动态 Web Services 的约束满足优化技术研究*)

刘 壮 郭荷清 韩 涛 李 冬
(华南理工大学计算机科学与工程学院 广州 510640)

摘 要 Web Services 合成是 Web Services 技术的重要方面,能够按要求提供选择新的服务。本文首先提出了 Web Services 服务约束的分类描述,进而分析了 Web Services 服务合成中如何按照服务约束指导服务选择,进而选择合适的服务集来满足客户的要求。该方法把动态 Web Services 合成转化为约束满足问题。

关键词 约束满足,优化,动态 Web Services

Towards Constraint Satisfaction and Optimization for Dynamic Web Service Composition

LIU Zhuang GUO He-Qing HAN Tao LI Dong
(Dept. of Computer Science and Engineering, South China University of Technology Guangzhou 510640)

Abstract Web Services composition is a crucial aspect of Web services technology, which gives us the opportunity for selecting new services and best suit for the customer's requirements. In this paper, We firstly propose the category descriptions of constraints in Web Services and analysis how to guide the selection of the services as constraints in Web Services Composition, thus choose an optimal set of services to satisfy the customers requirements. Meanwhile, this approach reduces the dynamic composition of Web services to a constraint satisfaction problem.

Keywords Constraint satisfaction, Optimization, Dynamic Web services

Web Services 模型定义了 3 个角色:服务提供者(Service Provider)、服务请求者(Service Requestor)和服务代理/注册(Service Broker/Registry),也定义了角色间相互作用的操作:发布(Publish)、发现(Find)和绑定(Bind)。服务提供者向服务代理发布服务,服务请求可以通过代理发现所需服务,并和服务提供者建立联系。服务描述记录在 WSDL(Web Services Description Language)中,提供了服务接口(Interface)和服务端口(Port),然而,却没有工具能够明确描述与 Web Services 操作相关的属性和值。

约束规范(Constraint Specifcation)作为描述与 Web Service 和服务操作相关的各种属性类型和有意义的值的规范,允许服务请求者根据与服务操作属性相关的值,指定服务请求约束。通过服务请求约束与注册服务请求约束的匹配,服务注册/代理能够选择最适合的服务。同时,服务约束能够提高 Web 服务发现匹配的速度,也能够提供最适合用户需求的合成服务。而假若没有服务约束,则将产生假阳性服务,或服务的操作产生不了预期的结果[1]。

目前,对 Web Services 的自动合成,已大量采用了语义 Web,因为语义 Web 能够提供服务本体更多的涵义,这样 Web Service 需要语义注解和高水平的处理过程描述。这些描述可以是目标的、应用逻辑的、等级计划的[2~5]。然而却没能够提供一个可理解的适合服务选择的框架模型。

约束求解与优化技术分属不同的领域,前者属于计算机科学及人工智能领域,而后者则属于数学中逻辑学范畴,由于二者在求解技术上的互补性,促进了二者在实际问题中的融合。本文试图从服务约束满足优化技术上进行研究,并作为对该领域研究的补充。

1 相关知识

1.1 动态 Web Services 合成

Web Services 是模块化的自描述、自包含的应用服务。Web Services 合成指异构 Web Services 的协作处理过程,可以是静态的,也可以是动态的。在静态合成时,服务事前在服务设计时已定义好。产生后,其接口可以用 WSDL 来描述,并在 UDDI 中发布具体服务信息供其他请求者发现使用。动态合成则涉及对运行时注册的服务搜索。

从商务的角度看,Web Services 合成有几个优点:(1)合成服务允许组织只需提供开发应用最小的工作量,就能够确保快速市场响应;(2)基于 Web Services 的应用开发能够减少商业风险,因为重用技术避免了错误的发生。最后,合成 Web 服务能够减少开发应用所需的技术和效果要求。

1.2 约束满足问题[6,7]

