 L_S 的具体服务 S_i 的映射定义如下:

$$\psi: Q_I \rightarrow \text{dom}(Q_I) \mid (\forall P \in Q_I) \psi(P) \in \text{dom}(P),$$

式中符号“ \rightarrow ”是条件连接词,符号“ \mid ”的右边谓词逻辑规定了映射需满足的约束.

2.2 关系到抽象操作的映射

查询层的关系定义了应用域一个具体的视图,可表达为针对抽象操作的连接查询,似乎抽象操作就是关系.

定义 2. 假设 R 是查询层定义的关系集合, V 是抽象操作集合, 则对任何关系 $R_i \in R$,

$$R_j(x_1, x_2, \dots, x_n) :- \bigwedge_j V_j(y_{j_1}, y_{j_2}, \dots, y_{j_m}),$$

式中 x_i 是 R_i 的属性, $V_j \in V$, y_{j_i} 是相应操作的输入和输出变量. 这个定义意味着为了获得 R_i 的元组, 需要调用不同的操作 V_j , 但并不要求对这些操作做任何的排序或限制并发.

用户能直接地使用抽象操作访问 Web 服务, 但关系的使用有两个好处. 首先, 让用户使用一个自然的方式表达和提交类似数据库的查询; 另一方面, 为对服务空间一些特定部分感兴趣的某个特定用户组提供了量身定制的视图.

2.3 抽象操作表示

抽象操作代表了一个特定应用域所能提供的典型功能. 对在查询中出现的任何抽象操作, 最终都要定位到相关的 Web 服务操作, 因此抽象操作需要有一个语义描述和句法属性, 同时还要有描述具体 Web 服务和抽象操作的一个共同的本体(ontology)^[8].

任何一个操作, 无论是抽象的还是具体的, 语义上都可以通过它的功能(function)和类别(category)来描述. 功能包含两个属性: (1) 功能性(functionality), 代表该操作所提供的业务功能; (2) 术语(term), 包含该操作的一个可替代功能命名术语列表. 类别同样也包含两个属性: (1) 领域(domain), 给出操作的具体应用领域; (2) 术语, 抽取与具体应用领域类似的领域分类术语列表.

任何一个操作在调用前都要求它的输入变量被赋值. 在构造查询时, 用户可为不同的变量自由地指定任意类型的条件. 这可能导致一种情形, 就是系统不能调用操作, 这是由于一些输入变量遗漏了赋值. 如果所有的抽象操作描述都给出了其输入变量的各种可能值, 上述问题就会解决(尽管对所有的输入变量不可能总是做到这一点). 本文扩展每个输入变量到所有可能取值, 以便尽量满足出现在查询中的任意一个条件.

下面的五元组形式化地表达了每个抽象操作:

$$V_{op} = \langle In, Out, Restriction, Category, Function \rangle,$$

上式中, In 是输入变量的集合; Out 是输出变量的集合; $Restriction$ 是输入变量的定义域, 它是由一组对偶 $(x, range)$ 构成的集合, 其中 x 是 In 中的枚举变量, 而 $range$ 是 x 的所有可能值的集合; $Category$ 描述兴趣域; $Function$ 描述业务功能.

2.4 抽象操作的匹配类型

由于 Web 服务提供者之间的竞争, 必然在请求

输入、返回输出和 QoS 等方面存在差异. 例如, 不同运输服务提供者的运输费用、快捷性和服务内容等可能有所区别. 这就意味着对某个给定的抽象操作并不总能找到一个精确的匹配. 作为替代, 仅查找与联合查询中出现的抽象操作精确匹配的具体操作, 一个允许抽象操作和具体操作的属性不一致的更灵活的匹配方案可能更满足用户的需求.

首先定义一个相似函数 *Similarity* 来检查两个操作中的两个属性是否相似: 如果 x 和 y 符合应用域本体库所定义的相似概念, 那么 *Similarity*(x, y) 就为真.

定义 3(操作的输入变量集相等). 对任意的两个操作 op_1 和 op_2 , $In(op_1) = In(op_2)$, 当且仅当:

(1) $In(op_1)$ 和 $In(op_2)$ 变量数相同, 且

(2) $\forall x \in In(op_1), \exists y \in In(op_2)$, 或者 $\forall x \in In(op_2), \exists y \in In(op_1)$, 使 *Similarity*(x, y) 满足真值.

类似地能定义 $Out(op_1) = Out(op_2)$. 用同样的方法, 也可以分别定义 $In(op_1) \subset In(op_2)$ 和 $Out(op_1) \subset Out(op_2)$, 其中第 1 个集合是第 2 个集合的子集. 假如 v 和 c 分别表示抽象操作和具体操作, 通过改变抽象操作和实际操作比较方式, 可以识别出 4 种不同的匹配类型.

定义 4(抽象操作匹配类型). 如果两个操作 v 和 c 有相同的输入和输出变量以及相同的 *Category* 和 *Function*, 称之为精确匹配 *Exact*(v, c); 如果两个操作 v 和 c 有同样的输入和输出变量, 且它们的 *Category* 和 *Function* 发生交叠, 称为重叠匹配 *overlap*(v, c); 假如两个操作 v 和 c 有同样的 *Category* 和 *Function*, 且 $Out(c) \subseteq Out(v)$ 或者 $In(c) \subseteq In(v)$, 称为部分匹配 *partial*(v, c); 如果两个操作 c 和 v 满足 $Out(c) \subseteq Out(v)$ 或者 $In(c) \subseteq In(v)$, 且它们的 *Category* 和 *Function* 属性交叠, 称为部分和重叠匹配 *Composite*(v, c).

由定义 4 可知, 精确匹配要求具体操作与抽象操作的每个属性概念都相同, 这是最高级别的匹配; 重叠匹配提供抽象操作的相近功能的具体操作; 部分匹配符合两个操作的输入和输出属性不一致的情形; 部分和重叠匹配组合了重叠匹配和部分匹配, 并且各种匹配存在如下关系.

