doi:10.11835/j.issn.1000-582X.2020.11.010

基于模式的个性化服务定制方法

陈春荣[1],何 霆[1],廖永新[1],李海波[1],徐汉川[2]
(1.华侨大学 计算机科学与技术学院,厦门 361021;2.哈尔滨工业大学 计算机科学与技术学院,哈尔滨 150001)

摘要:在服务互联网环境下,大量的用户需求可能被不同的用户同时提出。如何针对用户的个性化需求快速有效地构造服务解决方案是一个值得研究的问题。文中提出了一种基于模式的个性化服务定制方法,通过实验验证了文中所提算法的有效性。首先,利用历史服务请求及服务解决方案识别出的需求模式和服务模式,建立需求模式和服务模式之间的匹配关系。然后,提出一种基于模式的个性化服务定制算法(LPSC)来处理用户的个性化需求。在 LPSC 算法中,对于每个用户的个性化需求,根据用户需求的相似度进行分类构造虚拟需求。再用有限个数的需求模式去替代虚拟需求,通过需求模式与服务模式的匹配关系找到需求模式对应的最佳服务模式集,通过服务模式的组合来产生最终的服务解决方案。

关键词:需求模式;服务模式;个性化需求;模式匹配;服务组合
中图分类号:TP311 **文献标志码**:A **文章编号**:1000-582X(2020)11-099-12

A pattern-based personalized service customization method

CHEN Chunrong[1], *HE Ting*[1], *LIAO Yongxin*[1], *LI Haibo*[1], *XU Hanchuan*[2]
(1.School of Computer Science and Engineering, Huaqiao University, Xiamen 361021, P. R. China;
2. School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, P. R. China)

Abstract: In the service Internet environment, different requirements may be made simultaneously by different users. It is a great challenge to structure composite service quickly and effectively to fulfill personalized requirements. This paper presented a personalized service customization method based on patterns. The requirement patterns and service patterns from historical service request and service solutions were identified, and the matching relationship between the requirement patterns and service patterns were established. An algorithm called LPSC was proposed to deal with personalized requirements by which personalized requirements were classified according to the similarity between requirements and virtual requirements were constructed to replace them. Then each virtual requirement could be replaced by a limited number of requirement patterns. Through the matching relationship established between requirement patterns and service patterns, we could find optimal service pattern set that suited the requirement patterns. Finally, service solutions could be obtained by the composition of service patterns.

收稿日期:2020-07-19
基金项目:国家重点研发计划项目(2018YFB1402500);福建省社会科学规划项目(FJ2020B0033);华侨大学科研启动基金项目(16BS304)。
Supported by the National Key Research and Development Program of China (2018YFB1402500), the Social Science Planning Foundation of Fujian Province(FJ2020B0033), and the Research Initiation Funds of Huaqiao University(16BS304).
作者简介:陈春荣(1996—),男,硕士研究生,主要从事大规模个性化服务定制研究。
通讯作者:何霆,男,教授,博士生导师,(E-mail)xuantinghe@hit.edu.cn。

Experimental results prove the effectiveness of the proposed algorithm.

Keywords: requirement pattern; service pattern; personalized requirements; pattern matching; service compositio

近年来,服务组合一直是服务计算领域学术研究者所关注的一个热点问题。但现在服务组合的研究大都是针对用户提出的单一需求,采用启发式算法[1]来寻找服务解决方案。由于可用的服务越来越多,候选服务的规模越来越大,当多个用户提出不同的用户需求时,传统的服务组合方法[2-3]都是从原子服务出发,单独为每个需求构造解决方案。当用户的需求规模很大时,传统服务组合方法在处理用户的个性化需求时,效率低下。由于在大量的历史服务请求中,一些需求表现出相似性以及部分相似性,利用服务请求及其解决方案的历史数据中挖掘出的模式对服务组合效率的提高有极大的帮助,文中提出了一种基于模式的个性化服务定制方法。同时,为了在服务组合过程中最大限度地满足客户的个性化需求主张,实现服务的个性化定制,考虑不同需求的服务质量约束存在差异以及功能差异来充分体现用户需求的个性化主张。

在服务计算领域,对于模式的研究大多数都集中在服务模式一侧[4-5]。但服务模式并不是单独存在的,而是和一定的应用环境相关联。例如,医生根据自己的先验知识,对不同病人的不同的症状开出不同的处方。利用先验知识,建立需求模式和服务模式的双边匹配关系,这样服务模式可以更加精准地去满足用户需求。需求模式是客户需求的模块化描述,被认为是用户需求中经常出现的连续片段。对于每个用户的个性化需求,可以使用有限个数的需求模式来替代用户的个性化需求。利用先验知识,需求模式可以找到相匹配的服务模式,最后通过服务模式组合产生的解决方案,可以极大地提高服务定制的效率。

在大规模用户需求背景下,一些研究者采用服务网络[6-7]的方法来处理用户的大规模需求,虽然这种方式提高了服务组合的速度,却没有充分考虑用户的个性化主张,同时服务网络在服务系统的运行过程中成本高昂。因此,针对大规模需求背景,如何快速地为用户的个性化需求构造服务解决方案仍然缺乏好的方法。在大服务背景下,一种名为 RE2SEP[8]的软件服务工程新范式(需求-工程两阶段服务工程范式)被提出来处理大规模用户需求。受此启发,文中提出了一种基于模式的个性化服务定制方法,在大规模的用户需求背景的,快速地为用户构造服务解决方案。

1)对于历史服务请求和对应的解决方案中识别出的需求模式和服务模式,通过统计学的方法建立需求模式和服务模式之间的匹配关系。

2)对于不同用户的个性化需求,提出了一种基于模式的个性化服务定制方法(LPSC)。首先,为每个个性化需求找到被需求覆盖的需求模式集,通过需求模式与服务模式之间的匹配关系,找到每个需求模式集对应的服务模式集。在服务模式组合阶段,不通过寻找原子服务来构造解决方案,而是通过服务模式的组合来产生最终的解决方案。

