软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2015,26(8):1886−1906 [doi: 10.13328/j.cnki.jos.004598]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 一种高效的基于服务功能规约的服务选择方法*

白 琳[1,2,3], 叶 丹[1,2], 魏 峻[1,2], 黄 涛[1,2]

[1](中国科学院 软件研究所 软件工程技术中心,北京 100190)

[2](计算机科学国家重点实验室(中国科学院 软件研究所),北京 100190)

[3](中国科学院大学,北京 100190)

通讯作者: 白琳, E-mail: bailin@otcaix.iscas.ac.cn

**摘 要**: 服务因其灵活的应用机制,逐渐成为软件开发过程中的主要载体.面对网络中涌现出的大量功能相同或相似而 QoS(quality of service)不同的服务群体,如何快速、准确地定位到所需要的服务个体,仍是一项十分具有挑战性的工作.基于服务对功能的封装特性,提出一种基于服务功能规约的服务选择方法,将功能相关的抽象服务规约为一个粒度更大的服务级的抽象服务;然后,针对规约后的大粒度抽象服务完成服务发现和组合优化的过程.由于规约操作能够有效减少参与服务组合优化的抽象服务的个数以及相应的候选服务的个数,从而使算法的执行效率得到有效提升.实验结果表明:该算法与传统的启发式算法相比执行效率更高,并且对服务组合规模及候选服务规模表现出更好的扩展性.

**关键词**: 服务选择;QoS(quality of service)感知;服务功能规约;组合优化;遗传算法;非冗余集合覆盖

**中图法分类号**: TP311

中文引用格式: 白琳,叶丹,魏峻,黄涛.一种高效的基于服务功能规约的服务选择方法.软件学报,2015,26(8):1886−1906. http://www.jos.org.cn/1000-9825/4598.htm

英文引用格式: Bai L, Ye D, Wei J, Huang T. Efficient service selection approach based on functionality folding. Ruan Jian Xue Bao/Journal of Software, 2015,26(8):1886−1906 (in Chinese). http://www.jos.org.cn/1000-9825/4598.htm

## Efficient Service Selection Approach Based on Functionality Folding

BAI Lin[1,2,3], YE Dan[1,2], WEI Jun[1,2], HUANG Tao[1,2]

[1](Technology Center of Software Engineering, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science (Institute of Software, The Chinese Academy of Sciences), Beijing 100190, China)

[3](University of Chinese Academy of Sciences, Beijing 100190, China)

**Abstract**: Service is becoming the main carrier of the software development because of its flexible application mechanism. Since the emergence of a large number of services with the same or similar functionalities but different QoS (quality of service), how to quickly and accurately locate the right services user need remains a very challenging task. In this paper, a service selection approach based on the folding of service functionality is proposed. Abstract services with associated functions are folded into a coarse-grained service-level abstract service, based on which the service discovery and combinatorial optimization are performed. The efficiency of the algorithm is improved dramatically because of the reduced number of both abstract services and candidate services. Experimental results show that the presented approach is more efficient than other traditional heuristic algorithms, and exhibits better scalability on the scale of abstract services and candidate services.

**Key words**: service selection; QoS-aware; folding of service functionality; combinatorial optimization; genetic algorithm; non redundant set covering

* 基金项目: 国家自然科学基金(61170074, 61173005); 国家重点基础研究发展计划(973)(2009CB320704); 国家高技术研究发展计划(863)(2012AA011204)

收稿时间: 2013-06-21; 修改时间: 2014-01-21; 定稿时间: 2014-03-28

面向服务软件开发(service oriented software development,简称 SOSD)以服务作为编程范型的基本元素管理软件架构和软件开发过程[1],按需对网络中已有的服务资源进行动态组合,形成满足应用需求的增值服务,成为当前一种主流的软件开发方法.国内外许多知名企业和机构都纷纷推出了各自的面向服务解决方案[2−4]以及面向服务应用开发框架[5−8].

服务作为面向服务软件开发的基本要素,近年来得到了迅速的发展和增长.丰富的服务资源给服务发现和选择带来巨大的挑战.针对服务选择问题,工业界和学术界都展开了积极的探索和研究,并先后提出一系列行之有效的理论和方法[9−13].这些方法着眼于服务的功能属性,利用语义匹配方法选择出满足用户功能需求的服务.然而,随着服务计算技术的发展以及服务组合应用的不断成熟与完善,用户对组合应用的需求不再仅仅停留在功能层面,而是更关注于其非功能属性的表现,如可用性、可靠性、价格等.与此同时,随着服务资源的日益丰富,网络中不可避免地涌现出大量功能相同但 QoS(quality of service)不同的服务资源.特别是一些大的服务提供商,在不同地理位置提供全球性的服务,往往由于基础设施的差异使功能相同的服务表现出不同的 QoS.正是由于这些功能相同但 QoS 不同的服务的存在,导致产生大量功能相同但全局 QoS 不同的组合服务.如何从中选择满足用户 QoS 需求的服务组合方案,成为面向服务软件开发中服务选择面临的一个关键问题.

QoS 感知的服务组合优化问题已被证明是 NP 难问题[14,15].传统的组合优化算法,如穷举法、分支定界法等,对于服务组合的规模及候选服务的规模非常敏感,其计算复杂度随服务组合规模及候选服务规模的增加呈现出指数级的增长.相对而言,具有多项式时间复杂度的启发式算法由于其良好的计算性能和扩展性,近年来得到了广泛的关注和应用.然而,当面对实际应用需求,现有的启发式算法仍存在较大的提升空间.如何进一步提高组合优化算法在服务选择过程中的执行效率,已成为制约面向服务软件开发发展的关键问题.针对这一问题,本文提出一种基于服务功能规约的服务选择方法.该方法首先对用户描述的服务需求按照功能相关性进行合并,形成粒度更大的服务需求;然后,针对大粒度的服务需求,在与其功能匹配的候选服务中选择 QoS 能够满足用户需求的服务进行绑定,最终得到一个或一组能够同时满足用户功能及非功能需求的组合服务.

本文的贡献主要体现在以下两个方面:

(1) 提出了一种基于服务功能规约的服务选择方法.将用户对服务的功能需求按照功能的相关性合并为更大粒度的服务需求,有效减小服务组合的规模;在执行服务组合优化时,只针对合并后的大粒度的服务需求进行匹配和绑定,有效减小候选服务规模,提高服务组合优化的效率.

(2) 设计了基于遗传算法的组合优化算法,不仅在执行效率方面比传统的组合优化算法有显著提高,而且在对服务组合规模及候选服务规模等参数的扩展性方面也有所提高.

本文第 1 节介绍 QoS 感知的动态服务选择,重点分析其中存在的效率问题.第 2 节介绍本文所提出的基于服务功能规约的服务选择方法.第 3 节通过一组实验分析本文方法的有效性.第 4 节与相关工作进行对比分析.第 5 节总结全文并指出下一步的工作计划.

## 1 问题分析

### 1.1 QoS感知的动态服务选择

服务选择是面向服务软件开发过程中的一项关键内容,为了提高应用灵活性,使其能够快速响应需求变化,面向服务的软件开发在设计阶段往往并不指定具体服务,而只描述应用所需要的功能.不同的用户对应用会有不同的 QoS 需求,因而具体服务的绑定留待运行时进行.我们将这种考虑用户全局 QoS 需求并在运行时动态绑定具体服务的操作称为 QoS 感知的动态服务选择,其过程如图 1 所示.

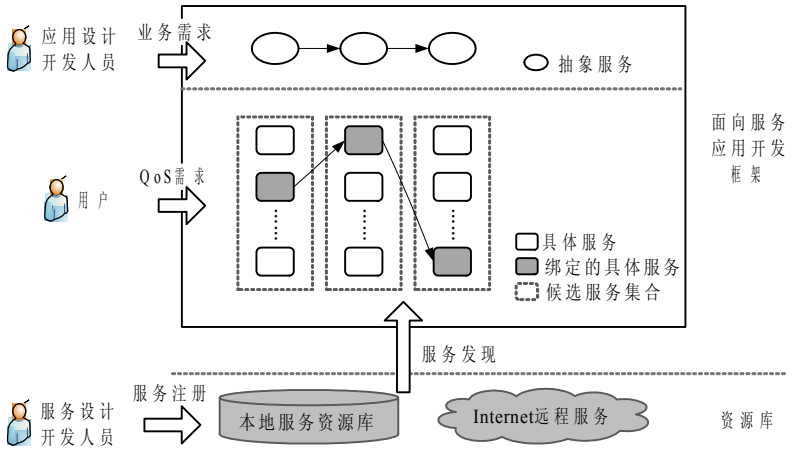



Fig.1 QoS-Awared dynamic service selection

图 1 QoS 感知的动态服务选择

QoS 感知的动态服务选择实质上是将服务选择分两个阶段完成:

第 1 阶段为服务发现阶段.

在此阶段,应用开发人员对应用的业务需求进行抽象,并刻画为抽象服务.本文中,抽象服务是对具备某一类特定功能的服务的抽象和概括.这些服务具有相同或相似的功能,但却表现出不同的 QoS.应用开发框架根据抽象服务所描述的功能需求,采用基于语法[9,10]或语义[11−13]的匹配算法,从服务资源库中选出与之功能匹配的具体服务作为该抽象服务的候选服务.例如,一个机票订购的应用设计了“机票查询”和“机票购买支付”两个抽象服务,服务发现即是查找能够实现上述功能的具体服务,如“国航机票查询”、“南航机票查询”、“支付宝”、“快钱”等服务.

第 2 阶段为服务动态绑定阶段.

为了使服务组合的 QoS 能够满足用户预先定义的 QoS 需求,开发框架需对若干组候选服务进行排列组合,在全部可能的组合方式中选择 QoS 最优的组合所对应的具体服务进行绑定.如通过计算得出“国航机票查询”与“支付宝”这一对服务组合的 QoS 最优,则在服务绑定阶段即可将“机票查询”绑定为“国航机票查询”服务,将“机票购买支付”绑定为“支付宝”服务.可见,服务的动态绑定过程实质上是候选服务基于 QoS 属性的组合优化过程.

### 1.2 服务选择的效率问题

用户的 QoS 需求是候选服务组合优化的重要依据.QoS 需求通常包括 QoS 约束与 QoS 目标两部分:

- QoS 约束通常可表示为用户要求应用的 QoS 属性取值范围 $a<g(x)<b$,其中,$a,b$ 为预设值;
- QoS 目标指用户对应用 QoS 属性的目标期望,通常表示为 $\min\{f(x)\}$,表示对于 QoS 属性 $x$,其函数值 $f(x)$ 越小越好.

然而,用户的 QoS 目标往往是多维的,并且可能存在冲突.例如,$\min\{cost\}\&\min\{response\text{-}time\}$ 表示用户期望应用的响应时间尽可能短,同时,所支付的服务使用费用尽可能低.但在实际中,响应时间短的服务往往价格较高;而价格低的服务,其响应时间却较长.因此,在进行服务组合优化时需要对多个 QoS 目标权衡.