命题 1. “ \Rightarrow ”表示蕴含关系, “ \wedge ”表示逻辑析取关系, “ \vee ”表示逻辑合取关系, 则存在

$$Exact(v, c) \Rightarrow partial(v, c) \wedge overlap(v, c);$$

$$Exact(v, c) \vee partial(v, c) \vee overlap(v, c) \Rightarrow$$

$Composite(v, c)$.

为了量化各种匹配类型的匹配精确度,还需要给每一个匹配类型赋予了一个匹配度(matching degree),以此直接地影响查询结果的质量.上述每个匹配类型分别获得 1、3/4、1/2 和 1/4 的匹配度.该取值是任意指定的,主要是为了区分不同的匹配精确度.

3 基于遗传算法的 Web 服务查询优化方法描述

由于存在不同 Web 服务的几个服务执行计划能潜在地解决相同的查询,因此,需要为在所有可能的执行规划中挑选最优的 Web 服务建立适当的标准.相互竞争的 Web 服务之间的一个关键特色是他们的 QoS,它包含几个定量的和定性的参数,该参数度量了 Web 服务所发布的功能是否好.现在已经建立了描述响应时间(time)、服务费用(cost)、可用性(availability)和可靠性(reliability)的通用 QoS 规格^[4].对于领域具体 QoS 规格要根据具体领域进行定义,本文利用可扩展本体技术建立了领域内专门的 QoS 规格.通常,每个 Web 服务都包含一组操作,调用的最小粒度为操作(operation).Web 服务 ws 的操作 op 的 QoS 定义如下:

$$QoS(ws, op) = \langle T(ws, op), C(ws, op), R(ws, op), A(ws, op), U(ws, op) \rangle,$$

其各参数的含义为 $T(ws, op)$ 表示执行时间; $C(ws, op)$ 表示费用; $R(ws, op)$ 表示可靠性; $A(ws, op)$ 表示可用性; $U(ws, op)$ 表示领域 QoS 规格, 定义为 $(U_1(ws, op), U_2(ws, op), \dots, U_n(ws, op))$, 即 n 个用户定义的属性组成的向量.为便于 QoS 模型的建立和求解,把某个 Web 服务 ws 的调用操作 op 记为任务 t ,则上面给出的 QoS 模型可简化为如下的一个 5 元组,即

$$QoS(t) = \langle T(t), C(t), R(t), A(t), U(t) \rangle.$$

每个质量标准都有相应的计算和评价方法,并且 QoS 模型有一个总体评价方法.

优化过程的目的是使下面 QoS 参数的每一个值为最大或最小:

(1) 响应时间(T). 它代表一个服务操作在被调用后平均返回结果的时间,包括服务计算时间(T_{com})、中间件开销(T_{mid})和往返通信时间(T_{net}).总的响应时间 $T(t) = T_{com}(t) + T_{mid}(t) + T_{net}(t)$.

(2) 服务费用(C). 它是一个 Web 服务消费者

每次服务调用所必须支付的相关费用.

(3) 可靠性(R). 它表示服务操作可用的概率,其定义如下:

$$R(t) = \frac{MTTF(t)}{MTTF(t) + MTTR(t)},$$

其中 $MTTF(t)$ 表示操作的平均无故障时间, $MTTR(t)$ 表示操作的平均修复时间.

(4) 可用性(A). 它代表一个 Web 服务正常运行的概率.取值越大意味着有较高的可用性,而较小的取值表示可用性低.由于 Web 服务是按次进行调用的,可以采用离散时间模型来描述其可用性:

$$A(t) = \lim_{n \rightarrow \infty} \frac{\sum_{i=1}^n \delta(t)_i}{n},$$

其中 $\delta(t)_i$ 定义如下:

$$\delta(t)_i = \begin{cases} 1, & \text{如果第 } i \text{ 次操作返回成功结果} \\ 0, & \text{否则} \end{cases}$$

(5) 用户定义(U). 它表示服务消费者定义的领域 QoS 规格,如声誉(reputation)、安全性(security)和互操作性(interoperability)等.

优化的目标是使负向参数(如反应时间和费用)减少到最小值,同时使正向参数(如可用性和可靠性)取得最大值.

正如引言所描述的那样,本工作的一个目标是基于 GA 的方法,能迅速地确定绑定到抽象服务的一个具体服务集,构成一个组合服务的工作流.其中的需求包括:

(1) 满足基于 SLA 的 QoS 约束.例如,服务的用户可能有一个有限的预算,因此服务费用受到一定的约束,或者响应时间不能超过某个约定.时常,局部限制(例如,特定操作的费用不能超过限制)和全局限制(例如,总的响应时间受约束)需要同时被满足;

(2) 使一些特别的 QoS 参数的某个函数最优化.例如,用户可能想使响应时间最段,同时保持费用低于某个限制.

随之而来,还应该考虑由 N 个抽象服务(操作)构成的一个组合服务 $S = \{ws_1, ws_2, \dots, ws_N\}$,其结构通过一些工作流描述语言来定义.每个元素 ws_i 能被绑定到 M 个具体服务 cs_{i1}, \dots, cs_{iM} ,它们在功能上是等价的.

在描述使用 GA 找到最优化问题的解之前,还需要描述怎样计算一个组合服务的 QoS,这要从元服务的 QoS 属性值计算开始.

3.1 计算组合服务的 QoS

计算组合服务的 QoS 方法类似文献[4]提出的方法,工作流的描述语言限定为 BPEL4WS. 对工作流中的一个 Switch 构造,每个 Case 语句被标注为可能的选择. 例如,对一个包含 Switch 的工作流,它由两个 Case 语句组成,费用分别是 C_1 和 C_2 且出现的概率为 p 和 $1-p$,则总的费用计算公式如下:

$$pC_1 + (1-p)C_2.$$

显然,概率是由 workflow 设计者初始指定,最终还需要通过监测 workflow 执行所获得的信息加以更新.