3)基于文中的理论提出了 LPSC 算法,通过与传统的服务组合算法-多目标进化算法[1](EDMOEA)以及基于服务网络的迭代增强方法[5](IEA)相比较,验证了文中算法的有效性。

1 需求模式/服务模式及其映射

1.1 需求模式和服务模式

为了提高用户需求服务定制的效率以及充分体现用户需求的个性化,文中提出了基于模式的个性化服务定制方法。一个用户的需求 r_i 被定义为 $r_i=(r_{i,\text{in}},r_{i,\text{out}},r_{i,q})$,其中,$r_{i,\text{in}}$ 和 $r_{i,\text{out}}$ 分别代表用户提供输入参数集合和期望获得的输出参数集合。$r_{i,\text{in}}$ 和 $r_{i,\text{out}}$ 代表了用户的功能需求,不同的用户需求其输入参数和输出参数不一样。$r_{i,q}$ 代表了用户需求的服务质量约束,不同的用户需求其要求的服务质量约束肯定也存在差异。文中选取的服务质量指标分别为可用性、可靠性以及相应时间,即 $r_{i,q}=(r_{i,q1},r_{i,q2},r_{i,q3})$, $r_{i,q1}$,$r_{i,q2}$,$r_{i,q3}$ 分别代表了服务的可用性,可靠性以及响应时间。

一个需求模式 r_{pi} 被定义为如下形式,$r_{pi}=(r_{pi,\text{in}},r_{pi,\text{out}},r_{pi,q})$,其中,$r_{pi,\text{in}}\neq\varnothing$,$r_{pi,\text{out}}\neq\varnothing$。需求模式 r_{pi} 代表在多个历史需求中抽取出来的频繁连续的需求片段。对于多个需求$\{r_1,r_2,\cdots,r_m\}$,假如,r_{pi} 是从这多个需求识别的需求模式,则对于任意一个需求 r_i,$1\leqslant i\leqslant m$,都有 $r_{pi,\text{in}}\subseteq r_{i,\text{out}}$,$r_{pi,\text{out}}\subseteq r_{i,\text{out}}$,$r_{i,q}\oplus r_{pi,q}$,其中,

⊕表示满足符号，$r_{i,q}\oplus r_{pi,q}$代表需求的服务质量约束程度大于需求模式的服务质量约束程度。一个服务模式 s_{pi}被定义为如下形式 $s_{pi}=(s_{pi,\text{in}},s_{pi,\text{out}},s_{pi,q})$。同理，服务模式是指从大规模的历史服务组合解决方案中所抽取出来的频繁出现的完整/部分服务组合。服务模式可以看作是通过服务组合的业务应用经验来抽取和定义的，是先验知识的载体。

对于模式的挖掘方法，已有大量的研究。使用已有的模式挖掘算法可以很好地获得所需要模式。对于需求模式，在大量的历史服务请求，使用 FP-growth[16] 频繁项集挖掘算法来识别所需要的需求模式。对于服务模式，使用文献[10]中的方法从工作流日志中挖掘出所需要的服务模式。

1.2　需求模式–服务模式的映射

对于从历史记录中挖掘出的需求模式和服务模式，为了充分利用历史先验知识，需要建立需求模式与服务模式两者的映射匹配关系。如表 1 所示，第 1 列表示用户的个性化需求，第 2 列表示从需求中挖掘的所有需求模式，第 3 列代表从需求对应的解决方案中识别的服务模式。对于每一个需求模式 r_{pi}都有一系列的服务模式集跟其相映射，对于每一对(r_{pi},s_{pj})，需要计算需求模式和服务之间的匹配度。主要考虑如下 2 个指标：

1)先验知识。先验知识代表了一个服务模式用来满足一个需求模式的后验概率。这个概率是从历史服务请求和历史服务解决方案中抽象出的经验知识。使用贝叶斯定理可以由下式来表示：

$$p_{ij}=\frac{p(r_{pi})p(s_{pj}/r_{pi})}{p(s_{pj})},\tag{1}$$

其中：$p(s_{pj})=s_+/n$，s_+代表在历史服务解决中服务模式 s_{pj}出现的次数；n 代表历史服务解决方案总个数；$p(r_{pi})=r_+/n$，r_+代表在历史服务请求中，需求模式 r_{pi}出现的次数；$p(s_{pj}/r_{pi})=c_+/r_+$，c_+表示服务模式 s_{pj}与需求模式 r_{pi}在服务历史记录里面共同出现的次数。

2)服务模式与需求模式的相似性。其中，I 代表指示函数，当 I 中条件成立时值为 1，条件不成立时值为 0；⊕代表满足符号。当服务模式的服务质量满足需求模式的服务质量约束时，指示函数值为 1 否则为 0。需求模式与服务模式的输出参数相同的比例越高时，a_{ij}也就越高，代表服务模式 s_{pj}对 r_{pi}更有用。

$$a_{ij}=\frac{I(s_{pi,q}\oplus r_{pi,q})\times|r_{pi,\text{out}}\cap s_{pi,\text{out}}|}{|r_{pi,\text{out}}|},\tag{2}$$

需求模式和服务模式匹配的分数可以由上述 2 个指标得出，w_1和 w_2代表了 2 个指标的权重，两者相加和为 1。

$$s(r_{pi},s_{pj})=w_1p_{ij}+w_2\alpha_{ij}。\tag{3}$$

定义一个需求模式的先验分数为需求模式与每个相映射的服务模式的匹配分数之和，如式(4)所示。一个需求模式的先验分数越高，代表需求模式在历史服务记录中越能被服务模式所满足。s_{pj}表示跟需求模式 r_{pi}相映射的第 j 个服务模式。

$$s_{\text{prior}}(r_{pi})=\sum_{j=1}^{u}s(r_{pi},s_{pj})。\tag{4}$$

表 1　需求模式–服务模式映射实例表

Table 1　Requirement pattern-service patteern mapping instance

用户需求	需求模式	服务模式
r_1	r_{p2},r_{p3}	s_{p1}
r_2	r_{p2},r_{p3}	s_{p1},s_{p2},s_{p3}
…	…	…
r_n	r_{p3},r_{p4},r_{p5}	s_{p1},s_{p5}