- 一种解决办法是多个 QoS 目标加权求和,将多维目标函数简化为一维目标函数[16,17].该方法的优点是简单易操作,缺点是各 QoS 目标加权系数难以确定,且对优化结果的影响较大.
- 另一种是启发式方法[14,18,19],基于 Pareto 最优原理选出一组满足用户目标需求的最优非劣解集合.该方法的优势在于无需用户指定 QoS 目标的加权系数,并且优化结果是一组最优非劣解集合,能够为用户

提供更多的选择[19].

在效率方面,启发式算法能够在多项式时间内得出优化结果,算法计算复杂度随问题规模呈线性增长趋势.然而,在本文所述的动态服务选择场景中,服务选择是由服务发现和服务绑定两个阶段完成,其中,服务绑定是在应用运行时由用户动态触发的.因此,从用户体验的角度而言,用户对服务绑定的及时性有着更高的要求.当服务选择的规模扩大到一定程度时,现有的算法将很难满足用户的这一需求.例如,表 1 为采用遗传算法实现服务组合优化的实验数据.实验中,假设每个抽象服务所对应的候选服务个数是相同的:

- 第 1 组实验设定每个抽象服务都对应 50 个候选服务,通过改变抽象服务的个数(即,服务组合规模)观察算法执行时间的变化;
- 第 2 组实验设定服务组合中包含 10 个抽象服务,通过改变抽象服务所对应的候选服务的个数观察算法执行时间的变化.

**Table 1** Execution time of genetic algorithm

**表 1** 遗传算法的执行时间

| 最大迭代次数 | 种群规模 | 实验 1(候选服务个数=50) | | 实验 2(抽象服务个数=10) | |
|---|---|---|---|---|---|
| | | 抽象服务个数 | 执行时间(ms) | 候选服务个数 | 执行时间(ms) |
| 10 000 | 1 000 | 5 | 25 343 | 20 | 23 766 |
| | | 10 | 48 547 | 40 | 40 656 |
| | | 15 | 109 125 | 60 | 55 235 |
| | | 20 | 157 822 | 80 | 70 938 |
| | | 25 | 252 079 | 100 | 106 360 |

表 1 的实验数据表明:当抽象服务个数和候选服务个数增加到一定规模时,算法的执行效率明显下降.例如,当抽象服务个数为 25、每个抽象服务对应 50 个候选服务时,算法的执行时间是 252 079ms.这意味着用户在访问应用时,需要在线等待约 4.2 分钟才能得到服务的响应,远远超出用户的忍耐极限.而响应时间又是用户感受最迫切且最关注的应用性能评价指标,是决定应用竞争力的关键因素.因此,在现有启发式算法的基础上研究更加高效的服务选择算法,使其对服务组合规模及候选服务规模体现出更好的扩展性,对于应用运行时快速选择出满足用户 QoS 需求的服务组合,提高应用响应效率和增强用户体验具有十分重要的意义.

本文将在传统启发式算法的基础上,针对服务封装以及服务选择过程中服务绑定的特点,提出一种基于服务功能规约的服务选择方法,通过缩小服务组合规模及候选服务规模,提高服务选择的整体效率.

## 2 基于服务功能规约的服务选择方法

由表 1 的实验数据可知,服务组合规模和候选服务规模是影响服务选择效率的两个关键因素.本文通过对应用开发人员所描述的抽象服务进行功能上的规约,即,将若干功能相关的抽象服务合并为粒度更大的抽象服务,减少抽象服务个数.并且在执行服务组合优化时,只基于规约后的大粒度抽象服务进行绑定,从而有效缩减候选服务的规模,进而提高服务选择的整体效率.

基于服务功能规约的服务选择方法可概括为 3 个步骤,过程如图 2 所示.

- 首先,将功能相关的抽象服务规约为一个粒度更大的抽象服务,如“机票查询”服务和“机票购买支付”服务可合并为一个同时包含“机票查询”和“购买支付”操作的“机票订购”服务.
- 接着,对合并后的抽象服务组合实施组合优化.若抽象服务间存在多种规约方案,则需对方案分别进行组合优化,每种规约方案都对应一组最优非劣解集合,每个解即是由若干具体服务组合而成的组合服务.为了提高效率,多组组合优化可并行实施.
- 最后,将上述最优非劣解集合进行合并和排序,得到最终的最优非劣解集合.集合中的每一个解所对应的服务组合都能满足用户的功能需求,且其 QoS 是最优非劣的,用户可根据实际需要选择其中的一个解进行服务绑定.

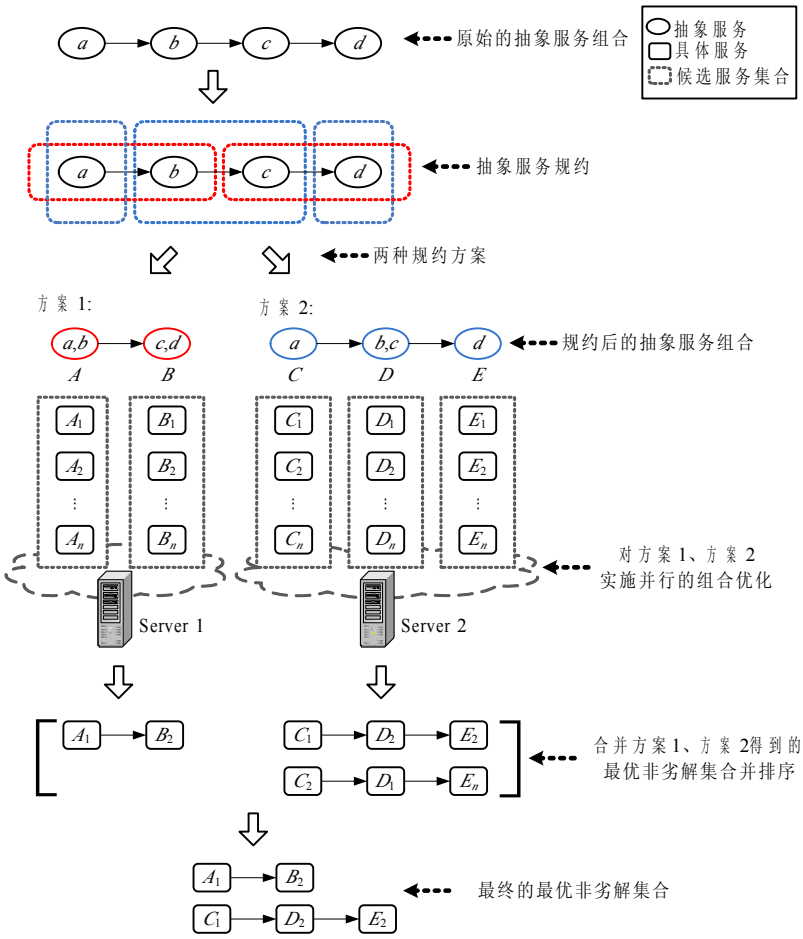



Fig.2 Service selection approach based on functionality folding

图 2 基于服务功能规约的服务选择方法

### 2.1 抽象服务规约

#### 2.1.1 规约概念及其意义

抽象服务是应用业务功能需求的一种描述.抽象服务规约即是对服务功能需求进行合并的操作.规约的目的是尽可能地减少抽象服务的个数,提高组合优化的效率.

在服务选择过程中,每个抽象服务都有一个候选服务空间.这些候选服务的一个共同特征是都具备抽象服务中所描述的功能操作.然而,服务通常是对若干相互关联操作的封装[20],例如,电影服务 *MovieService* 可能同时提供了查看电影列表的操作 *getMovieList*(·)以及获得电影详细信息的操作 *getMovieInfo*(*movieID*).这些操作在 Web 服务描述语言(Web service description language,简称 WSDL)中通过一组〈*operation*〉〈/*operation*〉标签定义,是服务功能的最小定义单元.因此,服务绑定时需要针对抽象服务指定一个候选服务的具体操作.本文把操作粒度的抽象服务称为操作级抽象服务(operation-level abstract service,简称 O-AS).

为了提高服务组合优化的效率,本文在操作粒度上对抽象服务进行规约,将可能绑定到同一具体服务不同操作的抽象服务规约为一个新的抽象服务,称为服务级抽象服务(service-level abstract service,简称 S-AS).如图 3 所示.

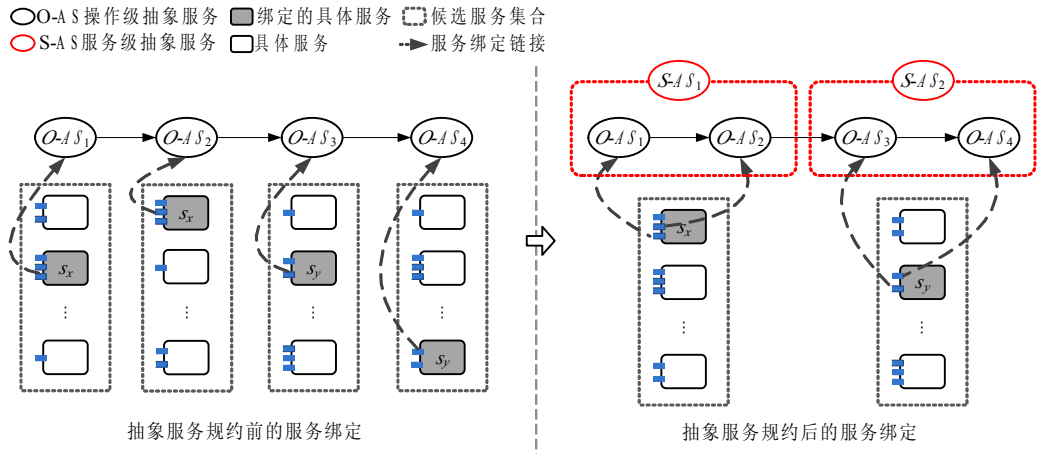



Fig.3 Service binding before/after abstract service folding

图 3 抽象服务规约前/后的服务绑定

规约操作是将操作级抽象服务(O-AS)合并为服务级抽象服务(S-AS)的过程.例如,用户的需求可通过两个操作级抽象服务 *O-AS*$_1$:*getMovieList* 与 *O-AS*$_2$:*getMovieInfo* 描述.服务 *MovieService* 封装了操作 *getMovieList*(·)与 *getMovieInfo*(*movieID*).在抽象服务规约前,服务 *MovieService* 将重复出现在抽象服务 *O-AS*$_1$:*getMovieList* 与 *O-AS*$_2$:*getMovieInfo* 对应的候选服务集合中,经过抽象服务规约后,操作级抽象服务 *O-AS*$_1$:*getMovieList* 与 *O-AS*$_2$:*getMovieInfo* 成为一个服务级抽象服务 *S-AS*:*MovieSearch*[*getMovieList*&*getMovieInfo*],要求其候选服务必须同时提供 *getMovieList* 与 *getMovieInfo* 两种操作.这样,服务 *MovieService* 只需出现在服务级抽象服务的候选服务集合中.可见,规约操作可以减少参与组合优化的候选服务个数,从而提高组合优化的效率.另一方面,通过规约操作,将用户需求的功能进行合并,减少应用对服务的需求数量,不但能提高服务组合优化的效率,还可以有效降低应用运行时,服务间的通信负担,提高应用的执行效率.