While 不同于文献[4]的 Loop 处理,主要的建议采纳一个到达 Switch 构造的机制(基于进入/退出 While 的概率). 我们的方法类似文献[6]所采用的方法. 例如,While 被标记为一个估计的迭代数 k . 代替展开的 While,这里的 While 的 QoS 计算考虑

迭代因子 k . 例如,假如 While 复合体有一个代价 C_i ,那么 While 的估计代价将是 kC_i . 这种处理 While 方法有两个优势:(1)能迅速计算总的工作流 QoS,而不必展开 While;(2)被估计的 QoS 反映了 While 的迭代数.

表 1 显示了单个工作流构造的 QoS 属性的聚合函数. 一个组合服务的具体应用例子就是一个组合服务的描述,在该描述里,每个抽象服务被绑定到某个相应的具体服务中,总的 QoS 能通过应用表 1 描述的规则被计算. 尽管对一些标准的 QoS 属性聚合函数已经被清楚地指定了^[4,6],但可能有其它属性(例如,依赖领域的属性)需要用户基于抽象操作本体库来定义,其单个工作流的聚合函数也要用户指定(见表 1 的最后一行),并和标准的 QoS 属性一样发布到服务注册中心的 WS-QoS 本体库中(见图 3).

表 1 计算单个工作流构造的 QoS 属性的聚集函数

QoS 属性	Sequence	Switch	Flow	While
$T(t)$	$\sum_{i=1}^m T(t_i)$	$\sum_{i=1}^n p_{ai} \times T(t_i)$	$\text{Max}\{T(t_i) i \in \{1, 2, \dots, p\}\}$	$k \times T(t)$
$C(t)$	$\sum_{i=1}^m C(t_i)$	$\sum_{i=1}^n p_{ai} \times C(t_i)$	$\sum_{i=1}^p C(t_i)$	$k \times C(t)$
$A(t)$	$\prod_{i=1}^m A(t_i)$	$\sum_{i=1}^n p_{ai} \times A(t_i)$	$\prod_{i=1}^p A(t_i)$	$A(t)^k$
$R(t)$	$\prod_{i=1}^m R(t_i)$	$\sum_{i=1}^n p_{ai} \times R(t_i)$	$\prod_{i=1}^p R(t_i)$	$R(t)^k$
$U(t)$	$f_s(U(t_i)), i \in \{1, 2, \dots, m\}$	$f_B((p_{ai}, U(t_i))), i \in \{1, 2, \dots, n\}$	$f_F(U(t_i)), i \in \{1, 2, \dots, p\}$	$f_L(k, U(t))$

表 1 并不是完备的,它只包含原型系统要使用的规则,且除了 While,聚合函数和文献[4]所提出的计算方法基本一致. 这些函数能被递归地定义在工作流的复合节点,即对任务 $\{t_1, t_2, \dots, t_m\}$ 的一个 Sequence 构造,响应时间和费用函数是加法,而可用性和可靠性是乘法. Switch 构造的 Case $1, 2, \dots, n$, 其相应概率分别为 $p_{a1}, p_{a2}, \dots, p_{an}$, 并满足 $\sum_{i=1}^n p_{ai} = 1$, 任务分别为 $\{t_1, t_2, \dots, t_n\}$, 总是被估计为每个任务的属性值之和,乘上它所属 Case 的概率. Flow 构造的聚合函数基本上同 Sequence 构造相同,除了时间属性,它是并行任务 $\{t_1, t_2, \dots, t_p\}$ 的最大值. 最后,具有任务 t 迭代 k 次的 While 构造等于 k 次 t 的 Sequence 构造.

3.2 Web 服务等级

一个 Web 服务在其生命期可能存在起伏,因而它不可能完全实现 SLA 所广告的 QoS 参数. 一般来说,大部分用户能接受 QoS 参数的实际值和广告值之间的细微偏差. 然而,巨大的差距就表明了

Web 服务所提交功能的性能正在下降. 由于上述原因,系统需要对被调用的 Web 服务的 QoS 参数实施监控. 本质上, QoS 参数波动的度量是给 Web 服务提供一个评估,而这种评估在优化过程中扮演了一个重要的角色.

一个用户每次选择和调用一个 Web 服务的某个操作 op 时,系统度量期望的 QoS 和实际发布的 QoS 之间的距离. 此处的 QoS 距离是所有 QoS 参数的偏差(或它们的倒数)加权之和. 更精确地说,假设 pQ_i 和 dQ_i 分别是 QoS 的第 i 个参数的期望值和实际交付值, pos 和 neg 分别是取最大值和最小值的 QoS 参数序号集, ω_i 指一个服务集成者(或用户)给一个特定的 QoS 属性所赋予的重要性,则 QoS 距离(D_{QoS})的计算公式如下:

$$D_{QoS}(op) = \sum_{i \in pos} \omega_i (dQ_i(op) - pQ_i(op)) + \sum_{i \in neg} \omega_i \left(\frac{1}{dQ_i(op)} - \frac{1}{pQ_i(op)} \right) \quad (1)$$

Web 服务等级用下面公式来刻画:

$$rating(op) = \begin{cases} 0.5 + \frac{1}{1 + e^{-q(D_{QoS}(op) - \theta_-)}}, & D_{QoS}(op) < \theta_- \\ 1, & \theta_- \leq D_{QoS}(op) \leq \theta_+ \\ 0.5 + \frac{1}{1 + e^{-q(D_{QoS}(op) - \theta_+)}}, & D_{QoS}(op) > \theta_+ \end{cases} \quad (2)$$

式(2)中的负阈值 θ_- 和正阈值 θ_+ 分别称为置信下限和置信上限; q 为调整 QoS 距离波动参数, 通常取值为 1. 公式表明: 如果一个 Web 服务的 QoS 距离满足 SLA 所规定的 QoS 参数波动区间 (θ_-, θ_+) , 那么它接收正常的等级 1; 如果一个 Web 服务的 QoS 距离低于负阈值 θ_- , 那么它的等级将减少; 如果一个 Web 服务的 QoS 距离高于正阈值 θ_+ , 那么它的等级将增加. 一个简单的 Web 服务等级把所有的 QoS 波动归纳到单一的公式中. 通过分配给每个参数的各自等级, 我们能分别处理每个服务的 QoS. 下节将使用每个参数的等级给目标函数中相应的 QoS 赋权值, 或聚集所有等级, 并用它们度量整个优化公式.

