第44卷 第2期 西 安 交 通 大 学 学 报 Vol. 44 №2
2010年2月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Feb. 2010

QoS敏感的服务组合动态配置研究

杨怀洲[1,2]，李增智[1]
(1. 西安交通大学计算机系统结构与网络研究所，710049，西安；2. 北京邮电大学
网络与交换技术国家重点实验室，100876，北京)

摘要：针对静态配置的Web服务组合系统无法适应组件服务QoS的动态变化，以及对易错环境缺乏自适应性和不能反映系统不断演进特征的问题，提出了一种QoS敏感的服务组合系统动态配置方法. 通过扩展Petri网对服务依赖关系进行建模，形成一个形式化的系统配置方案；利用无循环依赖关系验证算法和最终状态合法性验证算法验证了系统配置方案的正确性；提出了一种最优配置选择算法以选取具有最优QoS的服务配置. 仿真实验对比了动态配置、静态配置和随机配置对用户服务请求满意度的影响，结果表明，所提建模方法和相应算法能大幅改善组合服务的QoS.

关键词：Web服务组合；优化配置；Petri网；自适应系统
中图分类号：TP311 **文献标志码**：A **文章编号**：0253-987X(2010)02-0025-06

Research on QoS-Aware and Dynamic Configuration of Web Services Composition System

YANG Huaizhou[1,2], LI Zengzhi[1]
(1. Institute of Computer Architecture & Networks, Xi′an Jiaotong University, Xi′an 710049, China; 2. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: The Web services composition system with static configuration lacks of adaptive capacity to both the QoS variability of component services and the failure-prone environment. The static configuration can not reflect the dynamic evolution process of Web services composition when the number of component services increases continually. A method of QoS-aware and dynamic configuration for Web services composition is presented. The functional dependency relationship of Web services is described formally via an extended Petri net to form a configuration scheme. The configuration scheme is verified by two algorithms, which are used to confirm there are no loop dependency relationship and illegal end state in the configuration scheme. An optimal configuration selection algorithm is used to search the configuration with best QoS from the configuration scheme. Simulation experiments are conducted to compare the effects on the satisfaction rate of user request for service among the dynamic, static and random configurations. The results show that the QoS of composition service can be improved greatly by using the proposed modeling method and algorithms.

Keywords: Web services composition; optimal configuration; Petri net; adaptive system

Web服务组合系统不同于传统组件系统的一个显著特点是：针对某些系统功能，存在大量可供系统构建者选择的、功能相似或等价的组件服务. 可供组合的组件服务集合被称为Web服务社区[1]，目前的一个热门研究主题是如何基于服务社区进行服务组合系统配置，以满足不同用户的服务需求[2]. 服务

收稿日期：2009-07-28. 作者简介：杨怀洲(1970—)，男，博士生；李增智(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(60673170)；北京邮电大学网络与交换技术国家重点实验室资助项目.

依赖图(Service Dependency Graph,SDG)是一种常用的服务配置表达方法[3]. Xiong 等人扩展了 SDG,提出了一种基本 Petri 网表达的服务功能依赖配置网[2];Hwang 等人以服务操作为粒度,利用有限状态机对服务配置进行了建模[1]. 此外,包括遗传算法[4]、中间件[5]等理论与技术也应用于改善组合服务的 QoS. 但是,以上研究仍然存在一些不足,主要体现在:①系统需要适应组件服务 QoS 的动态变化;②系统需要增强容错性;③系统需要灵活配置,以适应服务社区不断演进的过程.

为了解决这些问题,本文提出一种 QoS 敏感的服务组合动态配置方法. 该方法分 3 个步骤完成:①利用扩展 Petri 网对服务社区中的组件服务功能依赖关系进行建模,形成基础的服务配置计划;②通过无循环依赖关系验证算法,和最终状态合法性验证算法,验证系统配置方案的正确性;③提出一种 QoS 敏感的最优配置选择算法,从配置计划中选择出具有最佳 QoS 的系统配置.

1 服务配置计划扩展 Petri 网建模

服务配置计划建模主要反映了服务社区中组件服务间的相互依赖关系,配置计划中包含了多个可能的服务组合方式. 如图 1 所示的 SDG 反映了组件服务功能的相互依赖关系.