2.1.2 抽象服务规约的建模

服务是对一组操作的封装.这可用于建立操作级抽象服务(O-AS)的规约关系.例如,电影服务 *MovieService* 封装了 *getMovieList*(·)和 *getMovieInfo*(*movieID*)两个操作,我们可以利用这两个操作将与之匹配的操作级抽象服务 *O-AS*$_1$:*getMovieList* 和 *O-AS*$_2$:*getMovieInfo* 规约到一个关于电影搜索的抽象服务 *S-AS*:*MovieSearch*.规约的原则是,将描述用户需求的操作级抽象服务(O-AS)尽可能地映射到同一服务上.因此,抽象服务规约问题可视为具体服务集合与操作级抽象服务(O-AS)集合的非冗余集合覆盖问题,其定义如下:

**定义 1**(**覆盖**). 令 *op* 为用户需求的某个功能点,$O=\{o_i|i=1,2,3,\ldots,w\}$为服务 *s* 封装的操作集合,若$\exists o_i\in O$,使得 $o_i=op$,则称服务 *s* 覆盖功能点 *op*,记为 *s*.*contains*(*op*)=TRUE.

**定义 2**(**集合覆盖**). 令用户需求的功能点集合为 $OP=\{op_i|i=1,2,3,\ldots,n\}$,对于服务组合 $CS=\{s_j|j=1,2,3,\ldots,m\}$,若$\forall op_i\in OP,\exists s_j\in CS$,使得 $s_j.contains(op_i)$=TRUE,则称服务组合 *CS* 覆盖功能点集合 *OP*,记为 $CS\blacklozenge_{covers}OP$.

**定义 3**(**非冗余集合覆盖**). 令用户需求的功能点集合为 $OP=\{op_i|i=1,2,3,\ldots,n\}$,对于服务组合 $CS=\{s_j|j=1,2,3,\ldots,m\}$,若 $\forall op_i\in OP,\exists s_j\in CS$,使得 $s_j.contains(op_i)$=TRUE,且 $\forall s_k\in CS,\forall s_u\in CS-\{s_k\},\exists op_w\in OP$,使得 $s_k.contains(op_w)$=TRUE,$s_u.contains(op_w)$=FALSE,即,对于 *CS* 中的任意服务 *s*,若集合 *OP* 中至少存在 1 个 *op* 是被 *s* 唯一覆盖的,则称服务组合 *CS* 非冗余覆盖功能点集合 *OP*,记为 $CS\diamondsuit_{covers}OP$.

非冗余集合覆盖问题的数学模型表示为

$$\begin{cases}\min\ y=[1,1,\ldots1]_r^T\times X^T\\ \text{s.t.}\ \ M\times X^T\geqslant[1,1,\ldots1]_r^T\end{cases}\tag{1}$$

其中,

(1) $M=(m_{ij})$为 $n$ 行 $r$ 列的 0-1 矩阵($0\leqslant i\leqslant n, 0\leqslant j\leqslant r$).$n=|OP|$,即,用户需求的功能点的个数;$r$ 为服务库中可用服务资源的个数.$m_{ij}=1$,当且仅当 $s_j.contains(op_i)$=TRUE,即,第 $j$ 个服务 $s_j$ 覆盖了用户需求的第 $i$ 个功能点 $op_i$;否则,$m_{ij}=0$.

(2) $X=(x_j)$为 $r$ 维 0-1 向量,$r$ 为服务库中可用服务资源的个数.$x_j=1$,当且仅当第 $j$ 个服务 $s_j$ 被选中;否则,$x_j=0$.

(3) $M\times X^T\geqslant[1,1,...1]_r^T$ 确保用户需求的每一个功能点,都至少被 1 个服务所覆盖.

(4) $\min y=[1,1,...1]_r^T\times X^T$ 确保对于每一个服务,都至少存在 1 个功能点是被该服务唯一覆盖的,即,保证解向量 $X$ 的非冗余性.

根据上述定义,抽象服务规约问题就是对于用户指定的功能点集合 $OP$,求解一组服务组合集合 $CSS$(*Composite Service Set*)=$\{CS_i|i=1,2,3,\ldots,t\}$,使每一个 $CS\in CSS$ 都满足 $CS\diamondsuit_{covers}OP$.其中,每一个服务组合 $CS$ 都对应一种抽象服务规约方案.如图 4 所示,$M$ 表示服务对用户需求的功能点的覆盖关系矩阵,$CS_1=\{s_4,s_6,s_8\}$,$CS_2=\{s_2,s_3,s_5\}$为两个非冗余覆盖功能点集合 $OP$ 的服务组合,体现了两种可能的规约方案.例如,服务 $s_4$ 封装了操作 $op_1$,$op_2$ 和 $op_6$,表明 $op_1$,$op_2$,$op_6$ 所对应的操作级抽象服务可规约为一个更大粒度的服务级抽象服务.

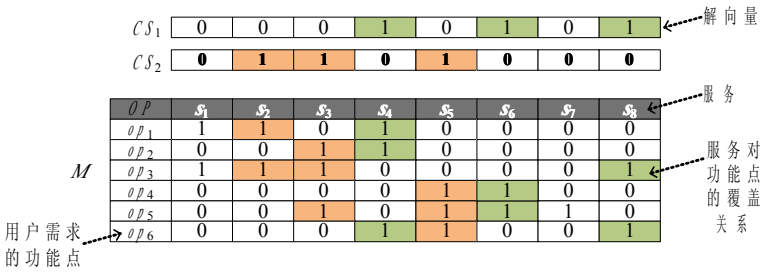



Fig.4 Sketch map of set covering

图 4 集合覆盖示意图

2.1.3 基于遗传算法求解抽象服务规约

本文基于遗传算法的思想,设计了 NR-SCP(non redundant - set covering problem)算法求解服务对需求功能点的非冗余集合覆盖问题,即,抽象服务的规约问题.

**算法 1**. NR-SCP.

输入:用户需求的功能点集合 $OP$,

服务库中的服务资源集合 $S$,

种群规模 $ps$,

最大进化代数 $g$max.

输出:非冗余覆盖 $OP$ 的服务组合集合 $CSS$.

1. $C$=*preprocess*($S$,$OP$);
2. $M'$=*initMatrix*($C$,$OP$);
3. $P_0$=*initPopulation*(|$C$|,$ps$);
4. for ($i$=0; $i$<$g$max; $i$++){
5. $P_{i+1}$=*evolve*($P_i$);
6. $P_{i+1}$=*complete*($P_{i+1}$);
7. $P_{i+1}$=*filter*($P_{i+1}$);
8. $P_{i+1}$=*select*($P_{i+1}$);
9. }
10. $CSS$=*decode*($P_{g\max}$);

步骤 1 对服务资源库中全部服务资源 $S$ 进行预处理操作,将其所封装的操作 $O=\{o_i|i=1,2,3,\ldots,w\}$ 在用户需求的功能点集合 $OP=\{op_i|i=1,2,3,\ldots,n\}$ 上进行投影,并根据投影结果对服务进行聚类,将投影结果相同的服务划分到同一簇中.若投影结果为空集,表明该服务不包含用户需求的功能点,则直接将该服务丢弃.经过聚类,得到的服务簇集合表示为 $C$,其中,每一个服务簇中的服务在集合 $OP$ 上都覆盖了相同的功能点集合,如图 5 所示.经过服务预处理操作,服务集合 $S$ 对 $OP$ 的非冗余集合覆盖问题即演化为服务簇集合 $C$ 对 $OP$ 的非冗余集合覆盖问题.

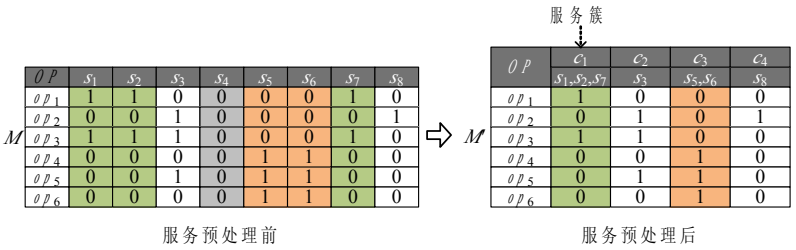



Fig.5 Service preprocess

图 5 服务预处理

步骤 2、步骤 3 分别对关系矩阵 $M$ 和种群 $P$ 进行初始化操作.其中,染色体编码采用定长二进制编码方式,以步骤 1 中得到的服务簇的个数 $|C|$ 为编码长度,每个基因位对应一个服务簇,基因位为 1 表示该服务簇被选中,基因位为 0 表示该服务簇未被选中.

步骤 5~步骤 8 实现种群的迭代进化.步骤 10 对进化结果进行解码,将染色体编码解析为相应的抽象服务组合.如图 6 所示,染色体 $CS_1$="1110"对应的抽象服务组合为 $\{c_1,c_2,c_3\}$.

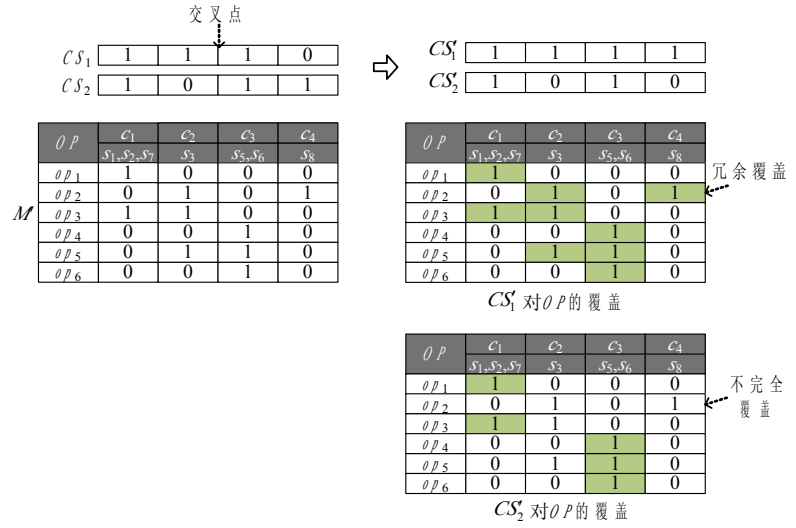



Fig.6 Impact of genetic manipulation on non-redundant set covering constrain

图 6 遗传操作对非冗余集合覆盖约束的影响

算法中,步骤 5 采用基本的遗传操作(多点交叉算子及基本位变异算子)完成种群进化.但由于交叉点和变异位选择的随机性,导致进化后的个体不一定满足非冗余集合覆盖的要求.如图 6 所示,个体 $CS_1$,$CS_2$ 经过交叉遗传操作后进化为个体 $CS'_1$ 和 $CS'_2$.其中,

- 个体 $CS'_1$ 全部基因位为 1,即,服务簇 $c_1,c_2,c_3,c_4$ 全部被选中,使得功能点 $op_2$ 被 $c_2$ 和 $c_4$ 重复覆盖;

- 相反地,个体 $CS'_2$ 的第 2 和第 4 基因位为 0,即,服务簇 $c_2$,$c_4$ 未被选中,使得功能点 $op_2$ 未能被覆盖.