在 Web 中, 目前已存在多个等级方案系统, 其实现主要是依赖用户的反馈, 包括消费者对产品和服务的评价以及来自提供者的个人体验. 本文基于图 1 的查询基础结构, 使用了一个专用 Agent^[9] 负责对 Web 服务操作调用的监控, 从而能方便地定期更新并细化自己的服务等级.

3.3 目标函数和约束条件

Web 服务是否成功地发布了它们的功能主要取决于 QoS. 因而, 优化过程的目标是选择和使用具有最优 QoS 参数的 Web 服务. 另外两个参数等级 (rating) 和匹配度 (md) 进一步细化了服务功能发布成功的定义. 因此, 我们能通过下面 3 个度量, 即期望的 QoS 参数、等级和匹配度, 来刻画解决一个查询所涉及的任何 Web 服务操作.

已存在计算一个基于它的 QoS 参数的操作总体质量的各种方法. 在决策过程中广泛使用的简单添加权重方法所取得的结果通常非常接近更为复杂的方法所取得的排序结果. 该方法包括以下 3 个基本步骤, 操作质量是最后一步获得的数值之和.

(1) 对 L 个不同的 QoS 参数标准化, 以便使它们可比较.

(2) 为参与查询的每一个 QoS 参数施加用户指定的权重 ω_i , 并且满足 $\omega_i > 0, \sum_{i=1}^L \omega_i = 1$.

(3) 为每一个操作添加经过加权和标准化处理

后的 QoS 参数.

为解析一个查询过程中的每个抽象操作 op_j (对应匹配的具体服务数组项目索引), 只需要定义一个取最大值的目标函数 $f_j(op_j)$:

$$f_j(op_j) = rating(op_j) \times md(op_j) \times \left(\sum_{i \in neg} \omega_i N_i(op_j) + \sum_{i \in pos} \omega_i P_i(op_j) \right) \quad (3)$$

式中

$$N_i(op_j) = \begin{cases} \frac{pQ_i^{\max}(op_j) - pQ_i(op_j)}{pQ_i^{\max}(op_j) - pQ_i^{\min}(op_j)}, & pQ_i^{\max}(op_j) - pQ_i^{\min}(op_j) \neq 0 \\ 1, & \text{否则} \end{cases} \quad (4)$$

$$P_i(op_j) = \begin{cases} \frac{pQ_i(op_j) - pQ_i^{\min}(op_j)}{pQ_i^{\max}(op_j) - pQ_i^{\min}(op_j)}, & pQ_i^{\max} - pQ_i^{\min} \neq 0 \\ 1, & \text{否则} \end{cases} \quad (5)$$

其中 $pQ_i^{\max}(op_j)$ 是匹配相同抽象操作 op_j 的所有具体操作中第 i 个 QoS 参数的最大值, 而 $pQ_i^{\min}(op_j)$ 是其最小值.

对目标 $f_j(op_j)$ ($j=1, 2, \dots, N$) 正规化处理, 设 $f_j^{\max}(op_j)$ 和 $f_j^{\min}(op_j)$ 分别为目标 $f_j(op_j)$ 的最大值和最小值, 线性正规化后的目标值 $v_j(op_j)$ 为

$$v_j(op_j) = \frac{f_j(op_j) - f_j^{\min}(op_j)}{f_j^{\max}(op_j) - f_j^{\min}(op_j)} \quad (6)$$

若 α_j 是操作 op_j 其权重, 并满足 $\alpha_j > 0, \sum_{i=1}^N \alpha_j = 1$, 而 $c_i(\mathbf{g})$ 是联合查询的约束条件, 可由算法直接从查询中获得 (等式条件), 或者从抽象操作定义的范围获得 (不等式条件), 则查询过程的可行解向量 $\mathbf{g} = (op_1, op_2, \dots, op_N)$ 是下列多目标优化问题的解:

$$\begin{aligned} \max F(\mathbf{g}) &= \sum_{j=1}^N \alpha_j v_j(op_j) \\ \text{s. t. } c_i(\mathbf{g}) &\geq 0, \quad i=1, 2, \dots, K \\ \mathbf{g} &\geq 0 \end{aligned} \quad (7)$$

3.4 使用遗传算法的搜索策略

GA 本质上是一个通过群体的迭代来不断优化过程^[10]. 它简单、通用, 对问题的种类有很强的鲁棒性, 很适合解决 NP 问题. 与线性整数规划不同, GA 对优化问题的目标函数和约束条件的线性化不施加限制, 从而允许优化模型使用各种可能的 QoS 属性 (甚至定制).

为了让 GA 找到问题的一个解, 首先需要使用一个合适的基因组给问题编码. 在本文中, 基因组用一个整数数组来表示, 数组中的项目数为组成我们服务的抽象服务数, 数组中的每个元素依次包含一

个与抽象服务相匹配的具体服务数组项目索引. 图 2 给出了基因组是怎样产生的原理.