2 基于模式的个性化服务定制方法

2.1 概述

针对大规模用户的个性化需求，提出了一种基于模式的个性化服务定制方法(LPSC)。LPSC总体思路如图1所示。在基于模式的个性化服务定制中，首先对用户的个性化需求集，根据需求的相似度进行分类，为每一类中的需求构造虚拟需求来满足同一类别中的所有需求。在需求模式选择阶段，对于构造的虚拟需求，找到被需求覆盖的需求模式集来代替虚拟需求。在服务模式选择阶段，利用需求模式与服务模式之间的双边匹配关系，找到满足需求模式的最佳服务模式集。最后，在服务模式组合阶段，通过服务模式的组合来为用户的个性化需求产生服务解决方案。

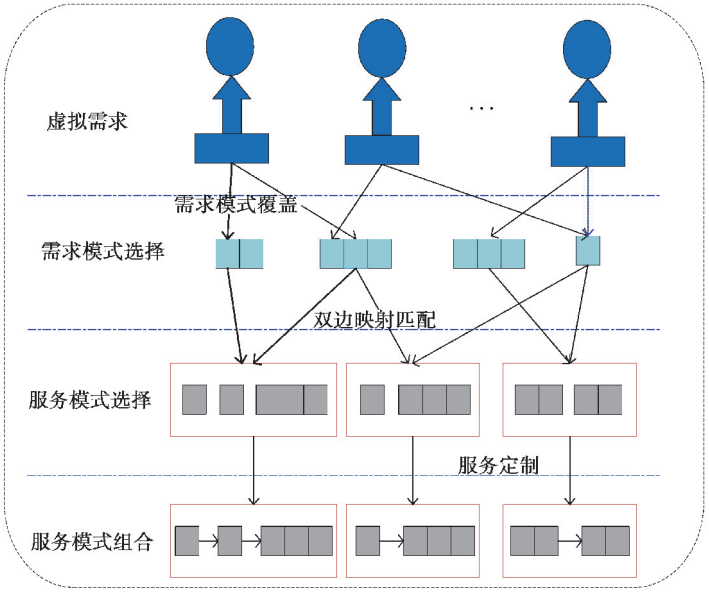

图1 基于模式的个性化服务定制思路

Fig. 1 Pattern-based personalized service customization idea

2.2 需求相似度分类

在大规模用户需求中，一些用户需求总会呈现一定程度的相似性，利用用户需求之间的相似性，对大规模的用户需求进行分类，可以极大地提高服务定制的效率。文中采用 K 最近领(KNN)算法对用户的个性化需求进行分类。利用KNN对用户需求进行分类，要计算用户需求之间的相似度。对于用户的个性化需求主要考虑两个方面的相似度，一个是需求期望实现的功能，另外一个为需求对应的Qos约束。

需求功能相似度定义式为

$$s_f(r_i,r_j)=e_1\times\frac{|r_{i,\text{in}}\cap r_{j,\text{in}}|)}{|r_{i,\text{in}}\cup r_{j,\text{in}}|}+e_2\times\frac{|r_{i,\text{out}}\cap r_{j,\text{out}}|}{|r_{i,\text{out}}\cup r_{j,\text{out}}|},\tag{5}$$

其中，e_1和e_2分别代表了需求提供的输入参数和输出参数的权重，$e_1+e_2=1$，由于希望分类在一起的需求输出的功能参数更接近，在公式(5)中$e_2>>e_1$。

对于2个需求之间Qos约束的相似度，使用皮尔逊相关系数来计算2个需求之间的Qos相似性。

$$s_q(r_i,r_j)=\frac{c(r_i,r_j)}{\sigma_{ri}\sigma_{rj}},\tag{6}$$

其中，$c(r_i,r_j)$，为需求r_i和r_j之间的协方差，定义如式(7)所示。文中r_q为一个三元组，$n=3$；$r_{i,q}'$和$r_{j,q}'$分别为需求r_i和r_j的服务质量约束均值。

$$c(r_i,r_j)=\frac{\sum_{k=1}^{n}(r_{i,qk}-r_{i,q}')(r_{j,qk}-r_{j,q}')}{n-1},\tag{7}$$

其中，σ_{r_i} 以及 σ_{r_j} 分别代表了 $r_{i,q}$ 和 $r_{j,q}$ 的标准差。

$$\sigma_{r_i}=\sqrt{\frac{\sum_{k=1}^{n}(r_{i,qk}-r_{i,q}')^2}{n-1}}, \tag{8}$$

$$\sigma_{r_j}=\sqrt{\frac{\sum_{k=1}^{n}(r_{j,qk}-r_{j,q}')}{n-1}}。 \tag{9}$$

在计算出 2 个需求之间的功能相似度和服务质量约束相似度之后，得到 2 个需求之间的整体相似度，其中 $0\leqslant\lambda\leqslant1$。

$$s_{f,q}(r_i,r_j)=(1-\lambda)s_f(r_i,r_j)+\lambda s_q(r_i,r_j)。 \tag{10}$$

在得到 2 个需求之间的相似度后，利用 KNN 对需求进行分类，K 值的大小会影响分类的结果。对于同一类中的 K 个需求，构造 1 个虚拟需求，使得满足这个虚拟的服务解决方案，满足同一组内的所有需求。

如表 2 所示一个分组，假设 $K=3$。按照最小输入参数，最大输出参数，以及最大 Qos 约束构造虚拟需求 $v_r=(AB,CDEF,0.84,0.94,1.1)$。虚拟需求 v_r 寻找的服务解决方案可以满足这个分组内的所有需求，从而提高服务定制的效率。在算法 1 中详细展示了需求分类算法的伪代码。在算法 1 的 6～7 行将与需求 r_i 相似度最高的 k 个需求分为一组，在第 9 行为这一组需求构造虚拟需求，来代表整组内的所有需求。