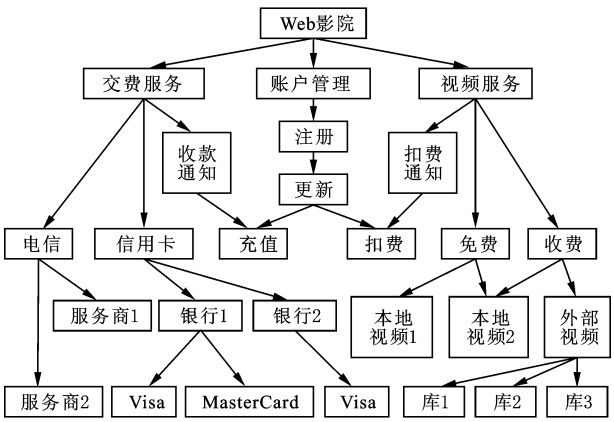


图 1 Web 影院服务组合系统服务依赖图

服务依赖关系可分为 4 种基本类型,分别表达为如图 2a~图 2d 所示的 Petri 网结构:①服务需求经 WS1 处理后被分解为可由 WS2、WS3 分别处理的子需求;②经 WS1 和 WS2 处理后的服务需求可由 WS3 统一进行处理;③服务需求可由 WS1 或 WS2 处理;④服务需求按照 WS1、WS2 的顺序分步处理. 如图 2e 所示,服务需求经 WS1、WS2 处理后又产生了同样的服务需求. 这种服务间的循环依赖关系不允许出现在服务配置计划模型中.

为了便于表达服务依赖关系和它们相应的 QoS,应先将基本 Petri 网的五元组(P, T, F, W, M_0)予以扩展,然后形成服务配置计划网(Service Configuration Scheme Net,SCS-Net).

定义 1 一个扩展 Petri 网(EPN)是一个七元组(P, T_i, T_q, F, W, M_0, Q),元素描述如下.

(1)P、T 分别为位置和变迁有限集,$T=T_i\cup T_q$, $P\cup T\neq\varnothing$, $P\cap T=\varnothing$, T_i 是空变迁,T_q 是实变迁.

(2)$F\subseteq(P\times T)\cup(T\times P)$是弧集合.

(3)$\mathrm{dom}(F)\cup\mathrm{cod}(F)=P\cup T$(无孤立元素)

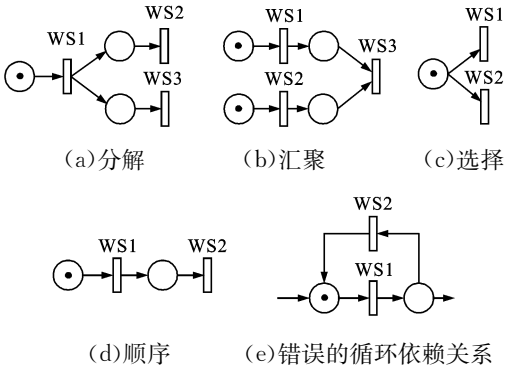
$\mathrm{dom}(F)=\{x\mid\exists y:(x,y)\in F\}$

$\mathrm{cod}(F)=\{x\mid\exists y:(y,x)\in F\}$

(4)$W:F\to N^+$是弧权函数,N^+是正整数.

(5)$M_0:P\to N$ 是初始标识,N 是自然数.

(6)$Q:T_q\to R^+$是 QoS 函数,R^+是正实数.



(a)分解 (b)汇聚 (c)选择

(d)顺序 (e)错误的循环依赖关系

图 2 服务依赖关系

定义 2 一个服务配置计划网(SCS-Net)是一个 EPN,且满足如下条件:①存在唯一输入位置 p_i,且 $M_0(p_i)=1$,$M_0(p_i)$ 表示配置任务,$\forall p'\in P/\{p_i\}$, $M_0(p')=0$;②存在唯一输出位置 P_o,配置任务到达 P_o 后不需要再进行处理;③网中没有环形结构,无循环依赖关系;④对于 $\forall t_q\in T_q$,服务社区中存在一个组件服务及其相应 QoS 值与 t_q 对应.

定义 3 一个有效配置是 SCS-Net 中的一个变迁集合 T_c,且满足以下条件:① $\exists T'\subset T$, $T'=\{t_1, t_2,\cdots,t_n\}$,$T'$ 构成一个有限变迁实施序列 $\sigma=M_0t_1M_1t_2\cdots t_nM_n$,$T_c=T_q\cap T'$,$T_c$ 是 T'的实变迁子集合;②对于 σ 中的标识 M_n,如果 $M_n(p)\neq0$,则 $p\in P_o$.