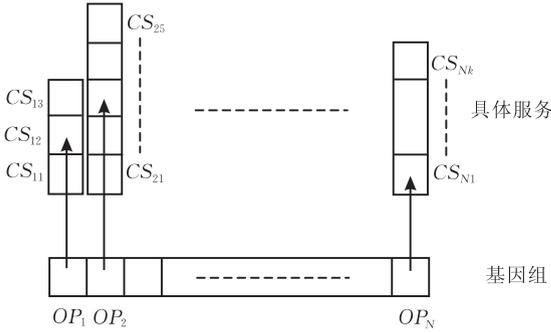


图 2 基因组编码

交叉(crossover)算子是标准的两点交叉, 而变异(mutation)算子随机地选择一个抽象服务(即基因组中一个位置), 并且随机地用另一个可利用具体服务取代对应的具体服务. 显然, 能从 GA 进化中获得仅有一个具体服务可利用的抽象服务.

选择(selection)算子采用经典的轮盘赌(roulette wheel)选择法. 其中, 若某个个体 i , 其适应度为 $Fitness(i)$, 则被选中的概率 P_i 为

$$p_i = Fitness(i) / \sum_{i=1}^{topsize} Fitness(i).$$

算法还在新一代个体产生之后采用最佳个体保存方法(elitist model), 即如果新一代群体中适应度最高的个体适应度小于老一代最佳个体, 而把老一代最佳个体保存入新一代.

接下来的问题是设计一个适应度(fitness)函数. 适应度函数需要使一些 QoS 属性(如可靠性)最大化, 而其它一些属性(例如费用)最小化. 当使用用户定义的领域特定的 QoS 属性时, 适应度函数规约留给 workflow 设计者. 另外, 适应度函数必须惩罚不满足约束条件的个体, 驱动进化有助于约束满足.

考虑到初始种群(设大小为 m)满足式(7)中约束条件, 结合问题所采用的编码和遗传算子, 满足约束条件 $\mathbf{g} \geq \mathbf{0}$ 是显然的. 为此, 可定义如下的一个约束满足的罚函数:

$$H(\mathbf{g}) = \sum_{i=1}^K c_i(\mathbf{g}) \times y_i \quad (8)$$

式(8)中系数 y_i 为

$$y_i = \begin{cases} 0, & c_i(\mathbf{g}) \geq 0 \\ 1, & c_i(\mathbf{g}) < 0 \end{cases}$$

进而得到, 包含一个罚函数的基因组 \mathbf{g} 的适应度函数如下:

$$Fit(\mathbf{g}) = F(\mathbf{g}) + rH(\mathbf{g}) \quad (9)$$

式(9)中 r 为罚函数尺度系数, $r > 0$.

最后, 有必要定义 GA 的优化准则. 根据离线性能函数(off-line performance):

$$PF(T) = \left[\sum_{t=1}^T f(*, t) \right] / T \quad (10)$$

其中 T 为当前计算中所出现的遗传代数, $f(*, t) = \max\{f(1, t), f(2, t), \dots, f(m, t)\}$, 即第 t 代中的最佳个体适应值. 该函数指标反映了群体中最佳个体适应值经平滑处理后的变化情况, 描述了个体的进化能力和 GA 的搜索能力, 并最终趋向一个收敛值, 所以可以用 $PF(T+1) - PF(T) < \epsilon$ 作为终止循环的条件之一, 文中取 $\epsilon = 1.0 \times 10^{-5}$. 为了防止群体规模太大, 搜索精度要求太高, 导致搜索时间过长, 本文同时采用设定最大代数的方法作为另一个终止循环条件.

3.5 动态适应度函数

定义在等式(9)中的适应函数对违反约束的个体包含了一个静态惩罚. 换句话说, 每一代的惩罚是同样的. 通常, 如果惩罚因子的权重 r 高, 那么违反约束的个体存在某种风险, 但却可能放弃了一个最为接近的好方案. 作为替代, 可以采用一个动态的惩罚, 即惩罚的权重随着代数而增加. 在遗传的早期世代, 还可能需要考虑一些个体违反约束. 经过若干世代后, 种群大体上应能满足约束, 进化余下的工作是如何改进适应度函数. 动态适应度函数的定义如下:

$$Fit(\mathbf{g}, t) = F(\mathbf{g}) + rH(\mathbf{g}) \times \frac{t}{tMax} \quad (11)$$

这里 t 是当前代数, $tMax$ 是最大遗传代数.

3.6 适应度函数的拉伸方法

遗传算法在运行早期个体差异较大, 当采用轮盘赌方式选择时, 后代产生个数与父个体适应度大小成正比, 因此在早期容易使个别好的个体的后代充斥整个种群, 造成早熟(premature); 在遗传算法后期, 适应度趋向一致, 优秀的个体在产生后代时, 优势不明显, 从而使整个种群进化停滞不前(stalling). 因此, 对适应度适当地拉伸(stretching)是必要的.

本文借鉴模拟退火(simulated annealing)^[10]思想, 提出如下适应度拉伸方法:

$$Fit = \frac{1}{\sqrt{\lambda}} e^{-\lambda(1-Fit(\mathbf{g}, t))}, \lambda > 0 \quad (12)$$

其中, 式中系数 λ 决定了复制的强制性, 其值越小, 复制的强制就越趋向于那些具有最大适应度的个体. 实验表明: 当 λ 取 0.95~1.05 时, 所取得的性能最好, 本文选取 $\lambda=1$.

4 原型系统的实现和仿真

4.1 原型系统的实现

为了验证所提的查询基础结构及其优化模型,我们从高端实现了一个服务查询引擎(service query engine),应用于我们开发的基于服务的企业资源计划协同集成框架 SEIFCW 中,完成对特定企业应用域的基本 Web 服务构件的组合查询和优化.如图 3 所示,SEIFCW 使用包含语义属性的 WSDL 描述企

业基本 Web 服务,使用 UDDI tModels 广播 SLA 中的 QoS 参数,并且在 UDDI 注册中心注册. UDDI 注册中心包含如下两类:(1) 基于 Internet 的全局 UDDI 企业注册表(UBR),实现公共 UDDI 注册存储的组织管理,例如企业信息门户(PL)、电子商务(EC)和领导查询系统(LQ)等;(2) 私有 UDDI 注册中心,企业内部的基本业务功能 Web 服务在这里注册,例如购销存管理(BM)、总账系统(FS)、人力资源管理(PM)、经营管理(RM)、基础定义(BD)和控制台(CP)等.