为了使进化后的个体仍能满足非冗余集合覆盖的要求,本文针对不完整个体(例如 $CS'_2$)和冗余个体(例如 $CS'_1$)分别设计了补偿算子 *complete*(·)和过滤算子 *filter*(·).

(1) 补偿算子

补偿算子用于将不能完全覆盖 *OP* 的个体补偿为能够完全覆盖 *OP* 的个体.补偿的原则是:优先选择那些能够尽可能多地覆盖尚未被覆盖功能点的服务簇,直到 *OP* 中的全部功能点都被覆盖为止.

候选服务簇的优先级计算公式如下:

$$Pri(j)=\sum_{i\in UCOP,\ j\in C\cap UCOP} M_{i,j} \tag{2}$$

其中,*UCOP* 表示未覆盖功能点的集合,$C\cap UCOP$ 表示能够覆盖 *UCOP* 中任意一个功能点的服务簇的集合.例如图 6 中,对于个体 $CS'_2$,其 *UCOP* 为{$op_2$},$C\cap UCOP$ 为{$c_2$,$c_4$},则 $Pri(c_2)=1$,$Pri(c_4)=1$.当 $C\cap UCOP$ 中全部服务簇的优先级都相同时,则随机选取其中的一个进行补偿,将该服务簇所对应的基因位由 0 更改为 1.

(2) 过滤算子

过滤算子用于过滤冗余个体中的冗余基因位,消除冗余覆盖的服务簇.冗余基因位的判定规则如下:

**规则 1**. 对于个体 *CS* 满足 $CS\blacklozenge_{covers}OP$,将 *CS* 中的某个基因位 $g_i$ 由 1 变更为 0,得到新的个体 *CS'*.若 *CS'* 仍满足 $CS'\blacklozenge_{covers}OP$,则基因位 $g_i$ 为个体 *CS* 中的冗余基因位.

图 6 中,对于个体 $CS'_1$,将其第 2 个基因位 $g_2$ 变更为 0 后,得到的新个体“1011”仍能覆盖 *OP*.根据规则 1,可推断 $g_2$ 为 $CS'_1$ 中的冗余基因,可予以过滤.

经过补偿和过滤操作后,种群中的个体都能满足非冗余集合覆盖的约束.

为了选择优势个体参与下一轮进化过程,本文以个体中服务簇的利用率作为判定个体优劣的标准,定义如下个体适应度函数:

$$\begin{cases} Fitness(CS)=\sum_{c_i\in C} UR(c_i) \\ UR(c_i)=|SCOP|/|COP| \end{cases} \tag{3}$$

其中,$UR(c_i)$为服务簇 $c_i$ 的利用率,$c_i$ 为个体 *CS* 中基因位为 1 的基因所对应的服务簇;*SCOP* 表示仅被服务簇 $c_i$ 单独覆盖的功能点集合;*COP* 表示被服务簇 $c_i$ 覆盖的全部功能点集合;|*SCOP*|,|*COP*|则分别为集合 *SCOP* 与 *COP* 的势.例如图 6 中,

$$Fitness(CS_1)=1/2+1/3+2/3=1.5,$$

$$Fitness(CS_2)=2/2+3/3+1/1=3.$$

个体 $CS_2$ 中,每个服务簇都得到有效利用,不存在功能点冗余覆盖的情况;而在 $CS_1$ 中,$op_3$ 被 $c_1$,$c_2$ 重复覆盖,$op_5$ 被 $c_2$,$c_3$ 重复覆盖.因此,$CS_2$ 优于 $CS_1$.

### 2.2 服务组合优化

经过抽象服务规约操作,可得到一组满足非冗余集合覆盖约束的抽象服务组合 *CSS*.服务组合优化的目的就是:为抽象服务组合中的每一个抽象服务绑定一个具体的服务,并使所得到的服务组合满足用户预先定义的 QoS 需求.

如图 3 所示,规约操作将用户定义的操作级抽象服务组合转换为一组功能等价的服务级抽象服务组合.由于服务级抽象服务是操作级抽象服务在功能粒度上进行合并规约得到的,因此,在服务级抽象服务的粒度上实施服务组合优化的问题空间与操作级抽象服务相比大大减少;并且,多个服务级抽象服务组合可以在多台服务器上实施并行的组合优化,从而有效降低服务组合优化的时间开销.

为了能够为用户提供更多的服务选择方案,同时避免 QoS 属性权重的不当设置对组合优化结果可能造成的不良影响,本文基于遗传算法的思想设计了 GA4SCO(genetic algorithm for service combinatorial optimization)算法,实现 QoS 感知的服务组合优化.

**算法 2**. GA4SCO.

输入:服务级抽象服务组合 *CS*,

服务簇集合 *C*,

用户 QoS 属性需求,包括 Qos 目标 $F(x)$及 Qos 约束 $G(x)$,

种群规模 *ps*,

最大进化代数 *g*max.

输出:满足用户 QoS 属性要求的具体服务组合 *CCS*.

1. $P_0$=*initialization*(*CS*,*C*,*ps*);
2. for ($i$=0; $i$<*g*max; $i$++){
3. $P_i'$ =*crossover*($P_i$);
4. $P_{i+1}$=*mutation*( $P_i'$ ,*C*);
5. $P_{i+1}$=*selection*($P_{i+1}$,$F(x)$,$G(x)$);
6. }
7. *CCS*=*decode*($P_{g\max}$);

其中,步骤 1 对种群 *P* 进行初始化操作.染色体编码采用定长整数编码方式,以算法 NR-SCP 得到的抽象服务组合 *CS* 中抽象服务的个数为编码长度,每个基因位对应一个抽象服务,基因位的取值范围为该抽象服务所对应的服务簇中的服务编码集合.如图 7 所示.

步骤 3、步骤 4 对种群实施交叉、变异操作.步骤 5 选择满足用户 QoS 约束的且有利于用户 QoS 目标最大化的优势个体进入下一轮进化过程,直到迭代次数达到预先设定的最大进化代数.该过程是一个典型的利用遗传算法求解多目标组合优化问题的过程[14],本文不再赘述.步骤 7 对求得的优势染色体进行解码,得到满足用户 QoS 需求的服务组合.如图 7 所示,染色体“1002,1003,1004”对应的服务组合为{$s_2$,$s_3$,$s_5$}.

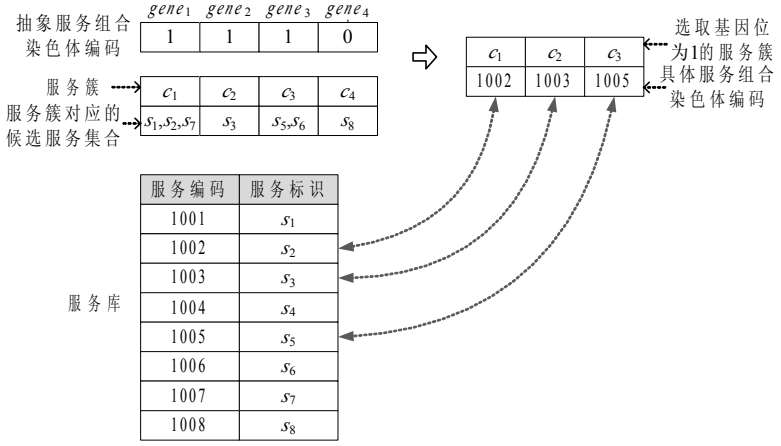



Fig.7 Example of chromosome coding

图 7 染色体编码示例

### 2.3 局部优化结果合并排序

通常,抽象服务间可存在多种规约方案.对多种规约方案分别实施组合优化操作,可得到多个局部的组合优化结果.本节将对局部优化结果进行合并和排序(如图 8 所示),得到全局最优非劣解集合,使得解集中的每一个解都能满足用户的 QoS 需求,且在全部可行解的范围内是最优非劣的.用户可根据当前应用的需要,选择其中的一个解进行服务绑定.

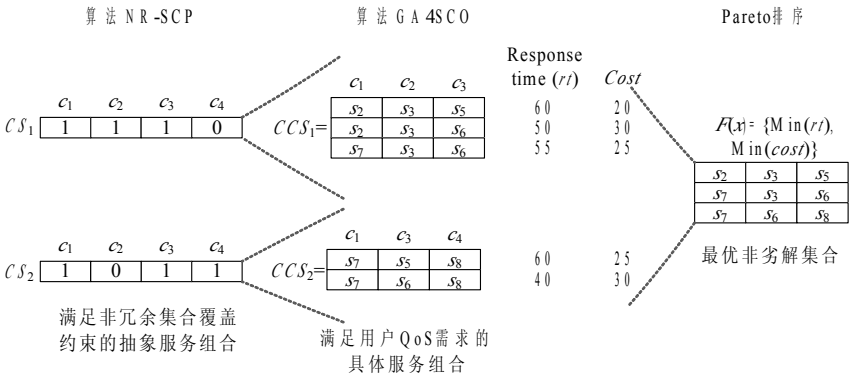



Fig.8 Merging and ranking of local optimization results

图 8 局部优化结果合并排序

为了提高排序效率,本文将局部优化结果(即局部最优非劣解集合)两两结为一组,各分组间可并行完成解集的合并和排序.各分组取得的局部最优非劣解集合继续两两合并排序,直到分组数为 0,此时得到的最优非劣解集合即为全局最优非劣解集合.算法描述如下:

**算法 3**. MERGING.

输入:$n$ 个局部最优非劣解集合 $S=\{LOR_1,LOR_2,\ldots,LOR_n\}$.

输出:全局最优非劣解集合 $GOR$.

```
1.  S'=∅;
2.  for (i=1; i≤|S|/2; i++){
3.      S'=S'+LOCALMERGING(LOR_i,LOR_{i+|S|/2});
4.  }
5.  if (|S|%2==1) S'=S'+LOR_{|S|};
6.  while (|S'|>1){
7.      S=S';
8.      MERGING(S);
9.  }
10. output(S');
```

**算法 4**. LOCALMERGING.

输入:局部最优非劣解 $LOR_i,LOR_j$,

取极小值的 QoS 目标函数 $F(x)$.