表 2　需求分类实例表

Table 2　Requirement classification example table

r	$r_{i,\text{in}}$	$r_{i,\text{out}}$	$r_{i,q1}$	$r_{i,q2}$	$r_{i,q3}$
r_1	A,B	C,D,F	0.82	0.94	1.2
r_2	A,C	C,F	0.80	0.93	1.1
r_3	B,C	C,E,F	0.84	0.93	1.2
r_4	A,B,C	C,D,E	0.81	0.95	1.3

算法 1. 需求分类算法

输入：用户个性化需求集合 R，数值 K

输出：虚拟需求集合 V_R

1) $V_R=\varnothing$，$i=0$

2) for 任意需求 $r_i\in R$

3) for 任意需求 $r_j\in R$ 且 $r_i\neq r_j$

4) calculate $s=s_{f,q}(r_i,r_j)$，$T=T\cup\{s\}$

5) sort(T)　$C=\varnothing$ //需求相似度大小排序

6) while $i<k$ then

7) $C=C\cup r_i$，$R=R/r_i$

8) $v_{ri}=$struct(C)，$V_R=V_R\cup vr$

9) return V_R

2.3　需求模式选择

对于分类后的需求集合 V_R，每一个虚拟需求 v_{ri}，算法遍历挖掘的需求模式集，找到被需求覆盖的需求模式加入到 R_{pi} 中。如果没有任何需求模式被需求所覆盖，则没有任何先验知识可以应用于需求 r_i，在服务模式组合阶段传统的服务组合方法 EDMOEA 将会调用来构造服务解决方案。如果需求模式集 R_{pi} 中需求模式互不重叠则将所有需求模式加入到 R_{Pi} 中。否则，采用遗传算法来寻找最佳的需求模式集合 R_{Pi}。在遗传算法中一条染色体 X 代表了一个需求对应的需求模式集，染色体 X 适应度函数设计考虑下面两个因素。

1)从利益出发考虑,由于互联网上存在大量的功能相同但 Qos 不同的服务,服务质量越高的服务价格也更高。当一个需求模式被需求覆盖时,需求模式的 Qos 约束满足需求的 Qos 约束,在这种情况下采用一种 just-enough 原则,即需求模式集 Qos 约束刚刚满足需求就行。例如,当一个需求要求响应时间为 2 s 时,一个需求模式集对应的响应时间为 2 s,一个为 1 s,这时应该选择 2 s 的需求模式集代表需求模式选择阶段选择的需求模式集。需求对应的各 Qos 属性值与需求模式集对应的 Qos 属性值差值应越小越好。$c_1(X)$应该最小化。染色体 X 代表选择的需求模式集 R_{Pi}。

$$c_1(X)=\left|r_{i,q}-\sum_{r_{pi}\in R_{Pi}} r_{pi,q}\right|。\tag{11}$$

2)需求模式的输出和需求的功能输出重叠程度越大越好,这样可以尽可能减少模式匹配的次数,从而服务定制效率将得到提高。其中,$c_2(X)$应该被最大化;R_{Pi}代表需求模式选择阶段选择的需求模式集;$|r_{i,\text{out}}|$代表了需求 r_i 的总共输出参数个数。

$$c_2(X)=\frac{\left|r_{i,\text{out}}\cap\left(\bigcup_{r_{pi}\in R_{pi}} r_{pi,\text{out}}\right)\right|}{\left|r_{i,\text{out}}\right|}。\tag{12}$$

3)考虑需求模式的先验知识,先验知识越大的需求模式越能被服务模式满足。其中,$c_3(X)$应该被最大化;$|R_{Pi}|$代表了需求模式集中需求模式的个数。

$$c_3(X)=\frac{1}{|R_{Pi}|}\times\sum_{r_{pi}\in R_{Pi}} s_{\text{prior}}(r_{pi})。\tag{13}$$

综合考虑以上因素,遗传算法的适应度函数可以设计如式(14)所示,通过遗传算法可以为每一个需求找到一个最佳的需求模式集合,w_1, w_2, w_3分别代表了 3 个指标的权重,权重值相加为 1。

$$f(X)=w_1c_1(X)+w_2c_2(X)+w_3c_3(X)。\tag{14}$$

在算法 2 中详细展示了需求模式选择的伪代码。算法 2 第 4 行判断每个需求模式与需求的覆盖关系,第 7 行调用遗传算法来寻找最佳的需求模式集。

算法 2.需求模式选择算法

输入:虚拟需求集合 V_R,需求模式集合 RP

输出:大规模需求对应的需求模式集 R_P

1. $R_P=\varnothing, R_{pi}=\varnothing$
2. for 任意需求 $v_{ri}\in V_R$
3. for 任意需求模式 $r_{pi}\in RP$
4. if cover(vr_i, r_{pi})=true then
5. $R_{pi}=R_{pi}\cup r_{pi}$
6. if overlap(R_{pi})≠0
7. $R_{Pi}=\text{SelsectByGA}(R_{pi})$
8. $\text{ELSER}_{Pi}=R_{pi}$
9. $R_P=R_P\cup R_{Pi}$
10. returnR_P

2.4 服务模式选择

在双边映射匹配中,假设每个需求对应的需求模式集 R_{Pi} 相映射的服务模式集为 S_{pi}。为了精准匹配需求模式,对于需求模式集 R_{Pi} 中的每一个需求模式通过式(15)找到匹配分数最高的服务模式 s_{pi} 加入到需求对应的服务模式集 S_{Pi} 中,并将 max 这些服务模式从 S_{pi} 去除掉。

$$s_{pi}=\forall_{s_{pj}\in Spi}\max\, s(r_{pi},s_{pj})。\tag{15}$$

在大多数情况下,由于服务模式和需求模式之间不能百分之百匹配,一个服务模式可能只满足需求模式的一半功能输出,这时需求寻找其它的服务模式来满足需求模式集的功能输出。通过式(16)计算 S_{Pi} 中的每个服务模式与需求模式集 R_{Pi} 的相关度,一个服务模式的相关度越大,表明这个服务模式对需求模式集的作用更大,能够满足更多的需求功能输出。

$$c_{\text{rela}}(s_{pi},R_{Pi})=\frac{\sum_{r_{pi}\in R_{Pi}} s(r_{pi},s_{pi})}{|R_{Pi}|}。\tag{16}$$