图 1 所示的 Web 影院系统同时提供了电信和信用卡 2 种交费方式,本地视频 2 服务可以同时提

供免费和收费视频服务，本地视频 1 服务和多个可选外部收费视频服务可以使系统具有均衡负载和容错的能力. 系统配置计划可以表达为图 3 所示的 SCS-Net，系统可能的配置为 $\{t_1,\cdots,t_6,t_7,t_9,t_{10},t_{14}\}$.

当服务社区中增加了新的组件服务时，通过分析新的组件服务与已有组件服务的功能依赖关系，新增组件服务便可加入至相应的 SCS-Net，以反映不断演进的服务组合过程.

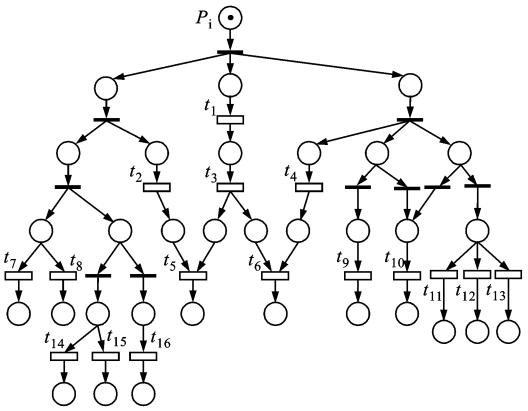


图 3　Web 影院服务配置计划网模型

2　服务配置计划验证

根据定义 2 和定义 3 验证服务配置计划的正确性，涉及以下 2 个步骤：①无循环依赖关系验证；②模型最终状态合法性验证.

2.1　无循环依赖关系验证

通过图 4 所示算法可以检查一个 SCS-Net 中是否具有循环结构. 该算法利用递归来实现，并通过一次深度优先搜索完成验证，因此算法复杂度为 $O(n)$，$n=|P|+|T|+|F|$，n 为 SCS-Net 节点和弧的数量之和. SCS-Net 的无环性是本文后续算法能够正确运行的基础.

2.2　最终状态合法性验证

首先给出 SCS-Net 的合法最终状态定义.

定义 4　SCS-Net 的一个合法最终状态是一个满足如下条件的标识 M：①$M\in[M_0\rangle$，M 属于 SCS-Net 的可达标识集；②不存在 $t\in T$，$M[t\rangle M'$，M 下无可实施变迁；③$\forall p\in P$，如果 $M(p)\neq 0$，则 $p\in P_o$.

SCS-Net 的状态代表了配置任务被处理的特定阶段，定义 4 中条件①和条件②限定了最终状态. 最终状态表示配置任务无法或无需再由服务社区中的组件服务进一步处理. 定义 4 中条件③检查了最终状态的合法性，合法的最终状态表示配置任务完全并正确地予以处理，而在一个 SCS-Net 中一般存在多个最终状态. 利用图 5 所示算法可以对一个 SCS-Net 中的所有最终状态进行合法性验证. 该算法通过一次广度优先搜索完成验证，因此算法复杂度为 $O(\eta)$，η 为 SCS-Net 状态数量.

```
Input：一个SCS-Net
Output：SCS-Net中是否具有循环结构
Begin
  HasLoop=False
  Q=∅      '祖先节点集合
  For Each n∈P∪T  n.visited=False '所有节点标为未访问
  Explore(p_i)
  Return HasLoop
End
Procedure Explore(n)
Begin
  n.visited=True
  Q=Q∪{n}
  For Each n'∈n·
    If n'.visited=False Then
      Explore(n')
    Else
      If n'∈Q Then
        HasLoop=True
        终止搜索
      End If
    End If
  Next
  Q=Q\{n}
End
```

图 4　无循环依赖关系验证算法

```
Input：一个无环SCS-Net
Output：SCS-Net中是否具有非法最终状态
Begin
Leagal=Ture
  S={M_0}  'S为已验证状态集合
Inject(Q,M_0)   'Q为一个状态队列
Do While(Q≠∅)
  M=Eject(Q)  'M为一个临时状态变量
  设H为状态M下可实施变迁集合
  If H≠∅ Then
    For Each t∈H
      M[t>M'   't实施产生状态M'
      If M'∉S Then
        Inject(Q,M')
        S=S∪{M'}
      End If
    Next
  Else
    If ∃p∈P,M(p)≠0∧p∉P_o Then
      Legal=False
      Exit Do
    End If
  End If
Loop
Return Leagal
End
```