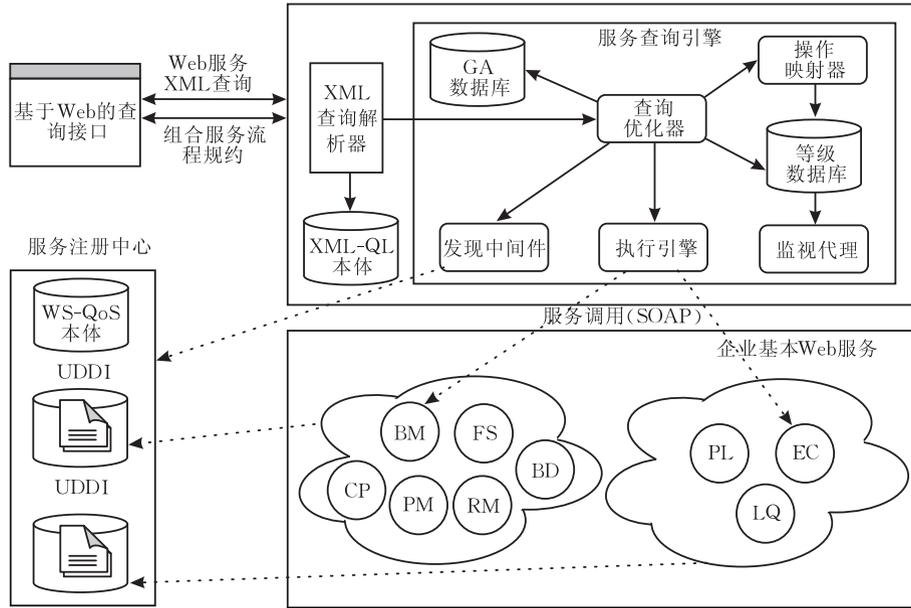


图 3 应用 SEIFCW 框架的 Web 服务查询基础结构

服务查询引擎(service query engine),作为在 SEIFCW 中实现本文所提查询方法的一个核心部分,包括如下组件:

(1) 查询优化器(query optimizer). 查询引擎的核心组件,它基于 QoS、服务等级、匹配度并调用 GA 算法库,以便确定最好的服务组合执行计划.

(2) 发现中间件(discovery middleware). 使用了一种基于多 Agent 和语义驱动的 Web 服务有效发现和搜索平台(MASWEM),可在适当配置后能自动访问 UDDI 注册中心,发现经过筛选处理后的企业基本 Web 服务构件的 WSDL 描述. 一旦发现中间件返回一个服务,查询引擎通过 SOAP 绑定存根,并使用 Apache SOAP API 调用它的操作.

(3) 操作映射器(operation mapper). 通过与发现中间件交互分析 WSDL 及其组成部分的 QoS 文档,决定服务执行计划中满足 QoS 属性约束的具体操作.

(4) 监视代理(monitring agent). 监控 Web 服

务调用,它的目标是根据已交付的 QoS 对其行为作出评价.

(5) 执行引擎(execution engine). 在优化器产生一个有效的服务执行计划后,通过 SOAP 调用实际的 Web 服务来行使组合服务执行计划.

Web 服务查询的处理过程是:用户通过可视化查询接口,首先使用一种从 XML 文档中抽取数据的查询语言规范 XQuery^[11]构造一个面向抽象服务关系库的联合查询 Q;然后 XML 查询解析器(XML query resolver)搜索抽象操作的库(XML-QL ontology)获得对应的抽象操作集;最后查询引擎根据这些抽象操作的 Category 和 Function 把它们与企业基本 Web 服务的各种操作(符合任一个匹配度)相匹配,并负责 SEIFCW 中的 Web 服务的正确性和最优化执行.

4.2 仿真和性能评价

本节通过实验分析所提模型的性能,包括观察适应度因子随 GA 代数的进化;比较各种适应度函

数(即具有静态和动态约束惩罚以及拉伸). GA 的主要参数取值如下:每个抽象操作的服务提供者数为 1~63,选择机制采用轮盘赌选择,保留 2 个最佳个体一直保持活跃整个世代,交叉概率 $P_c=0.7$,变异概率 $P_m=0.05$,最大迭代数 $tMAX=100$,种群大小 $pSize=50$,搜索精度 $\epsilon=1.0 \times 10^{-5}$,复制的强制性系数 $\lambda=1$,匹配度 $md=1$.

此外,在适应度函数中,对不同 QoS 属性赋予均等的权值,对某些案例场景的研究最终还要禁用一些属性.对每次实验,GA 被执行 50 次,报道的是平均值.标准偏差总是控制在平均值的 5% 之下.为了比较不同的适应度函数,这里提出的结论来自优化一个由包含 9 个截然不同抽象服务的 12 个原子操作调用构成的 workflow,其圈复杂度(cyclomatic complexity)等于 10.基因组的二进制编码长度是 72.

图 4 和图 5 分别从不同的角度,给出了 3 个不同适应度函数的查询服务的 QoS 参数总体演化过程.图 4 采用当前群体发现的最佳可行解的改善情况指标,即式(10)中的 $f(*,t)$ 来反映 GA 搜索到全局最优解的过程、速度和早熟等情况,这里考虑 GA 用于函数优化的目的是发现问题的全局最优解.图 5 使用式(10)的离线性能反映群体中最佳个体适应值经平滑处理后的变化情况,描述了个体的进化能力和 GA 的搜索能力.进化显示 3 个不同的适应

度函数都能使 GA 是找到满足约束的解,同时,使全局的 QoS 参数总体最优化.