在计算相关度时,考虑当前服务与 S_{Pi} 中已选择服务的重叠度。每次迭代中选择与需求模式集相关度最大的服务模式加入到 S_{Pi} 中,同时设置一个阀值 a,将小于阀值的服务模式从 S_{pi} 中去除掉,算法迭代直到 S_{pi} 为空。算法中每个需求对应的服务模式集 S_{Pi} 构成 S_P。

$$c_{\text{rela}}(s_{pi},R_{Pi}) \times = (1 - c_{\text{overlap}}(s_{pi},S_{Pi}))。 \tag{17}$$

定义服务模式之间的重叠度,如式(18)所示,服务模式间重叠度代表了在服务模式的服务质量相同时,服务模式之间输出参数的重叠度。一个服务模式与一个服务模式集合的重叠度定义为服务模式与服务模式集中每个服务模式重叠度和的平均值。

$$c_{\text{overlap}}(s_{pi},s_{pj}) = \frac{s_{pi,\text{in}} \bigcap s_{pj,\text{in}}}{s_{pi,\text{in}} \bigcup s_{pj,\text{in}}} \times I(s_{pi,q} = s_{pj,q})。 \tag{18}$$

在算法 3 中,3～5 行选择了每个需求模式匹配分数最高的服务模式加入到 S_{Pi} 中,算法 6～10 迭代的计算每个服务模需求模式集的相关度,将相关度最大的服务模式加入到 S_{Pi} 中。迭代直到 S_{pi} 为空。

算法 3. 服务模式选择算法

输入:需求模式集 R_P,相映射的服务模式集 S_{pi},阀值 a

输出:服务模式集 S_P

1. for 任意 $R_{Pi} \in \overline{RP}$
2. $S_{pi} = \varnothing, S_{Pi} = \varnothing$
3. for 任意 $r_{pi} \in R_{Pi}$ THEN
4. $s_{pi} = \text{maxscore}(r_{pi},s_{pj}), s_{pj} \in S_{pi}$
5. $S_{Pi} = S_{Pi} \bigcup \{s_{pi}\}, S_{pi} = S_{pi}/\{s_{pi}\}$
6. while $S_{pi} \neq \varnothing$ then
7. rela$= c_{\text{rela}}(s_{pi},R_{Pi}), s_{pi} \in S_{pi}$
8. $S_{Pi} = S_{Pi} \bigcup \{s_{pi}\}$, max $c_{\text{rela}}(s_{pi},r_{pi})$
9. if rela$< \alpha$ then
10. $S_{pi} = S_{pi}/\{s_{pi}\}$
11. $S_P = S_P \bigcup \{S_{Pi}\}$
12. return S_P

2.5　服务模式组合

在服务模式组合中,对于需求分类后的虚拟需求集合 V_R,算法,根据需求的服务质量约束程度进行排序。在为下一个需求构造解决方案时,递归地寻找已经构造的解决方案是否能满足当前的需求。如果不能,则使用需求对应的服务模式集,对服务模式进行组合生成新的解决方案。当服务模式不能满足全部输出功能时,不能被服务模式满足的参数调用传统的服务组合方法来寻找解决方案,并与之前的服务解决方案相结合产生完整的解决方案。

算法 4 第 1 行根据服务质量约束程度对需求进行排序。在算法 5～7 行,用户的个性化需求对应的服务模式集为空,表明没有任何的先验知识可用,直接调用传统的服务组合方法 EDMOEA 来生成解决方案。在算法 12～14 行,对于服务模式不能满足的输出参数,在服务质量的约束下额外的寻找原子服务来满足这些输出参数,并与之前的模式生成的解决方案相结合来获得最终结果。算法中 findCover 函数表示为一个新需求迭代的在已经生成的解决方案寻找解决方案,combine 表示服务模式组合函数。在算法的 18～22 行,去除了服务模式之间输出功能的可能存在的冗余参数。

算法 4.服务模式组合算法

输入:用户个性化需求集 R 以及对应的服务模式集 S_P

输出:用户个性化需求对应的服务解决方案{soln_1.soln_2,…}

1　$R = \text{sort}(R)$

2.for 任意需求 $r_i \in R$

3.　soln$= \varnothing$, subFun$= r_{i,\text{out}}$, soln$=$ findCover(r_i, soln_{i-1})

4.　if soln$= \varnothing$ then

5. if $S_{Pi}=\varnothing$ then
6. $soln_i$=EDMOEA(SubFun,$r_{i,q}$,soln)
7. end
8. for 任意服务模式 $s_{pi}\in S_{Pi}$ 且 $soln^q\oplus r_{i,q}$
9. soln=combine(soln,s_{pi}),subFun=subFun\$s_{pi,out}$
10. if subFun=$\varnothing$,$soln_i$=soln then
11. end
12. if subFun$\varnothing$then
//服务模式不能产生所有输出参数
13. soln=EDMOEA(subFun,$r_{i,q}$\$soln_q$,soln)
14. $soln_i$=soln
15 else
16. $soln_i$=soln
17. end
18. $\Omega=r_{i,out}$
19. for 任意服务模式 $w\in soln_i$ then
20. if $w_{out}\in\Omega$ then
21. $l=l\bigcup\{w\}$,$\Omega=\Omega/w_{out}$,$soln_i=soln_i\backslash w$
22. $soln_i=l$
23. end