约束满足问题(Constraints Satisfaction Problem,简称 CSP)是一个给变量分配与之相一致的值的问题。CSP 基本组成元素包括变量的域 D、一个变量的有限集和一个约束集。每个变量和一系列可能的值相关联。约束指定义在这些变量子集的关系,并表明这些值的有效组合。对约束满足问题的一个有效解决方案是给域中每个变量分配一个值,并使所有的约束都满足。

约束满足涉及给问题变量主题寻找一个值,以实现对可接受的值组合的约束。CSP 模型被广泛用来提供解决各种人工智能相关的问题,如日程安排、计划或计算机辅助设计,并提供基本的工具应用在真理维护、专家系统或约束逻辑编程中。同时,CSP 也可以提供简单的、自然的,然而却有效的知识表征框架和各种算法来解决各种请求类型。

*)国家 973 高技术研究发展计划基金资助项目(G20000263)。刘 壮 博士研究生,主要研究方向为信息系统的集成与安全。

根据约束满足问题中变量的域 D 的不同，约束满足问题可以分为：布尔约束满足问题、有限约束满足问题和混合约束问题。

1.2.1 布尔约束满足问题

布尔约束满足问题要求变量只能在 0 或 1 上取值，其约束实际上就是一组命题逻辑公式，所谓命题逻辑公式是布尔变量与逻辑连接符按照如下的规则形成的组合体：

①布尔变量是公式；

②如果 φ 是公式，则 $\neg\varphi$ 也是公式；

③如果 φ 和 ϕ 是公式，则 $\varphi * \phi$ 也是公式；

④只有上面四条规则生成的表达式是公式。

1.2.2 有限约束满足问题

有限约束满足问题就是变量只能在有限域上取值，采用列出所有满足该条件的变量取值组合的形式。

1.2.3 混合约束问题

混合约束问题中变量可以在多个域中取值，如变量可以取布尔值，也可以在有限数值域及无限数值域上取值。其实质上是对布尔满足问题的一种扩展。

定义 1(数值约束) 形如 Exp1 rop Exp2 的约束为数值约束，其中 Exp1 与 Exp2 为数学表达式，rop 为数学中的关系操作符，包括=，≠，≤，≥，<，>。

定义 2(混合约束条件) 由布尔变量及数值约束与逻辑连接符组成，按照与命题逻辑公式同样的规则形成组合体。

总的来说，用于求解约束满足问题的方法包括完备算法与不完备算法。所谓完备算法是指能够完全确定某约束满足问题是有解的还是无解的算法；而不完备算法则在未找到解时，不能确定该约束满足条件无解[8]。

1.3 优化问题极其求解方法

优化问题是数学领域的重要分支，研究如何在众多方案中找到最优的一个。优化问题可以用下式描绘：min f(x)

s. t. $g_i(x) \leqslant 0,\ i=1,2,\cdots,m$

$h_j(x)=0,\ j=1,2,\cdots,n$

其中，$f(x)$称为目标函数，$g_i(x)$和 $h_j(x)$称为约束函数。若 $f(x)$，$g_i(x)$，$h_j(x)$都是 x 的线性函数的优化问题，称为线性优化问题；否则称为非线性优化问题。如果要求得到的解为整数或者部分为整数，那么该优化问题就是整数规划问题。

求解线性优化问题最常用的算法是单纯形法，它的提出标志着优化问题成为一个独立的学科[9]。而求解非线性优化问题的主要算法包括牛顿法、最小二乘法、乘子法等，这些算法本质上是迭代法。目前，常采用基于区间分析等技术的全局优化算法，其本质上是分枝定界法(branch-and-bound)，即初始为每一个变量赋一较大区间，然后对各区间不断进行检验、折半，直到最后每一变量的区间宽度小于一定的值。另外一大类用于求解优化问题的算法为随机算法，包括随机搜索(random search)、禁忌搜索(tabu search)、模拟退火(simulated annealing)、演化计算(evolutionary computation)等[10]。