对本案例的优化问题,动态适应度并不比静态适应度强,甚至适应度权重的不同刻度对优化也不提供帮助.除了动态适应度的启动快以及获得结果的最终进化不同,显著性水平(significance level) $\alpha=5\%$ 的 t -检验表明差别并不是很大.但通过对动态适应度作适当的拉伸,获得最优解的波动性、迭代代数和收敛时间等性能有明显提高.实验在不同规模 workflow、复杂性上重复做,基本上确认了上述结论.

5 相关工作

由于面向服务集成环境中应用之间的资源竞争,如何基于 QoS 约束快速地发现、选择并组合成最终满足用户需求的服务,已成为面向服务软件工程中的一个感兴趣话题^[1,6].动态服务发现和 QoS 感知组合的优化模型的约束处理方法主要来自运筹学或人工智能的搜索策略.事实上,在面向服务体系结构中,约束集是用来表达搜索或监控服务的功能性和非功能性属性的,且该服务可能根据一些 QoS 优化标准被选择.更多的挑战是组合服务的情况,它的结构被抽象流程描述,即包含抽象服务的流程,目标是从 QoS 观点动态地找到最好的具体服务组合.由于 Web 服务环境的动态特性,这种组合可能在执行过程中随时改变,因此,一个重要的需求是一个可行的并符合 SLA 的方案在短期内应该被发现.

文献[8]设计了一个用于语义 Web 服务的 QoS 描述的本体,并提供了一个 QoS 度量定义模型.文献[12]提出了一种有效的、动态的、QoS 感知的 Web 服务选择和监控方法,但没有解释优化问题怎样能被解决.文献[13]设计了一种基于服务质量的 Web 服务描述语言 QWSDL,并提出了一种前提条件是服务提供者发布的广告描述是真实可信的匹配模型.服务发现问题转化为请求描述与广告描述之间的匹配问题.文献[14]提出了一种服务质量计算模型,并通过增加 QoS 注册中心支持了基于 QoS 的 Web 服务选择.服务注册中心根据反馈信息对 QoS 排序,同时对 QoS 属性进行规范化处理.不过在对 QoS 进行规范化处理时没有考虑用户的个性化需求.文献[15]认为质量驱动的 Web 服务组合选择可归结为优化问题,并能使用有效的线性规划方法解决.文献[16]提出一种用于 QoS 感知的选择

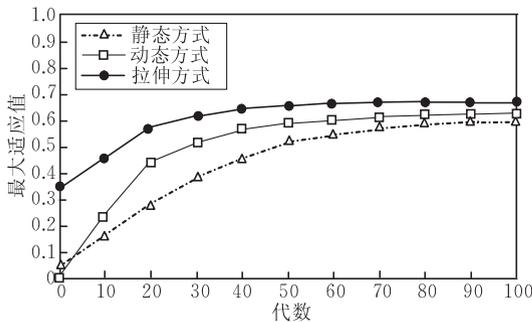


图 4 最大适应度值的进化过程

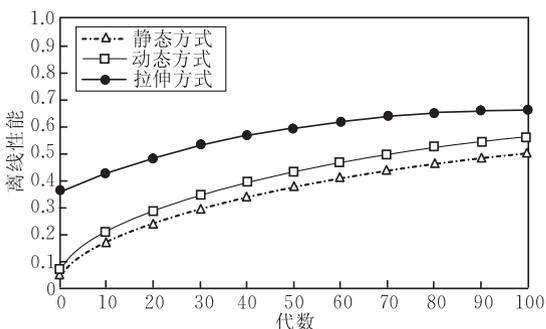


图 5 离线性能的进化过程

Web 服务的遗传算法. 与已有的研究工作相比, 本文提出在服务请求者使用传统的关系查找服务时提供 QoS 的期望值、权重及匹配度, 基于 GA 的多目标服务优化选择模型通过比较实际值与期望值的差别, 同时考虑权重和匹配度, 从而满足用户的个性化综合查询需求.

6 结 论

本文提出了一种 QoS 感知的领域 Web 服务综合查询基础结构与优化选择模型, 可方便地解决从多个提供类似功能的领域服务中发现、选择并集成最佳服务的问题. 查询基础结构允许用户在查询层使用传统的关系表达查询请求, 且把它们转换到一个组合的抽象操作调用, 然后使用各种匹配模式把抽象操作映射到实际的 Web 服务中的具体操作. 查询基础结构基于 Web 服务的领域、尺度和量制属性支持对 QoS 的描述, 并通过一个专用 Agent 负责对 Web 服务操作调用的监控, 进而能够对服务的 QoS 属性值进行动态评估, 真实地反映 Web 服务的服务质量. 同时在查找服务时通过引入权重和匹配度来实现用户的个性化选择, 选择最佳服务的方法简单可行, 提高了服务的查准率.

为了提高基础结构中 QoS 感知的 Web 服务选择的总体优化能力, 提出了一个基于相似性、QoS、匹配度、等级和可行性的多目标优化选择模型, 并给出了一个相应的 GA 算法. 该算法做到: (1) 采用恰当的编码方式, 以有效地减少编码空间和搜索空间; (2) 通过遗传策略的选择, 以使算法一定能收敛到最优解; (3) 实验决定复制的强制性系数、匹配度和变异等参数, 以使算法的性能最优; (4) 合理选择有关 QoS 属性的适应度函数, 以进一步提高算法性能. 实验表明我们提出的遗传算法有着不错的性能, 在令人满意的时间内收敛到最优解; 实验也表明了我们选择的适应度函数拉伸方法的合理性与正确性, 因为采用模拟退火思想的适应度函数的拉伸方法在最小的代数和最短的时间内找到了最优解.

下一步的工作是把所提的 Web 服务查询模型和选择优化算法进一步应用到一些实际的大规模面向服务系统开发中, 以便对其可靠性、适应性和性能作更深入的分析 and 改进.