3 实 验

实验中的Web服务来自公开的数据集QWS和WS-DREAM。QWS数据集里面包含了服务的Qos属性信息,同时,可以从服务的WSDL文件中提取服务的输入和输出参数。由于没有公开的数据集去收集用户的服务请求,基于Toronto 311模拟生成一个服务请求集。每个请求的服务组合方案可以由EDMOEA算法产生。通过Toronto总共得到了15 670个用户历史需求,通过这些需求以及对应的服务解决方案可以识别出4 783个需求模式与6562服务模式。

1)在第1项实验中,进行了算法敏感度实验,包括4个小实验,分别观察在双边模式挖掘中需求模式支持度s_{rp},服务模式支持度s_{sp},以及需求分类参数K和服务模式选择过程中阀值a对LPSC算法性能的影响。具体的实验参数在表3中详细的进行了说明。n代表了在实验中,所采用的用户需求数量。实验中一个服务解决方案完全满足需求代表服务解决方案在满足需求Qos约束下,满足需求的所有功能输出,图2到图5显示了实验结果。

表3 实验参数列表

Table 3 List of experimental parameters

参数	s_{rp}	s_{sp}	K	a
n	1 000	1 000	1 000	1 000
s_{rp}	—	0.04	0.04	0.04
s_{sp}	0.06	—	0.06	0.06
K	3	3	—	3
a	0.13	0.13	0.13	—