1.4 约束求解与优化技术的结合

有限约束求解与整数优化规划求解技术从不同的角度对同一问题进行了处理。二者的结合有共同之处，同时又是互补的。二者的共同之处在于，求解问题所采用的方法本质上都是搜索法；而互补之处在于，约束求解能够剔除不可行空间，整数优化规划能够利用松弛问题缩小目标函数的范围。二者的有效结合，实质上是约束求解与优化技术能够求解同一问题的不同子问题[11,12]。本文所使用的 LINDO 求解器，实现了许多种约束优化算法，能够有效地实现约束求解与优化技术的结合。

2 Web Services 服务约束描述[13]

Web Services 目标的效果描述表达了用户需要达到的功能目标，而对用户的非功能目标(如：成本、时间、安全等要求)，则无法进行描述。执行计划对用户非功能目标的满足程度决定了用户的满意度。在 Web Service 中，有 3 类约束：

(1)QoS 约束：本文通过使用约束优化模块，实现用户目标的执行计划 QoS 属性的满意度，它描述了用户对服务质量的要求。

(2)安全约束：定义用户的隐私策略，所能提供的安全能力以及对服务提供者的安全要求。在某种意义上，执行计划的服务质量也可以包含安全约束，但是考虑到安全约束的复杂性，将其独立为一种约束。安全约束有 3 种类型：

- SecurityCapability：表示用户所能提供的安全机制；
- SecurityRequirement：表示用户要求的实现的服务必须具备的安全机制；
- PrivacyConstraint：表示实现服务所需的用户隐私信息的限制。

(3)服务过滤约束：用于对实现目标的相关服务进行过滤，主要在服务发现过程中使用。该约束由类 ServiceFilterConstraint 描述，主要在规划期用来选择可以实现目标的服务。这些非功能的服务描述信息可以由 OWL-S 中的 Service Profile 提供。

3 动态 Web Service 合成架构

动态 Web Services 合成架构用服务模板(Service Template，简称 ST)提供必要服务的功能性。该架构的主要组件是：OWL-S/UDDI 注册、发现引擎(discovery engine)和约束优化器(constraint optimizer)，如图 1 所示。

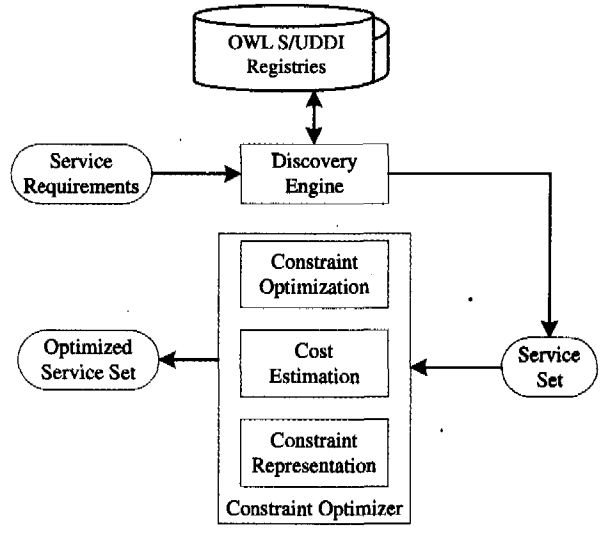

图 1 动态 Web Services 合成架构

3.1 OWL-S/UDDI 注册

服务注册是服务描述的可搜索注册，服务提供者发布服务描述，而服务请求者可发现服务并获得绑定信息。OWL-S/UDDI 注册能够减少搜索时间，增加服务发现等操作的精确性。

3.2 发现引擎

发现引擎是 OWL-S/UDDI 注册上的接口，能够提供语义发布和发现。发现引擎使用类包含关系把查询中指定的本体概念和发布在注册中的本体概念进行比较。通过比较在相关服务发现中概念的特性、匹配集的势和数据类型公共父节点的距离，实现公共度量单位。按照匹配的程度，对由用户/工具返回的结果进行分类。其他的一些特性，如可信性、服务质量(QoS)也可以用来对返回的结果进行分类。发现引擎中，发现方法主要是基于语义描述和服务提供者的服务约束，其中，查询模板被用来建立指定必要服务功能特性的查询。查询模板包括服务的细节，如操作名、操作行为(功能语义)、输入/输出名和语义类型，异常，前/后条件、域名和定位等。