参 考 文 献

- [1] Menasce D A. QoS issues in Web service. *Internet Computing*, 2002, 6(6): 72-75
- [2] Davis L, Gamble R, Hepner M, Kelkar M. Toward formalizing service integration glue code//*Proceedings of the 2005 IEEE International Conference on Services Computing*. Orlando, FL, USA, 2005, 1: 165-172
- [3] Jiang Zhe-Yuan, Han Jiang-Hong, Wang Zhao. Study on ERP information integration framework with cooperative work based on Web services. *Chinese Computer Engineering and Applications*, 2005, 41(6): 24-28(in Chinese)
(蒋哲远, 韩江洪, 王钊. 面向 Web 服务的 ERP 协同集成框架研究. *计算机工程与应用*, 2005, 41(6): 24-28)
- [4] Cardoso J. Quality of service and semantic composition of workflows [Ph. D. dissertation]. Athens, GA, USA: University of Georgia, 2002
- [5] Jin Hai, Chen Han-Hua, Lu Zhi-Peng, Ning Xiao-Ming. QoS optimizing model and solving for composite service in CGSP job manager. *Chinese Journal of Computers*, 2005, 28(4): 578-588(in Chinese)
(金海, 陈汉华, 吕志鹏, 宁小敏. CGSP 作业管理器合成服务的 QoS 优化模型及求解. *计算机学报*, 2005, 28(4): 578-588)
- [6] Zeng L Z, Benattallah B, Ngu Anne H H, Dumas M, Kalagnanam J, Chang H. QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering*, 2004, 30(5): 311-327
- [7] Gronmo R, Jaeger M C. Model-driven methodology for building QoS-optimised Web service compositions//*Proceedings of the 5th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 05)*. Athens, Greece, 2005: 68-82
- [8] Zhou C, Chia L T, Lee B S. DAML-QoS ontology for Web services//*Proceedings of the IEEE International Conference on Web Services (ICWS 2004)*. San Diego, CA, USA, 2004: 472-479
- [9] Huhns M N. Agents as Web services. *IEEE Internet Computing*, 2002, 6(4): 93-95
- [10] Goldberg D E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Publishing Company, 1989
- [11] Robie J. XML processing and data integration with XQuery. *Internet Computing*, 2007, 11(4): 62-67
- [12] Tian M, Gramm A, Ritter H, Schiller J. Efficient selection and monitoring of QoS-aware Web services with the WS-QoS framework//*Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004)*. Beijing, China, 2004: 152-158
- [13] Hu Jian-Qiang, Zou Peng, Wang Huai-Min, Zhou Bin. Research on Web service description language QWSDL and service matching model. *Chinese Journal of Computers*, 2005, 28(4): 505-513(in Chinese)
(胡建强, 邹鹏, 王怀民, 周斌. Web 服务描述语言 QWSDL 和服务匹配模型研究. *计算机学报*, 2005, 28(4): 505-513)

- [14] Liu Y T, Ngu Anne H H, Zeng L Z. QoS computation and policing in dynamic Web service selection//Proceedings of the 13th International World Wide Web Conference (WWW 2004). New York, NY, USA, 2004: 798-805
- [15] Zeng L Z, Benatallah B, Dumas M, Kalagnanam J, Sheng Q Z. Quality driven Web services composition//Proceedings of the 12th International Conference on World Wide Web. Bu-

dapest, Hungary, 2003: 411-421

- [16] Zhang Cheng-Wen, Su Sen, Chen Jun-Liang. Genetic algorithm on Web services selection supporting QoS. Chinese Journal of Computers, 2006, 29(7): 1029-1037(in Chinese) (张成文, 苏森, 陈俊亮. 基于遗传算法的 QoS 感知的 Web 服务选择. 计算机学报, 2006, 29(7): 1029-1037)



JINAG Zhe-Yuan, born in 1966, Ph. D., associate professor. His research interests include service oriented software engineering, software architecture, and software engineering environment.

HAN Jiang-Hong, born in 1954, professor, Ph. D. supervisor. His research interests include intelligent control technology and distributed system.

WANG Zhao, born in 1958, associate professor. His current research interests focus on e-commerce and software engineering.

Background

The group's research interests are currently focused on the service-oriented software development approaches, modeling notation, and the integration techniques that rely on the specific domain ERP systems and Grid Web services computing resources across the Internet. They have presented a service-oriented ERP integration framework with Cooperative Work (SEIFCW) from the views of the distributed computing and XML technique, a novel XML-based Architecture Description Language named S-XADL for describing service-oriented software architectures, a Multi Agent-based and Semantics-driven Web Services Middleware (MASWEM) platform using existing standards instead of proposing other standards, and so on. One of the critical challenges in reusing and integrating existing Web service components is the efficient query, selection and composition of potentially relevant Web service components based on functional, operational and QoS requirements. This paper proposes a new QoS-aware query infrastructure and dynamic selection of composite Web Services, which takes account of both the QoS properties and interface parameters matching degree. When doing

the selection, the authors consider that the task is more or less a multistage decision-making process. Motivated by genetic algorithm's powerful computation ability, a genetic algorithm is proposed in this paper for such task. In order to make this algorithm more adaptable for multistage decision-making problem, they use various fitness functions varying in terms of static penalty, dynamic penalty, and stretching for the optimization performance of the genetic algorithm. The proposed approach has been applied to a service query engine system for SEIFCW, which consequently shows its applicability, feasibility and efficiency. This work is partly supported by the National High Technology Research and Development Program (863 Program) of China under grant No. 2002AA415280, the National Research Foundation for the Doctoral Program of Higher Education of China under grant No. 20050359004, the Program for New Century Excellent Talents in University of the Ministry of Education of China under Grant No. NCET-04-0562, and the Science Research and Development Foundation of Hefei University of Technology under grant No. 2007GDBJ012.