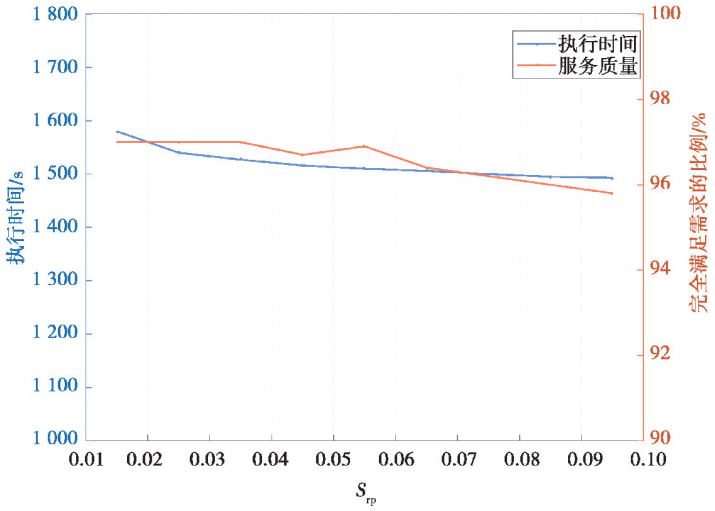

图 2　s_{rp} 对 LPSC 性能影响图

Fig. 2　Impact of s_{rp} on LPSC performance

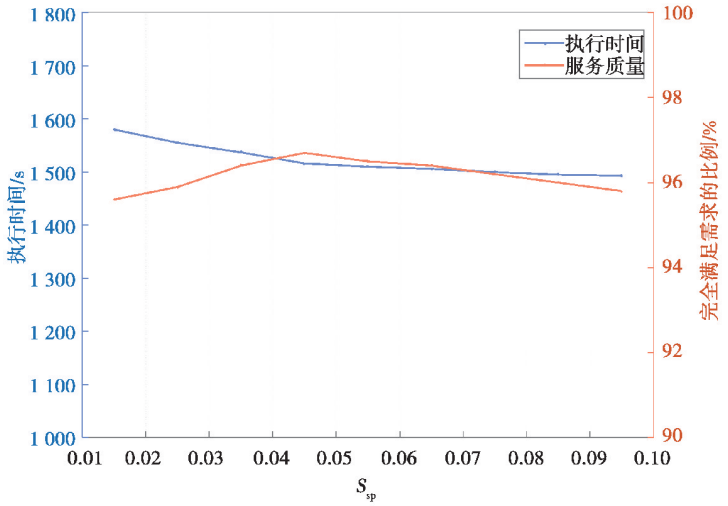

图 3　s_{sp} 对 LPSC 性能影响图

Fig. 3　Impact of s_{sp} on LPSC performance

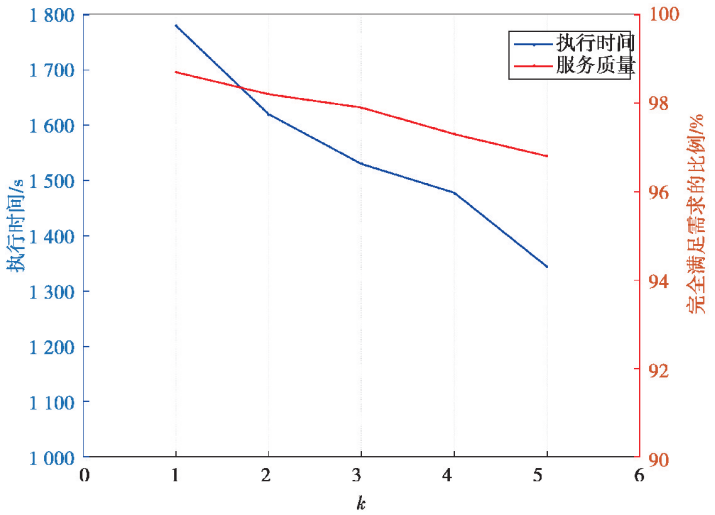

图 4　K 对 LPSC 性能影响图

Fig. 4　Impact of K on LPSC performance

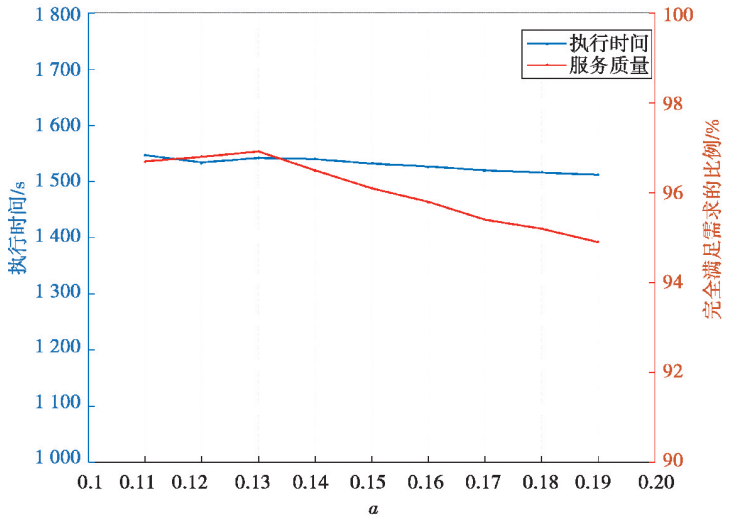

图5 a 对LPSC性能影响图

Fig. 5 Impact of a on LPSC performance

①第1项实验中，观察了需求模式挖掘过程，需求模式支持度的大小对LPSC算法执行效率以及服务解决方案质量的影响。可以发现，s_{rp}的不同会对服务解决方案的服务质量产生一定的影响，而且当s_{rp}达到一定值时，服务解决方案的质量呈现下降趋势。同时可以发现，LPSC的执行时间随着s_{rp}的增大呈现下降趋势，这是由于s_{rp}的增大，会使得被需求覆盖的需求模式个数变小，从而在需求模式和服务模式匹配阶段以及服务模式组合产生的时间花销减少。

②在服务模式支持度的实验中可以发现，随着服务模式支持度的增大，同样算法的执行时间呈现下降趋势，并且逐渐趋于平缓。但是服务解决方案的质量随着s_{sp}的增大刚开始呈现上升趋势，而后在某一点后，逐渐下降。因此，算法在实际运用过程中支持度大小不是越大越好，而是需要在执行效率和质量中寻找一个最佳的平衡点。

③从图4可以发现，在需求分类过程中，K值的选择会极大的影响算法的执行效率以及服务解决方案的质量。随着K值的增大，算法执行时间和服务解决方案的质量都呈现明显下降的趋势。很明显K值越大，用户需求的数量被压缩的越大，从而算法执行效率会下降。但K值的增大，同一组内构造的虚拟需求在功能以及Qos约束上更加严格，服务解决方案的质量会有所降低。

④从图5可以看出，LPSC算法的执行时间随着a的变化会出现轻微的波动。同时阀值a的逐渐变大，LPSC产生的服务解决方案质量出现了降低。这是由于阀值增大时，服务模式选择阶段更多的服务模式被排除在外，从而服务模式组合阶段在服务质量的约束下，不能产生需求的所有功能输出，服务解决方案的质量降低。

2)在第2项实验中，比较在处理不同数量的用户需求时，LPSC，EDMOEA[1]与基于服务网络的迭代增强算法[3](IEA)在执行效率上的表现。从模拟生成的服务请求中选取了2 000个服务请求来进行实验，服务请求个数从0～2 000，间隔为200，服务模式选择阀值a设置为0.15，模式挖掘支持度$s_{rp}=0.06$和$s_{sp}=0.18$，实验结果如图6所示。实验中，计算了3种算法在生成的服务解决方案上的质量结果，如图7所示。

由图6可知，在需求个数较小时，LPSC算法在执行时间上优于EDMOEA，并且随着用户需求规模的扩大，LPSC在执行时间上相对于EDMOEA算法的优势越来越大。对比LPSC和IEA在执行时间上的表现，发现在不同的需求规模时，LPSC算法在执行时间上都略微优于IEA。同时，当用户的需求规模越来越大时，LPSC算法在执行效率的增长率方面也变得越来越低。