3.3 约束优化器

约束优化器能够从来自发现引擎的候选服务集中，动态选择理想服务。

约束优化模块基于商务技术和处理过程约束，产生最理想的服务集。约束优化器能够确保发现的服务满足客户要求，并按照要求优化服务集。

· 约束表征模块(Constraint representation module)：类、属性、实例和关系的集合。商务约束可以是任何因子和规则，能够影响到给定处理过程的服务选择。

· 成本估计模块(Cost Estimation)查询存储在约束表征模块里的信息，按照能够影响对动态合成处理的服务选择的各种因素，进行成本评估。如供应链系统涉及到成本、时间、适应性和可信性等各种因素的成本评估。

成本估计模块查询约束特征并基于查询结果分配一个值给某一处理过程，使得查询服务本体时，可获得提供者关系状态。如 ss 表示提供者状态(SupplierStatus)，s 表示提供者(Supplier)，则成本函数在 ss 倾向某一合作伙伴时，supplierStatus(ss)=1；当 ss 是另一服务提供者时，supplierStatus(ss)=0.5；否则为 0。

同样，对两个部分适应提供者而言，本体查询如图 2 所示。

```
Query("type_x works type_y?")
Query Pattern:{(works ? x ? y)(type ? x Type_x)(type
  ? y Type_y)}
Must_Bind Variables List:(? x, ? y)
Answer Pattern:{(works ? x ? y)}
Answer:("type_x works type_y")
Answer Pattern Instance:{(works "Type_x" "Type_y")}
```

图 2 合成本体服务查询

基于查询结果的适应成本函数为：

$$\text{Compatibility}(type\_x, type\_y)=\begin{cases}1 & \text{当 type\_x 与 type\_y 部分相适应时}\\ 0 & \text{当 type\_x 与 type\_y 不相适应}\\ 0.5 & \text{当 type\_x 或 type\_y 没有出现在列表中}\end{cases}$$

处理过程约束中，用户需对 QoS 参数，指定聚集操作(aggregation operator)，QoS(p)=(T(p),C(p), R(p),A(p), DS1(p),DS2(p),…,DSn(p))，其中 T(p)为全部 Web 处理过程的执行时间，C(p)为调用处理过程中所有服务的代价，R(p)为处理过程中所有服务的累积可靠性，A(p)为处理过程中所有服务的累积可用性，DSi(p)为特定域 QoS 参数的累计积分。

每个度量规范包括一个 5-元组集 QoSi(p)={name, comparison()p, value, unit, aggregation()p}，其中 name 指 QoS 参数的名字，value 指数值，comparison()p 指比较操作符，unit 指测量单位，aggregation()p 指累积操作符。对大多度量标准而言，QoS 处理过程可以通过累积操作的求和、乘积、最大值或最小值计算出来。

然而，一些情况下，用户需要对累积操作，定义用户函数。在处理过程的许多领域内，需为每个领域进行约束描述，处理过程 QoS 也通过所有范围的累积操作计算出来。如对下列产品订购的不同商店领域的 QoS(scope)处理过程为：

序号	产品	数量
1	CD	1000
2	DCD	900
3	AV	1200

QoS(scope1)=⟨{CD},{DVD},{AV},{'cost','≤',3000,'dollars','Σ'},
{'supplytime','≤','7','days','Max'},
{'partnerStatus','≤','1','Max'},
{'compatibility','≥','2','F1'},
{'discoveryScore','≥','1','Min'}⟩

QoS(scope2)=⟨{CD},{DVD},{AV},{'cost','≤','4000','dollars','Σ'},
{'supplytime','≤','7','days','Max'},
{'partnerStatus','≤','1','Max'},
{'compatibility','≥','2','F1'},
{'discoveryScore','≥','1','Min'}⟩