输出:合并后的最优非劣解.

```
1.  for (m=0; m<|LOR_j|; m++){
2.      lor_m=LOR_j.getElement(m);
3.      x=0;
4.      for (n=0; n<|LOR_i|; n++){
5.          lor_n=LOR_i.getElement(n);
6.          if (F(lor_m)≺F(lor_n)){
7.              LOR_i.deleteElement(lor_n);
8.              if (LOR_i.notContain(lor_m))
9.                  LOR_i.addElement(lor_m);
```

```
10.       }
11.       else if (F(lor_n)==F(lor_m)){
12.          if (lor_n≠lor_m && !LOR_i.containElement(lor_m))
13.             LOR_i.addElement(lor_m);
14.       }
15.       else if (F(lor_n)≺F(lor_m)) break;
16.       else x++;
17.    }
18.    if (x=|LOR_i|) LOR_i.addElement(lor_m);
19. }
20. return LOR_i;
```

算法中,不同解集中的个体,基于 Pareto 最优[21]关系进行比较和排序.若个体 $lor_m$ 至少有 1 个 QoS 目标函数值小于个体 $lor_n$,而其他 QoS 目标函数值都不大于个体 $lor_n$,则个体 $lor_m$ 优于个体 $lor_n$,记为 $F(lor_m)\prec F(lor_n)$,其中,$F(x)$为取极小值的 QoS 目标函数.若个体 $lor_m$ 既有小于 $lor_n$ 的目标函数值,又有大于 $lor_n$ 的目标函数值,则个体 $lor_m$ 与 $lor_n$ 之间不存在 Pareto 占优关系,即,个体 $lor_m$ 与 $lor_n$ 不可比.最优非劣解集合中的解之间都是不可比的,统称为 Pareto 最优解.如图 8 所示,若要追求高响应速度,则必然花费高成本;相反,若要追求低成本,则必然牺牲响应时间.本文方法将全部 Pareto 最优解返回给用户,用户可根据应用的实际需要自主选择.

## 3 实验分析

### 3.1 实验数据准备

为了验证本文所提出的基于服务功能规约的服务选择方法正确有效,并且在执行效率上优于传统的基于启发式算法的服务选择方法,本文设计了以下的实验.

实验数据来源于 WS-DREAM(distributed reliability assessment mechanism for Web services)[22]的数据集 Dataset 1(http://www.wsdream.net/dataset.html),Dataset 1 中包含了 150 个文件.每个文件包含一个用户对 100 个服务的 10 000 次调用信息,即,每个服务调用 100 次.每条信息包含以下内容:用户 IP、所调用服务 ID、响应时间、数据大小、HTTP 响应代码以及 HTTP 响应信息(见表 2).

**Table 2** Samples of service invocation information

**表 2** 服务调用信息示例

| ClientIP | WSID | Response time (ms) | Data size | HTTP code | HTTP message |
|---|---|---|---|---|---|
| 35.9.27.26 | 8 451 | 2 736 | 582 | 200 | OK |
| 35.9.27.26 | 8 451 | 804 | 14 419 | 200 | OK |
| 35.9.27.26 | 8 451 | 20 176 | 2 624 | −1 | java.net.SocketTimeoutException:connect timed out |

本文对 Dataset 1 中每个用户对每个服务的 100 次调用取平均值,作为该用户访问该服务时,服务的 QoS 表现,并以不同用户访问同一服务所获得的不同 QoS 来模拟部署在不同数据中心的服务实例的 QoS.同时,采用随机方式在给定范围内自动生成服务的其他 QoS 属性值.本文选取服务代价 cost、服务评价 rating 以及服务调用成功率 successful percentage of invocation(sp)作为除响应时间 response time、数据大小 data size 之外的服务 QoS 属性,对表 2 中的服务调用信息进行加工.加工后的服务调用信息见表 3.

**Table 3** Processed information of service invocations

**表 3** 加工后的服务调用信息

| Data center IP | WSID | Response time (ms) | Data size | Cost | Rating | Successful percentage (%) |
|---|---|---|---|---|---|---|
| 35.9.27.26 | 8 451 | 905 | 582 | 61 | 10 | 93.66 |
| 193.63.75.18 | 8 451 | 1 212 | 582 | 5 | 40 | 83.76 |
| 129.105.15.37 | 8 451 | 841 | 582 | 58 | 27 | 76.03 |

经上述数据加工后,本文共获取服务45个,每个服务对应100个部署在不同数据中心的服务实例,这些服务实例构成参与服务组合优化的候选服务集合.本文称加工后的数据集为Dataset 1′.

### 3.2 实验方案设计

本实验的目的是在验证本文方法(F-WSS)正确、有效的基础上,验证其与传统的QoS感知的服务选择方法(WSS)相比,在算法的执行效率上是否有所提升,以及服务组合规模和候选服务规模等参数对算法执行效率的影响.由于本文算法主要基于遗传算法实现,因此,本文以传统的基于遗传算法实现的QoS感知的服务选择作为本文方法的对比对象,通过变换服务组合规模、候选服务规模以及用户QoS需求等参数,考察算法在执行效率上的变化.

其中,服务组合规模的大小通过用户需求的功能点集合$OP$的势来反映,用户需求的功能点越多,表明服务组合的规模越大.候选服务规模通过两个参数反映:一是数据集Dataset 1′中服务的个数,即,数据集中WSDL描述文件的个数$ws_num$;二是每个服务所对应的部署在不同数据中心的服务实例的个数$iws_num$.候选服务规模即$ws_num$与$iws_num$的乘积.用户QoS需求包括QoS目标和QoS约束.将描述QoS目标的属性称为QoS目标属性.本文通过变换目标属性的个数考查算法的执行情况.

传统的WSS方法将用户的功能需求建模为一组抽象服务的组合,每个抽象服务对应一个功能需求,然后在抽象服务组合上实施组合优化,为每个抽象服务绑定一个具体服务的一个操作(operation);而本文的F-WSS方法首先对建模得到的抽象服务组合进行规约,将功能相关的若干抽象服务规约为一个粒度更大的抽象服务,然后在规约后的抽象服务组合上实施组合优化,为每个规约后的大粒度抽象服务绑定一个具体服务的多个操作(operation).

本文在数据集Dataset 1′上完成4组对比实验:

(1) 第1组实验考查用户需求的功能点数即服务组合规模对算法执行效率的影响.参数范围设定为$5\leqslant op_num\leqslant 25$,步长为5.其他实验参数设定为WSDL服务描述个数$ws_num$=45,每个服务描述对应的服务实例个数$iws_num$=50,QoS目标属性个数$qos_num$=5.实验结果如后文图10(a)所示.

(2) 第2组实验考查数据集中服务的规模对算法执行效率的影响.参数范围设定为$5\leqslant ws_num\leqslant 45$,步长为5.其他实验参数设定为$op_num$=10,$iws\_num$=50,$qos\_num$=5.实验结果如后文图10(b)所示.

(3) 第3组实验考查服务实例规模,即,每个服务对应的功能相同但QoS不同的服务实例的个数对算法执行效率的影响.为实验方便,本文假设数据集中每个服务都对应相同数目的服务实例,且其取值范围设定为$10\leqslant iws_num\leqslant 100$,步长为10.其他实验参数设定为$op_num$=10,$ws\_num$=45,$qos\_num$=5.实验结果如后文图10(c)所示.

(4) 第4组实验考查用户指定的QoS目标属性的个数对算法执行效率的影响.参数范围设定为$2\leqslant qos_num\leqslant 5$,步长为1.若用户指定$qos_num$个属性为应用的QoS目标,则其他$(5-qos_num)$个属性为应用的QoS约束.根据数据集Dataset 1′中数据的分布情况,QoS属性作为QoS约束时的取值范围设定为$response\ time\leqslant$2000ms,$data\ size\leqslant 3000$,$cost\leqslant 80$,$rating\geqslant 60$,$successful\ persentage\geqslant 60\%$.其他实验参数设定为$op_num$=10,$ws\_num$=45,$iws\_num$=50.实验结果如后文图10(d)所示.

由第1.1节可知,本文中,服务选择分为服务发现和服务动态绑定两个阶段.服务发现为每个抽象服务构造一个候选服务集合,可在应用的设计阶段完成.当应用的设计人员根据应用需求构造出一个抽象服务组合后,即可启动服务发现过程.与之相对地,服务动态绑定是在应用的运行阶段,由用户的QoS需求动态触发的,目的是为每个抽象服务在与之对应的候选服务集合内指定一个具体服务.可见,服务动态绑定是影响服务选择效率的关键因素.本文实验主要比较两种服务选择方法中服务绑定的时间代价.后文图10中的算法执行时间是指服务绑定算法的执行时间,其数值为算法执行10次所取的平均值.

本文中,参与对比的两种服务选择方法的服务绑定过程均采用遗传算法实现,而遗传算法中主要通过交叉和变异操作扩大搜索空间范围,保证种群个体的多样性,因此,应设置较高的交叉率和变异率.本文中,设定交叉率$PC$=1,变异率$PM$=0.1.此外,由于两组对比实验中遗传算法的初始种群都采用随机的方式生成,为了保证初始

种群的多样性,避免算法过快地陷入某一局部最优解中,本文设置了较大的种群规模和进化代数.根据本文实验数据集的大小及文献[23]的经验值,设定种群规模 *POP_SIZE*=候选服务规模/20,最大进化代数 MAX_*GENERATIONS*=1000.

本文实验硬件环境配置为 Intel(R) Core(TM)2 Duo CPU T9400 @ 2.53GHz 处理器,2GB 内存,Windows 7 旗舰版 32 位操作系统;软件环境配置为 eclipse3.4-jee-ganymede 开发运行平台,JDK6.0 版本.

### 3.3 实验结果与分析

#### 3.3.1 算法的有效性

F-WSS 方法通过规约操作,减少参与组合优化的抽象服务个数以及抽象服务对应的候选服务个数,从而提高算法的执行效率.但在规约的同时,可能引起丢解现象的发生.例如,抽象服务组合{*A*,*B*,*C*},其中,抽象服务 *A* 对应的候选服务为 $s_1$,$s_2$,记为 *A*{$s_1$,$s_2$}.同理,有 *B*{$s_1$,$s_3$},*C*{$s_4$}.可能的组合包括{$s_1$,$s_4$},{$s_1$,$s_3$,$s_4$},{$s_2$,$s_1$,$s_4$},{$s_2$,$s_3$,$s_4$}.对{*A*,*B*,*C*}进行规约,将 *A* 和 *B* 规约为 *A*′,则规约后的抽象服务组合为{*A*′,*C*},其中,*A*′对应的候选服务为 $s_1$,记为 *A*′{$s_1$}.规约后,可能的组合为{$s_1$,$s_4$}.由于 $s_1$ 同时包含了抽象服务 *A* 和 *B* 所描述的功能,因此,组合{$s_1$,$s_3$,$s_4$},{$s_2$,$s_1$,$s_4$}事实上分别冗余了服务 $s_3$ 和服务 $s_2$.对于组合{$s_2$,$s_3$,$s_4$},单从组合的 QoS 分析,{$s_2$,$s_3$,$s_4$}可能优于{$s_1$,$s_4$}.即,F-WSS 方法有可能丢失 QoS 更优的组合{$s_2$,$s_3$,$s_4$},而仅给出次优解{$s_1$,$s_4$}.但考虑到应用运行时服务间的通信代价以及应用的整体响应速度,则{$s_1$,$s_4$}明显优于{$s_2$,$s_3$,$s_4$},因为与后者相比,前者拥有更少的服务个数.