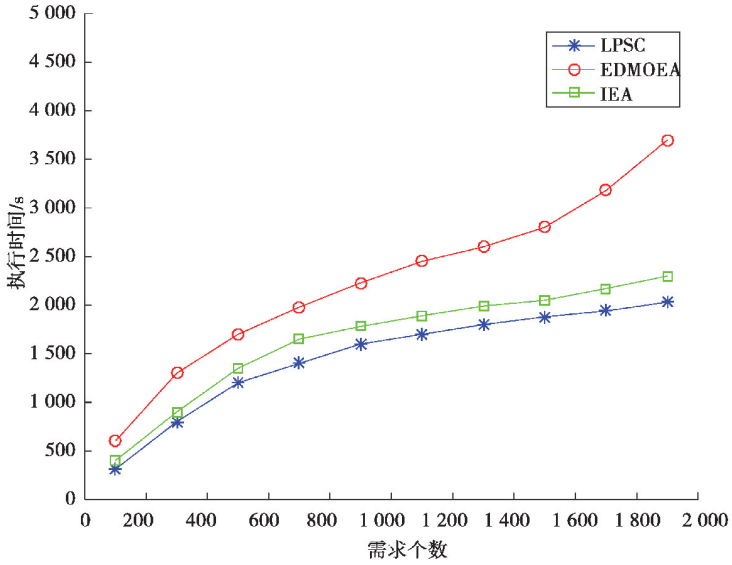

图 6　算法执行时间对比图

Fig. 6　Comparison of algorithm execution time

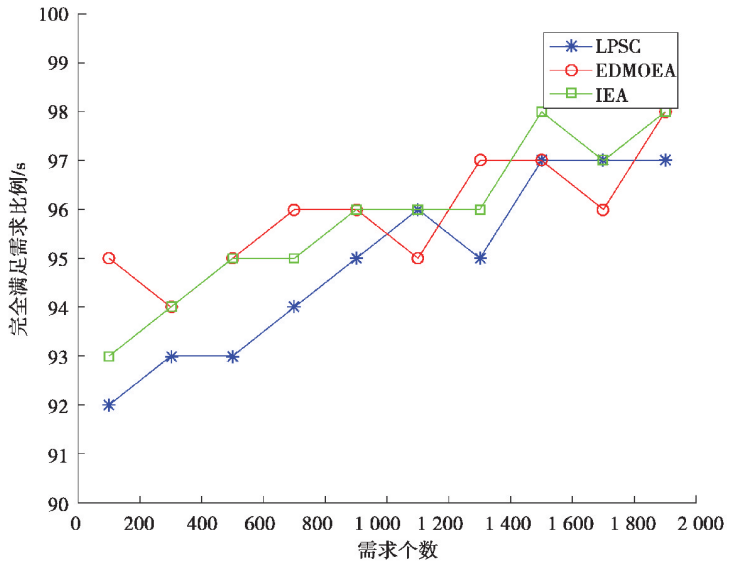

图 7　服务解决方案质量对比图

Fig. 7　Comparision of service solutin quility

由图 7 可知，在最初需求比较少时，LPSC 算法在产生服务解决方案质量方面低于 EDMOEA 和 IEA。但随着需求规模的增大，3 种算法在完全满足需求比例方面，总体上都呈现上升趋势，但 LPSC 算法的上升速度更快，最终当需求个数达到一定规模时，三者产生的服务解决方案质量接近一致。这表明 LPSC 算法在处理大量用户需求时更有优势。

4　结　论

针对用户的个性化需求如何快速定制服务解决方案的问题，提出了一种基于模式的服务定制方法。对于不同用户的不同需求，首先根据用户需求的相似度，对用户需求进行分类。对分类后的每一组需求构造一个虚拟需求来满足组内的所有需求，在服务定制中可以用有限个数的需求模式去替代构造的虚拟需求，同时充分利用先验知识，通过需求/服务模式的映射关系为需求模式集寻找最佳的服务模式集。最后通过服务模式的组合产生服务解决方案，提高了服务解决方案定制的效率。通过实验发现，文中方法在处理少量的用户

需求时，产生的服务解决方案质量略低于EDMOEA和IEA。但当用户的个性化需求规模扩大时，LPSC在服务解决方案质量方面跟其他算法相差无几。同时在用户需求规模比较大时，LPSC相比传统的服务组合(EDMOEA)以及基于网络的迭代增强算法(IEA)在服务定制的执行效率上面都有所提升，从而验证了所提出方法的有效性。

参考文献：

[1] Chen F Z, Dou R L, Li M Q, et al. A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing[J]. Computers & Industrial Engineering, 2016, 99: 423-431.

[2] Shi Y L, Chen X. A survey on QoS-aware web service composition[C]// 2011 Third International Conference on Multimedia Information Networking and Security. Piscataway, NJ: IEEE, 2011: 283-287.

[3] Lin S Y, Lin G T, Chao K M, et al. A cost-effective planning graph approach for large-scale web service composition[J]. Mathematical Problems in Engineering, 2012, 2012: 1-21.

[4] Hu H T, Han Y B, Huang K, et al. A pattern-based approach to facilitating service composition[M]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004: 90-98.

[5] Tut M T, Edmond D. The use of patterns in service composition[M]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002: 28-40.

[6] Wang Z, Xu F, Xu X, et al. Cost-effective service network planning for mass customization of services[J]. Services Transactions on Services Computing, 2014, 2(4): 15-31.

[7] 王忠杰，徐飞，徐晓飞. 支持大规模个性化功能需求的服务网络构建[J]. 软件学报，2014，25(6)：1180-1195.
WANG ZhongJie, XU Fei, XU XiaoFei. Service network planning method for mass personalized functional requirements[J]. Journal of Software, 2014, 25(6): 1180-1195. (in Chinese)

[8] Xu X F, Liu R L, Wang Z J, et al. RE2SEP: a two-phases pattern-based paradigm for software service engineering[C]// 2017 IEEE World Congress on Services (SERVICES). Piscataway, NJ: IEEE, 2017: 67-70.

[9] Huang Z C, Huai J P, Liu X D, et al. Business process decomposition based on service relevance mining[C]// 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. Piscataway, NJ: IEEE, 2010: 573-580.

[10] Perryea C, Chung S. Community-based service discovery[C]// 2006 IEEE International Conference on Web Services (ICWS06). Piscataway, NJ: IEEE, 2006: 903-906.

[11] Feng Z W, Peng R, Raymond K, et al. QoS-aware and multi-granularity service composition[J]. Information Systems Frontiers, 2013, 15(4): 553-567.

[12] Upadhyaya B, Tang R, Zou Y. An approach for mining service composition patterns from execution logs[J]. Journal of Software: Evolution and Process, 2013, 25(8): 841-870.

[13] Zhao Y, Wang S H, Zou Y, et al. Automatically learning user preferences for personalized service composition[C]// 2017 IEEE International Conference on Web Services (ICWS). Piscataway, NJ: IEEE, 2017: 776-783.

[14] Wang Z J, Xu F, Xu X F. A cost-effective service composition method for mass customized QoS requirements[C]// 2012 IEEE Ninth International Conference on Services Computing. Piscataway, NJ: IEEE, 2012: 194-201.

[15] Wang S, Wang Z J, Xu X F. Mining bilateral patterns as priori knowledge for efficient service composition[C]// 2016 IEEE International Conference on Web Services (ICWS). Piscataway, NJ: IEEE, 2016: 65-72.

[16] Han J W, Pei J, Yin Y W. Mining frequent patterns without candidate generation[C/OL]. Proceedings of the 2000 ACM SIGMOD international conference on Management of data-SIGMOD '00. New York, USA: ACM Press, 2000[2020-09-29]. https://doi.org/10.1145/335191.335372

[17] Li T Y, He T, Wang Z J, et al. An approach to iot service optimal composition for mass customization on cloud manufacturing[J]. IEEE Access, 2018, 6: 50572-50586.

[18] Liu J, Chen Y L. A personalized clustering-based and reliable trust-aware QoS prediction approach for cloud service recommendation in cloud manufacturing[J]. Knowledge-Based Systems, 2019, 174: 43-56.

(编辑 詹燕平)