其中，F1 是用户定义函数，通过调用适应性函数 Compatibility(ss_i,ss_j)($i\neq j$)计算领域兼容性。

· 约束优化(Constraint Optimation)：用户指定的约束存储在服务模板中。服务提供者可以指定服务操作，实现 QoS 或服务约束。优化模块通过匹配来自 UDDI 的服务模板或调用服务提供者指定的操作，重获服务约束。约束优化器能够根据目标功能，确保已发现的服务满足客户的约束，并优化服务集。这是服务执行前的最后一个阶段。成本评估模块计算实现所有 QoS 参数的候选服务的成本评估。优化目标函数来自用户定义的服务模板，是参数的线性组合。在 LINDO 线性编程解析器的作用下，把过程约束直接转化为对整型线性程序解析器的约束。

LINDO 包括一系列内置的解析器，如线性解析器(Linear Solvers)、最初和双重单一解析器(Primal and Dual Simplex Solvers)、障碍解析器(Barrier Solver)、整型解析器(Integer Solver)、非线性解析器(Nonlinear Solvers)、一般非线性解析器(General Nonlinear Solver)、全局解析器(Global Solver)、多起点解析器(Multistart Solver)、二次解析器(Quadratic Solver)和二次曲线解析器(Conic Solver)，这些解析器能够处理许多问题，通过提供对算法和解析参数的控制，允许用户对单个应用定制解析策略，从而获得最佳控制和运行速度，实现约束满足性限制[14]。

如：处理过程实例中，S_i 是注册中的第 i 个服务，O_{ij} 是 S_i 的第 j 个操作，ST_k 是第 k 个服务模板，OT_{kl} 是 ST_k 中的第 1 操作。对服务模板 k，其约束为：

$$x_{kli}=\begin{cases}1 & \text{当服务 } S_i \text{ 中的操作 } O_{ij} \text{ 与服务模板}\\ & \text{STK 中的操作 } OT_{kl} \text{ 匹配时}\\ 0 & \text{否则}\end{cases}$$

$$\min(w1 * \sum_{kli} c_{ij} x_{kli} + w2 * \sum_{kli} p_{ij} x_{kli})$$

遵从 $\sum_{kli} x_{kli} = 3$ 和 $\sum_{i} x_{kli} = 1$

其中 c_{ij} 和 p_{ij} 分别是服务 i 中操作 j 上的代价(cost)和同伴优先性(partner preference)。w1 和 w2 是处理过程设计中，成本和优先选择的权重。

优化目标函数是参数的线性组合，能够从用户定义的服务模板中提取出来。

这些约束输入到 LINDO 整型线性编程解析器时，将产生大量等级为最优到接近最优解决方案的可行集。可行集的分类是基于目标函数的值。每个约束的值，如时间(time)、成本(cost)和同伴优先(partner preference)也提供给可行集。这样处理过程设计者将可以选择可行集，并提供给运行模块，再由运行模块提供所需的服务。

3 实例

图 3 是一个基于约束的订购处理过程实例。

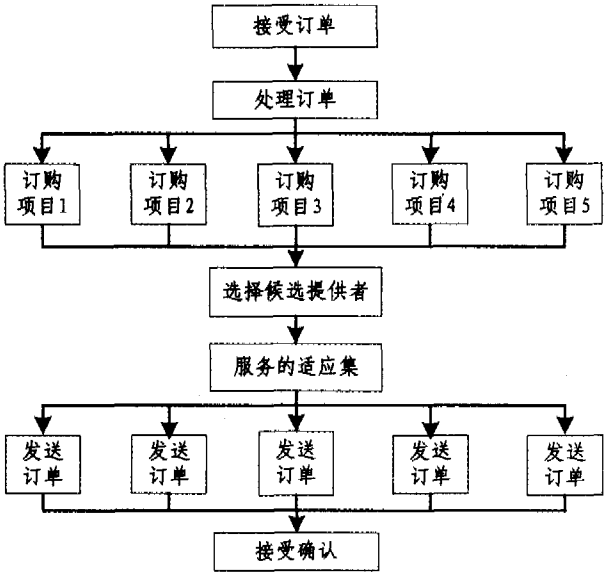

图 3 基于约束的订购过程

该例从顾客接收订单开始，订单处理后就选择潜在的服务提供者，潜在服务提供者存在为数量配额而相互作用的可能性。取得数量配额后，基于约束将产生最优服务提供者。

如对服务提供者 1 的约束为：

$$\begin{cases} x_j^i = 1 & \text{若候选服务 } ss_j^i \text{ 被选择时} \\ 0 & \text{否则} \end{cases}$$