WSS 方法虽然没有对抽象服务进行规约操作,但初始种群的多样性以及交叉率和变异率等遗传操作参数的设定,都会影响算法收敛的效果,从而产生次优解,即,在某些 QoS 目标函数上的取值大于最优解的取值(以目标函数取极小值 min{*f*(*x*)}为例).本文通过统计次优解在解集中所占比例来判定算法的有效性.令服务组合规模即用户需求的功能点数 *op_num*=10,WSDL 服务描述个数 *ws_num*=45,每个服务描述对应的服务实例个数 *iws_num*=50,用户需求的 QoS 目标属性个数 *qos_num*=3.对 10 个用户的 10 个不同需求分别执行 F-WSS 和 WSS 两种方法,统计两种方法所产生的次优解在全部解中所占的百分比.

由图 9 可以看出,规约操作对产生次优解的影响远小于遗传算法中参数设定的影响;尤其是当服务组合规模和候选服务规模较大时,交叉率、变异率以及种群规模都会对算法的收敛产生较大影响.这也是为什么 WSS 方法中次优解的发生率较高的原因.而 F-WSS 方法通过规约操作,极大地降低了服务组合规模和候选服务规模,从而削弱了遗传算法参数对结果的影响.

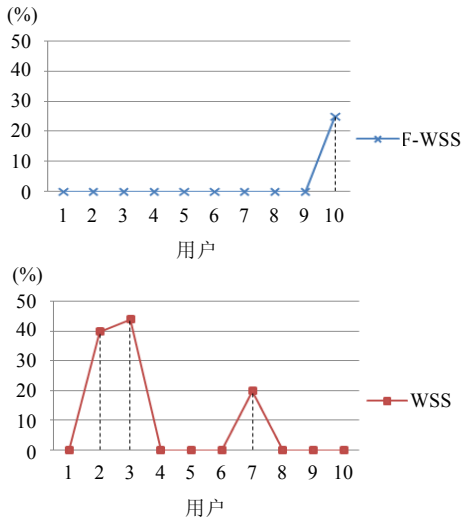



Fig.9 Experimental results on the effectiveness of the algorithm

图 9 关于算法有效性的实验结果

3.3.2 算法的执行效率

由图 10 可以看出,本文方法(F-WSS)在服务绑定的执行效率上优于传统的基于遗传算法实现的 QoS 感知的服务选择方法(WSS);尤其是当服务组合规模及候选服务规模较大时,本文方法在执行效率方面的优势就愈加明显.原因如下:

- WSS 方法中,服务绑定的时间代价 $T_{WSS}$就是服务按照用户给出的 QoS 需求完成组合优化的时间消耗,记为 $t_{CAS}$,表示在抽象服务组合 *CAS* 上实施组合优化所需花费的时间代价.
- 本文的 F-WSS 方法分 3 个步骤完成服务发现与绑定过程.首先对抽象服务组合 *CAS*,按照功能相关性进行规约,然后在规约后的抽象服务组合 F-CAS 上完成服务发现过程.由于抽象服务可能存在多种规约方案,而每一种规约方案都对应一个服务的组合优化过程,因此,服务绑定的时间代价 $T_{F\text{-}WSS}$由以下两部分组成:执行时间最长的服务组合优化的时间消耗(记为 $\max\{t_{F\text{-}CAS}\}$)以及多个组合优化结果合并排序的时间消耗(记为 $t_{ranking}$),即

$$\begin{cases} T_{WSS} = t_{CAS} \\ T_{F\text{-}WSS} = \max\{t_{F\text{-}CAS}\} + t_{ranking} \end{cases} \tag{4}$$

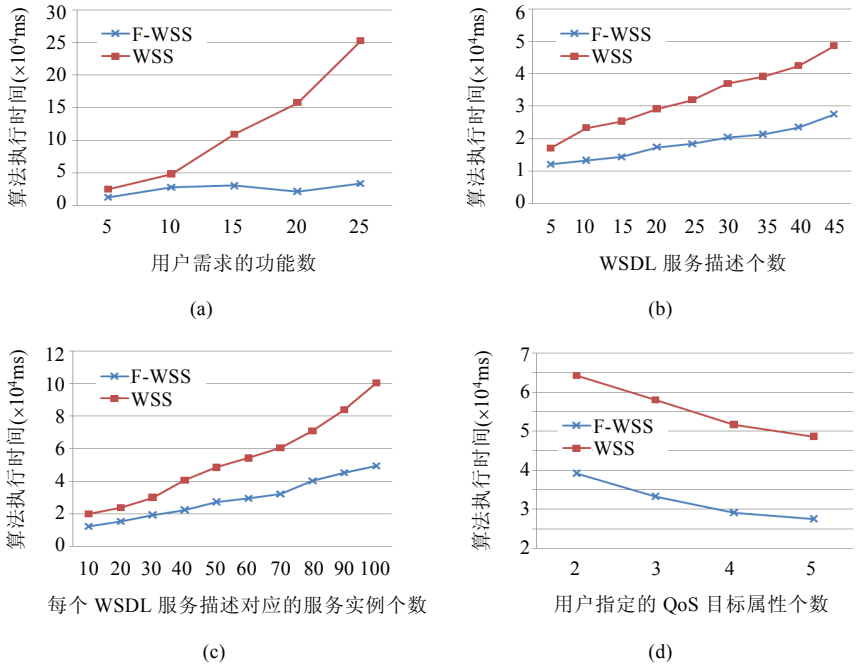



Fig.10 Experimental results on the efficiency of the algorithm

图 10 关于算法执行效率的实验结果

由于本文方法(F-WSS)在实施服务组合优化前对抽象服务采取了规约操作,而规约操作能够减少参与服务组合优化的抽象服务的个数(相当于缩短遗传算法中染色体的长度),同时减小参与服务组合优化的候选服务的规模.由第 2.1.1 节可知,规约操作将若干功能相关的抽象服务(O-AS)合并为一个粒度更大的抽象服务(S-AS),随后的服务发现过程只针对大粒度抽象服务(S-AS)构造候选服务集合,即,要求集合中的每一个服务必须涵盖大粒度抽象服务(S-AS)所描述的全部功能,因此,与规约前相比,抽象服务所对应的候选服务个数减少了.而候选服务个数的多少直接影响了遗传算法中种群大小的设定.候选服务规模越大,算法设定的种群规模往往也越大,以此保证种群的多样性,避免算法过快陷入某一局部最优解中.由文献[19]可知,服务组合优化的计算复杂度可表示为 $O((Q+L)N^2T)$,其中,$Q$ 为用户需求的 QoS 属性的个数,$L$ 为染色体长度,$N$ 为种群规模,$T$ 为算法迭代次数.当

$Q$与$T$保持不变时,算法的计算复杂度与染色体长度和种群规模成正比,因此有$t_{F\text{-}CAS}\leqslant t_{CAS}$.最坏的情况是:抽象服务组合$CAS$中不存在功能相近的抽象服务,规约的结果与规约前相同.既没有减少抽象服务的个数,也没有减少相应的候选服务的个数.此时,$t_{F\text{-}CAS}=t_{CAS}$.

实验结果表明(见表 4):

- 当用户需求的功能数为 5 时,规约操作的效果为 0,即,规约后抽象服务的个数保持不变;
- 当用户需求的功能数为 10~25 时,规约操作效果较为明显,规约百分比(即,规约后抽象服务个数与规约前个数的比值)最高可达 49.2%.

**Table 4** Statistics of the number of abstract services before and after folding

**表 4** 规约操作前后抽象服务个数统计

| 规约前抽象服务个数(用户需求的功能数) | 规约后抽象服务个数(运算 10 次取平均值) | 平均规约百分比(%) |
|---|---|---|
| 5 | 5 | 100 |
| 10 | 6.7 | 67 |
| 15 | 8.9 | 59.3 |
| 20 | 10.1 | 50.5 |
| 25 | 12.3 | 49.2 |

公式(4)中,$t_{ranking}$是对多个组合优化结果进行合并排序所消耗的时间代价,与参与比较的个体的数量成正比[19].而根据实验数据统计可知:抽象服务规约方案的个数以及每个方案所对应的组合优化结果的个数都不超过 10,即,参与比较排序的个体数量最大不超过 $10^2$.而基于遗传算法实现的服务组合优化,由于设定了较大的种群规模和迭代次数,根据实验参数设定,相当于对 1 000 个个体比较排序 1 000 次,参与比较排序的个体规模达 $10^6$.因此,$t_{ranking}<<t_{F\text{-}CAS}$,可忽略不计.

综上可得:$T_{F\text{-}CAS}\leqslant T_{CAS}$,且服务组合规模越大,意味着越多的抽象服务可能被规约为一个抽象服务,规约的效果越明显.因而,本文方法在执行效率上的优势也就越明显.

3.3.3 算法的扩展性

由图 10(a)可以看出,WSS 方法的执行时间随服务组合规模($op\_num$)的增长呈现出近似线性增长的趋势,而 F-WSS 方法的执行时间则表现出明显的波动的变化趋势.这是因为:

- 在 WSS 方法中,增加服务组合规模相当于增加参与组合优化的抽象服务的个数,因而导致算法执行时间的增长;
- 而在 F-WSS 方法中,由于采取了规约操作,使得增加服务组合规模未必引起抽象服务个数的增加.

例如,初始抽象服务个数为$a$,采取规约操作后,得到$a'$个抽象服务.现增加$b$个抽象服务,重新进行规约操作,可能得到 3 种结果:

(1) 新增加的$b$个抽象服务完全被规约到原来的$a'$个抽象服务当中.此时,抽象服务个数仍为$a'$,但由于其中包含了$b$个抽象服务所描述的功能,相当于服务发现时增加了功能匹配的过滤条件,使得$a'$个抽象服务所对应的候选服务的个数减少,其结果是增加$b$个抽象服务后,算法的执行时间反而降低了.

(2) 新增加的$b$个抽象服务中没有一个抽象服务能被规约到原来的$a'$个抽象服务中.此时,抽象服务个数增加为$a'+b'$,其中,$b'$为$b$个抽象服务的规约结果.算法的执行时间由于抽象服务个数的增加而增加.

(3) 新增加的$b$个抽象服务中有$\Delta b$个抽象服务能够规约到原来的$a'$个抽象服务中,其中,$0<\Delta b<b$.此时,抽象服务个数增加为$a'+(b-\Delta b)'$,其中,$(b-\Delta b)'$为剩余的$b-\Delta b$个抽象服务的规约结果.同时,$\Delta b$个抽象服务规约到$a'$个抽象服务中,使得$a'$个抽象服务所对应的候选服务的个数减少.因此,在抽象服务个数增加和候选服务个数减少的双重因素影响下,算法执行时间的增减趋势并不确定.