$$\sum_i \sum_j \cos t(x_j^i) \leqslant \$\ 3000$$

$$\sum_i \sum_j partner(x_j^i) \leqslant P$$

$$\forall (i,j) \text{supplytime}(x_j^i) \leqslant 7days$$

$$\forall (i,j) discovery(x_j^i) \geqslant D$$

$$\sum_i \sum_j x_j^i = 3, \sum_i x_j^i = 1$$

如对服务提供者 2 的约束为：

$$\begin{cases} x_j^i = 1 & \text{若候选服务 } ss_j^i \text{ 被选择时} \\ 0 & \text{否则} \end{cases}$$

$$\sum_i \sum_j \cos t(x_j^i) \leqslant \$\ 3000$$

$$\sum_i \sum_j partner(x_j^i) \leqslant P$$

$$\forall (i,j) \text{supplytime}(x_j^i) \leqslant 7days$$

$$\forall (i,j) discovery(x_j^i) \geqslant D$$

$$\sum_i \sum_j x_j^i = 3, \sum_i x_j^i = 1$$

这些约束输入到 LINDO 整型线性编程解析器时，将产生从最适合集到一些可行集的分级，从而根据该分级，动态地选择合适的 Web 服务组合。

结论 本文提出了动态 Web 服务合成的约束满足方法，也提出了动态 Web 服务合成约束满足性方法的架构模型。

约束优化模块，使用整型线性编程解析器能够实现基于过程和商务约束的处理过程优化。

成本评估模块查询存储在约束表征模块里的信息，能够影响对处理过程的服务选择。

参 考 文 献

1 Stanley S D, Su Y W. Constraint Specification and Processing in Web Services Publication and Discovery. In: Proceedings of the IEEE International Conference on Web Services (ICWS'04), IEEE Computer Society, 2004

2 Bussler C, Fensel D, Maedche A. A Conceptual Architecture for Semantic Web Enabled Web Services SIGMOD Record. Special Issue Semantic Web and Databases, 2001

3 McIlraith S, Son T. Adapting Golog for Composition of Semantic Web Services. In: Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002), April 2002

4 Benatallah B, Sheng Q Z, Dumas M. The Self-Serv Environment for Web Services Composition. IEEE Internet Computing, 2003, 7(1): 40～48

5 Wu D, Parsia B, Sirin E, et al. Automating DAML-S web services composition using SHOP2. In: Proceedings of 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, 2003

6 Channa N, Li Shanping, Shaikh A W. Constraint Satisfaction in Dynamic Web Service Composition. In: Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA'05), IEEE Computer Society, 2005

7 Dubois D, Fargier H, Prade H. Possibility theory in constraint satisfaction problems: handling priority, preference and uncertainty. Applied Intelligence, 1996,6 (1): 287～309

8 Dechter R. Constraint Processing. San Francisco: Morgan Kaufmann Publisher, 2003

9 Dantzig G. Linear Programming and Extensions. New Jersey: Princeton University Press, 1963

10 Mendivil F, Shonkwiler R. Optimization by stochastic methods: [Technical Report]. GA 30332. Institute of Technology Atlanta, Georgia, 1999

11 Hentenryck P. Constraint and integer programming in OPL. INFORMS Journal on Computing, 2002,14(4): 345～372

12 Hooker J N, Ottosson G. A scheme for unifying optimization and constraint satisfaction methods. Knowledge Engineering Review, 2000, 15(1):11～30

13 Aggarwal R, Verma K, Miller J A, et al. Constraint Driven Web Service Composition in METEOR-S. IEEE SCC, 2004. 23～30

14 LINDO. API version4. 0-WIN32, Lindo System Inc. Http://lindo.com