图 10(b)、图 10(c)反映了 F-WSS 和 WSS 两种方法的执行时间随候选服务规模($ws\_num\times iws\_num$)增长的变化趋势.无论增加 WSDL 服务描述的个数 $ws\_num$,还是增加每个 WSDL 服务描述所对应的候选服务实例的个数 $iws\_num$,F-WSS 和 WSS 两种方法的执行时间均呈现出增长的趋势.这是因为在抽象服务个数一定的情况下,不论是否进行规约操作,当候选服务规模增加到一定程度后,会影响算法中种群规模参数的设定,从而引起算法执行时间的增长.但由于规约操作能够缩减抽象服务所对应的候选服务个数,使得增加 WSDL 服务描述个

数 *ws_num* 或增加候选服务实例个数 *iws_num*,对规约后候选服务规模的影响小于规约前的影响,因此,F-WSS 方法时间开销的增长趋势与 WSS 方法相比略显趋缓.

此外,F-WSS 和 WSS 两种方法的执行时间随 QoS 目标属性个数的增长呈现出下降的趋势,如图 10(d)所示.这是由于算法执行时间的消耗很大一部分源于种群中个体的比较.用户指定的 QoS 目标属性越多,相当于个体比较时的对比条件越多,从而过滤出的最优非劣解的个数越少,择优效果越好,算法越能快速收敛到最优解;相反地,用户指定较少的 QoS 目标属性,使得大部分个体被选进最优非劣解集,择优效果较差,从而影响算法的收敛速度.

## 4 相关工作比较

QoS 感知的服务选择问题一直是服务计算相关领域中的热点话题.就问题建模而言,多数工作[16,24−27]采用抽象服务建模的方法,将用户的功能需求建模为一组抽象服务的组合,每个抽象服务对应用户的一个功能需求.抽象服务之间采用编制(orchestration)或编排(choreography)的方式定义逻辑关系.其中,服务编制[16,24,25]以流程的方式定义服务间的执行顺序,服务编排[26,27]以对等的消息交换模型定义服务间的点对点协作.服务选择,即是为模型中的每个抽象服务指定一个具体的、用以实现其描述功能的服务(具体到服务中的某个功能操作),同时保证所选择的服务能够满足用户的服务质量需求.本文在传统的抽象服务建模的基础上对抽象服务进行了规约,将其中功能相关的抽象服务合并为一个粒度更大的抽象服务.在传统的服务模型中,每个抽象服务对应一个具体服务中的一个操作;而在本文的服务模型中,抽象服务的刻画粒度更大了,每个抽象服务对应一个具体服务中的若干操作.

就服务选择策略而言,当前的研究工作可概括为局部最优策略、全局最优策略以及混合策略.局部最优策略[28,29]独立地考虑每个抽象服务,在各个抽象服务所对应的局部候选服务集合中选择 QoS 最优的具体服务.该策略的优点是思想简单,易于实现.其不足之处在于没有考虑服务组合的全局 QoS 约束和目标,使得按此策略选择出的具体服务所形成的组合服务不一定是全局最优的,也不一定满足用户的全局 QoS 约束.全局最优策略[16,18,19]从服务组合的全局 QoS 需求出发,综合考虑每个具体服务的 QoS,使各个具体服务组合而成的组合服务都能满足用户的 QoS 约束,且是全局最优的.但这种策略的计算复杂度较高,不适合实时性要求较强的应用场景.混合策略综合了上述两种策略的优势,既兼顾用户的全局 QoS 需求,又能保证算法的性能在用户可接受的范围内.文献[30]先采用局部最优策略对每个抽象服务所对应的候选服务集合进行过滤,筛选掉违反用户 QoS 约束的服务,缩小服务组合优化的空间,然后再采用全局最优策略,计算出满足用户全局 QoS 约束的最优解.文献[24,31,32]则采取约束分解的策略:先将用户的全局 QoS 约束利用混合整数规划(mixed integer programming,简称 MIP)分解为针对各个抽象服务的局部约束;然后,再采用分布式选择方法找出满足局部约束的最佳具体服务.本文所提出的基于服务功能规约的服务选择方法也属于一种混合策略,与文献[30]类似,都是通过缩减服务组合优化的问题空间来提高计算效率.有所不同的是:文献[30]保持原有抽象服务不变,在原有抽象服务的粒度上对候选服务进行过滤;而本文则是基于服务操作,将若干功能相关的抽象服务规约为一个大粒度的服务级的抽象服务,然后,针对服务级抽象服务进行过滤,进一步缩减候选服务集合,从而更加有效地提高服务组合优化的效率.

就服务选择的方法而言,包括穷举法、分支定界法以及以遗传算法为代表的智能优化算法等.穷举法是最简单、直观的一种服务选择方法,其核心思想是:列举出所有可能的服务组合方案,然后计算出每一个服务组合方案的 QoS,并从中选择最优的组合方案.假设有 $N$ 个抽象服务,每个抽象服务对应 $M$ 个候选服务,那么可能的服务组合方案有 $M^N$ 个.可以看出:当服务组合的规模和候选服务规模增大时,服务选择的时间代价将呈现出指数级的增长趋势.因此,穷举法比较适合于小规模的服务选择场景.文献[16,33]将 QoS 感知的服务选择问题建模为整数或线性规划问题,利用加权法将多个 QoS 目标函数转换为单目标函数,然后,应用分支定界法求解线性约束条件下的目标函数最优值.这种方法要求目标函数和约束条件都是线性的,且其优化的结果对 QoS 权值非常敏感,而用户通常很难给出恰当的权值分配.这些不足在一定程度上限制了此类方法的应用.近年来,以遗传算

法为代表的智能优化算法得到广泛关注和应用.文献[14,19,26−38]将 QoS 感知的服务选择问题建模为多约束下多目标组合优化问题,利用遗传算法的思想,通过种群的迭代进化,得到问题的近优解.文献[19]针对基本流程模型(串联、并联、选择、循环)定义了常见 QoS 属性(执行时间、执行费用、信誉等级、可靠性)的计算公式,并在此基础上,应用遗传算法求解最佳服务聚合流程.文献[34]提出了动态适应函数的概念,在进化过程中,使那些虽然违背了约束条件,但更接近于优势个体的个体得以保留,从而加快算法的收敛速度,提高解的质量.文献[35]提出了云环境下服务选择的解决方案,对 QoS 模型和算法都进行了扩展,增加了虚拟机服务、数据库服务以及网络服务等效用计算服务(utility computing services)特征,使服务选择算法能够适应云环境下的应用场景.文献[38]分析了云环境下服务选择面临的两种挑战:(1) 由服务部署位置所带来的网络延迟会对服务的响应时间造成不可忽视的影响,服务选择算法不仅要考虑服务本身的 QoS 属性,而且还需要考虑网络 QoS;(2) 服务的多实例部署(同一个服务在多个地方或多台机器上部署为多个服务实例)增加了问题规模,对算法的可扩展性提出了更高的要求.基于上述分析,我们扩展了 QoS 模型,增加了网络 QoS 属性特征,并采用遗传算法对大规模服务选择问题给出了多项式时间复杂度的解决方案.

尽管遗传算法在执行效率和扩展性方面都体现出了相当大的优势[36],但面对动态网络环境和实时应用场景却表现得差如人意.研究更加高效的服务选择算法,仍然是当前服务计算研究领域中的核心内容[25].文献[25]通过优化初始解来提高算法的整体执行效率.文中认为:初始解越接近最优解,算法的收敛速度就越快.于是,利用线性规划模型找出一个接近最优解的初始解;然后,再应用启发式算法(如爬山算法)对初始解进行迭代优化,直到找出满足用户 QoS 约束的近优解.文献[25]中获取初始解的方法是:利用文献[39]的成果,为每个抽象服务所对应的候选服务,根据其被调用的历史标识一个调用概率,然后选择调用概率最高的候选服务构成初始解.可见,这种初始解的构造方法是建立在大量服务调用的历史数据上的.这就不可避免地会遇到“冷启动”问题,也就是说,当系统无法获取服务调用的历史数据或者数据尚未丰富的时候,该方法将很难构造出令人满意的初始解.另外,考虑云环境下网络 QoS 对服务调用的影响,单从服务的调用概率很难评判出个体的优势与否,因为服务和用户的位置信息会对服务的使用情况产生巨大影响.而本文主要是通过合并用户需求(即抽象服务)减小服务组合规模,同时减小候选服务规模,达到提高算法执行效率的目的.本文将服务选择分两个阶段完成:第 1 阶段,将功能相近的抽象服务进行合并,该操作可以在设计阶段完成,不会影响应用运行时的执行效率;第 2 阶段,针对合并后的抽象服务组合,在缩减后的候选服务集合上,应用遗传算法进行求解.文献[40]与本文思想类似,将功能相似的服务聚合在一个“服务池”中,根据用户 QoS 需求的不同,提供声明式服务发现和编程式服务发现两种服务发现方案.有所不同的是,文献[40]聚合的是服务;而本文聚合的是用户需求,即抽象服务.前者通过服务聚合为用户屏蔽服务的多样性和多变性,提供相对稳定和统一的资源视图;后者通过抽象服务聚合,减小服务组合优化的问题规模和搜索空间,提高算法的执行效率.

## 5 结 论

服务选择是面向服务软件开发过程中的重要内容,从面向功能的服务选择到 QoS 感知的服务选择、从 QoS 局部最优的服务选择到 QoS 全局最优的服务选择,服务选择方法无论在方法的执行效率方面,还是在结果的质量方面都得到了很大的提升.尤其是遗传算法、蚁群算法等智能优化算法的应用,使得服务选择算法能够在多项式时间内完成,并对问题规模(服务组合规模及候选服务规模)体现出良好的扩展性.然而,随着网络应用的不断发展,用户对应用的实时性要求越来越高,现有的服务选择方法很难满足用户的这一需求.针对这一问题,本文提出基于服务功能规约的服务选择方法,将功能相关的抽象服务规约为一个粒度更大的抽象服务,并在大粒度的抽象服务上完成服务发现和组合优化,从而减少参与组合优化的抽象服务的个数及相应的候选服务空间的规模,提高算法的执行效率.实验结果表明,该方法与传统的利用遗传算法实现服务组合优化的方法相比,能够有效降低算法的执行时间,并在服务组合规模及候选服务规模上表现出更佳的扩展性.虽然规约操作可能造成部分近优解的丢失,但这远远小于算法参数本身对解的质量的影响,并且就算法的执行效率而言,得到了极大的提升,特别是考虑到应用运行时服务间的通信代价对应用整体响应速度的影响,这种代价也是值得的.

提高服务选择的质量和效率是服务计算领域中的一个永久性话题.本文主要从效率方面入手,以缩短算法执行时间和提高算法扩展性为主要目的.下一步的工作将主要集中在服务选择的质量方面,通过融入用户偏好,努力提高解的质量,为不同用户提供个性化的解决方案,提高用户对服务选择结果的满意度.

**References**:


[1] Keith, M. Towards the coordination of service-oriented software development [Ph.D. Thesis]. Tempe: Arizona State University, 2009.

[2] High R, Kinder JS, Graham S. IBM SOA Foundation: An Architectural Introduction and Overview. Version 1.0, 2005.

[3] Ghosh A, Haugland S, Kennedy M, Smith R, Thom C, Vijayaraghavan S. Oracle Fusion Middleware: Developer's Guide for Oracle SOA Suite. 2012.

[4] Bieber G, Carpenter J. Introduction to Service-Oriented Programming (Rev 2.1). Openwings, 2001.

[5] Hoyer V, Janner T, Delchev I, Fuchsloch A, López J, Ortega S, Fernández R, Möller KH, Rivera I, Reyes M, Fradinho M. The FAST platform: An open and semantically-enriched platform for designing multi-channel and enterprise-class gadgets. In: Proc. of the Int'l Conf. on Service-Oriented Computing (ICSOC). 2009. [doi: 10.1007/978-3-642-10383-4_21]

[6] Lizcano D, Soriano J, Reyes M, Hierro JJ. EzWeb/FAST: Reporting on a successful mashup-based solution for developing and deploying composite applications in the upcoming Web of services. In: Proc. of the 10th ACM Int'l Conf. on Information Integration and Web-based Applications & Services (iiWAS 2008). 2008. 24–26. [doi: 10.1145/1497308.1497317]

[7] Samson E. Dynamic composite applications and enterprise mashups with convertigo. A Convertigo White Paper, 2010. http://www.convertigo.com/en/resources-services/white-paper.html

[8] Jugel U. Methodology for service-based interactive application development. Deliverable of ServFace Project, 2010.

[9] Liang QH, Chakarapani LN, Su SYW, Chikkamagalur RN, Lam H. A semi-automatic approach to composite Web services discovery, description and invocation. Int'l Journal of Web Services Research, 2004,1(4):64−89. [doi: 10.4018/jwsr.2004100105]

[10] Li Y, Zou FT, Wu ZD, Ma FY. PWSD: A scalable Web service discovery architecture based on peer-to-peer overlay network. In: Proc. of the Advanced Web Technologies and Applications. Berlin, Heidelberg: Springer-Verlag, 2004. 291−300. [doi: 10.1007/978-3-540-24655-8_32]

[11] Bouillet E, Feblowitz M, Feng HH, Liu Z, Ranganathan A, Riabov A. A folksonomy-based model of Web services for discovery and automatic composition. In: Proc. of the IEEE Int'l Conf. on Services Computing (SCC 2008). 2008. [doi: 10.1109/SCC.2008.77]

[12] Syeda-Mahmood T, Shah G, Akkiraju R, Ivan AA, Goodwin R. Searching service repositories by combining semantic and ontological matching. In: Proc. of the IEEE Int'l Conf. on Web Services. 2005. [doi: 10.1109/ICWS.2005.102]

[13] Wu J, Wu ZH, Li Y, Deng SG. Web service discovery based on ontology and similarity of words. Chinese Journal of Computers, 2005,28(4):595−602 (in Chinese with English abstract).

[14] Canfora G, Di Penta M, Esposito R, Villani ML. A framework for QoS-aware binding and re-binding of composite Web services. Journal of Systems and Software, 2008,81(10):1754−1769. [doi: 10.1016/j.jss.2007.12.792]

[15] Garey M, Johnson D. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H.Freeman, 1979.

[16] Zeng L, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H. Qos-Aware middleware for Web services composition. IEEE Trans. on Software Engineering, 2004,30(5):311−327. [doi: 10.1109/TSE.2004.11]

[17] Ardagna D, Pernici B. Adaptive service composition in flexible processes. IEEE Trans. on Software Engineering, 2007,33(6):369–384. [doi: 10.1109/TSE.2007.1011]

[18] Yu T, Zhang Y, Lin KJ. Efficient algorithms for Web services selection with end-to-end qos constraints. ACM Trans. on Web, 2007,1(1):Article 6. [doi: 10.1145/1232722.1232728]

[19] Liu SL, Liu YX, Zhang F, Tang GF, Jing N. A dynamic Web services selection algorithm with QoS global optimal in Web services composition. Ruan Jian Xue Bao/Journal of Software, 2007,18(3):646−656 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/646.htm [doi: 10.1360/jos180646]

[20] Web services description language. http://en.wikipedia.org/wiki/Web_Services_Description_Language

[21] http://en.wikipedia.org/wiki/Pareto_efficiency

[22] Zheng ZB, Lyu MR. Collaborative reliability prediction for service-oriented systems. In: Proc. of the ACM/IEEE 32nd Int'l Conf. on Software Engineering (ICSE 2010). 2010. 35−44. [doi: 10.1145/1806799.1806809]

[23] Li MQ, Kou JS. Basic Theory and Application of Genetic Algorithms. Beijing: Science Press, 2003 (in Chinese).

[24] Alrifai M, Risse T, Nejdl W. A hybrid approach for efficient Web service composition with end-to-end QoS constraints. ACM Trans. on Web, 2012,6(2):1−31. [doi: 10.1145/2180861.2180864]

[25] Klein A, Ishikawa F, Honiden S. Efficient heuristic approach with improved time complexity for Qos-aware service composition. In: Proc. of the 2011 IEEE Int'l Conf. on Web Services. IEEE Computer Society, 2011. [doi: 10.1109/ICWS.2011.60]

[26] Yoon Y, Ye CY, Jacobsen HA. A distributed framework for reliable and efficient service choreographies. In: Proc. of the 20th Int'l Conf. on World Wide Web (WWW 2011). New York: ACM Press, 2011. 785−794. [doi: 10.1145/1963405.1963515]

[27] Goldman A, Ngoko Y, Milojicic D. An analytical approach for predicting QoS of Web services choreographies. In: Proc. of the 10th Int'l Workshop on Middleware for Grids, Clouds and e-Science (MGC 2012). New York: ACM Press, 2012. [doi: 10.1145/2405136.2405140]

[28] Hu JQ, Li JZ, Liao G. A multi-QoS based local optimal model of service selection. Chinese Journal of Computers, 2010,33(3):526−534 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2010.00526]

[29] Lamparter S, Ankolekar A, Studer R, Grimm S. Preference-Based selection of highly configurable Web services. In: Proc. of the 16th Int'l Conf. on World Wide Web (WWW 2007). New York: ACM Press, 2007. 1013−1022. [doi: 10.1145/1242572.1242709]

[30] Liu DM, Shao ZQ, Yu CZ, Fan GS. A heuristic QoS-aware service selection approach to Web service composition. In: Proc. of the 2009 8th IEEE/ACIS Int'l Conf. on Computer and Information Science. IEEE Computer Society, 2009. [doi: 10.1109/ICIS.2009.76]

[31] Alrifai M, Risse T. Combining global optimization with local selection for efficient QoS-aware service composition. In: Proc. of the 18th Int'l Conf. on World Wide Web. ACM Press, 2009. [doi: 10.1145/1526709.1526828]

[32] Alrifai M, Risse T, Dolog P, Nejd W. A scalable approach for QoS-based Web service selection. In: Proc. of the Service-Oriented Computing ICSOC 2008 Workshops. Springer-Verlag, 2009. 190−199. [doi: 10.1007/978-3-642-01247-1_20]

[33] Zhao JF, Xie B, Zhang L, Yang FQ. A Web services composition method supporting domain feature. Chinese Journal of Computers, 2005,28(4):731−738 (in Chinese with English abstract).

[34] Canfora G, Di Penta M, Esposito R, Villani ML. An approach for QoS-aware service composition based on genetic algorithms. In: Proc. of the 2005 Conf. on Genetic and Evolutionary Computation. Washington: ACM Press, 2005. [doi: 10.1145/1068009.1068189]

[35] Ye Z, Zhou X, Bouguettaya A. Genetic algorithm based QoS-aware service compositions in cloud computing. In: Proc. of the 16th Int'l Conf. on Database Systems for Advanced Applications: Part II. Springer-Verlag, 2011. [doi: 10.1007/978-3-642-20152-3_24]

[36] Jaeger MC, Mühl G. QoS-Based selection of services: The implementation of a genetic algorithm. In: Proc. of the 2007 ITG-GI Conf. on Communication in Distributed Systems (KiVS). 2007. 359−370.

[37] Claro DB, Albers P, Hao J. Selecting Web services for optimal composition. In: Proc. of the 2nd Int'l Workshop on Semantic and Dynamic Web Processes (SDWP 2005). 2005. 32−45.

[38] Klein A, Ishikawa F, Honiden S. Towards network-aware service composition in the cloud. In: Proc. of the 21st Int'l Conf. on World Wide Web. ACM Press, 2012. [doi: 10.1145/2187836.2187965]

[39] Klein A, Ishikawa F, Honiden S. Efficient QoS-aware service composition with a probabilistic service selection policy. In: Proc. of the ICSOC. 2010. 182−196. [doi: 10.1007/978-3-642-17358-5_13]

[40] Liu XZ, Huang G, Mei H. Consumer-Centric service aggregation: Method and its supporting framework. Ruan Jian Xue Bao/Journal of Software, 2007,18(8):1883−1895 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1883.htm [doi: 10.1360/jos181883]

**附中文参考文献**:

[13] 吴健,吴朝晖,李莹,邓水光.基于本体论和词汇语义相似度的 Web 服务发现.计算机学报,2005,28(4):595−602.

[19] 刘书雷,刘云翔,张帆,唐桂芬,景宁.一种服务聚合中 QoS 全局最优服务动态选择算法.软件学报,2007,18(3):646−656. http://www.jos.org.cn/1000-9825/18/646.htm [doi: 10.1360/jos180646]

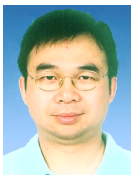
[23] 李敏强,寇纪淞.遗传算法的基本理论与应用.北京:科学出版社,2003.

[28] 胡建强,李涓子,廖桂.一种基于多维服务质量的局部最优服务选择模型.计算机学报,2010,33(3):526−534. [doi: 10.3724/SP.J.1016.2010.00526]

[33] 赵俊峰,谢冰,张路,杨芙清.一种支持领域特性的 Web 服务组装方法.计算机学报,2005,28(4):731−738.

[40] 刘譞哲,黄罡,梅宏.用户驱动的服务聚合方法及其支撑框架.软件学报,2007,18(8):1883−1895. http://www.jos.org.cn/1000-9825/18/1883.htm [doi: 10.1360/jos181883]

**白琳**(1980－),女,河北石家庄人,博士,助理研究员,主要研究领域为网络分布式计算,软件工程.



**魏峻**(1970－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布式计算,软件工程.

**叶丹**(1971－),女,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布式计算,软件工程.

**黄涛**(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布式计算,软件工程.