图 5　SCS-Net 最终状态合法性验证算法

3　最优服务配置的动态选择

一个验证后的服务配置计划 SCS-Net 中包含了多种功能可行的服务配置方式. 一个组件服务通常具有多种 QoS 指标，如通用 QoS（响应时间、吞吐

量、可靠性和费用)、Web 服务 QoS(可用率、安全性等). 设 $Q(t)$ 代表组件服务的某一种 QoS 值,t 代表组件服务的实变迁,即 $t\in T_q$. $Q(t)$ 值小服务质量好的指标有响应时间、费用等;$Q(t)$ 值越大服务质量越好的 QoS 指标有吞吐量等,该类指标值设为负值;对于失效的组件服务,$Q(t)=\infty$. $Q(t)$ 的值可以利用中间件通过实测的方式获得[5]. 设 $\delta(t_{Set})$ 为计算服务配置 QoS 的函数,t_{Set} 为实变迁集合. 当仅用于比较不同服务配置 QoS 的优劣时,$\delta(t_{Set})$ 可以设为加法操作[2],此时 $\delta(t_{Set})=\sum_{t\in t_{Set}}Q(t)$.

图 6 所示算法适用于一个 SCS-Net 中搜索 QoS 最优服务配置. 由于 SCS-Net 能够反映一个组件服务被选后对后续组件服务选择的影响,因此最优服务配置选择算法是体现服务依赖关系和最优 QoS 双重约束下的服务配置选择方法. 本文算法的目的可以简述为:自动搜索一个从 SCS-Net 初始状态到最终状态的 QoS 最优实变迁实施序列. 不同于常用的最短路径搜索算法,如 Dijkstra 算法,本文算法所处理的问题和实现过程有以下特点:①以 SCS-Net 为输入,而不是输入 SCS-Net 的状态可达图;②SCS-Net 具有一个输入状态和一个或多个最终状态,只需要得到从初始状态 M_0 到最终状态的最优 QoS 路径,不需要获得从 M_0 到所有可达集状态的最优路径;③无需维护一个整体搜索节点空间,也无需在每一个搜索步骤后对所有节点的搜索信息进行更新,因此减小了算法的时空消耗;④采用深度优先搜索,以便于在不满足最优 QoS 要求时终止当前路径的搜索,从而提高了搜索效率.

定理 1 SCS-Net 中不存在状态循环.

证明 采用反证法,如果存在 $M't_1M_1t_2M_2\cdots t_nM'$ 的状态循环,则变迁序列 $\sigma=t_1t_2\cdots t_n$ 可循环执行,因此构成了服务循环依赖关系,这违反了 SCS-Net 定义 2 的条件.

定理 1 是最优服务配置选择算法的主要设计依据. 由于不存在状态循环,因此从任意可达状态开始,每次选择当前状态下任意一个可实施的变迁点火,状态转移过程都将到达一个最终状态. 算法用堆栈 S 实现深度优先搜索,用压栈表示存储一个分支路径搜索点,用弹栈表示从一个路径分支点继续搜索. 一个变迁实施序列堆栈 σ 被用来记录从 M_0 到当前搜索点的路径信息,路径信息与搜索过程同步. 一个实变迁集合 t_{Set} 被用来同步记录当前路径信息 σ 中的实变迁. 当搜索达到一个最终状态时,此时的

```
Input: 一个验证后的SCS-Net
Output: 一个QoS最优服务配置
初始化:
Structure StateStru  '定义一个状态结构
    State        '一个SCS-Net状态
    Transition   '引发该状态的变迁
    Depth        '该状态位于搜索路径中的深度
End Structure
Dim M_Stru As StateStru '定义状态结构变量M_Stru
M_stru.State=M_0
M_stru.Transition=Null
M_stru.Depth=0
S=∅          'StateStru状态结构元素堆栈
σ=∅          '变迁实施序列堆栈
Layer=-1     '搜索所位于的层数
ConQoS=∞     '服务配置的QoS初设值
t_Set=∅      '实变迁集合
Begin
  Push M_stru Into S
  Do While (S≠∅)
    M_stru=PoP(S)
    If M_stru.Depth ≤ Layer Then '同层或上层新分支搜索
      For i=1 To Layer-M_stru.Depth+1
        t=PoP(σ) '变迁实施序列回退,t为变迁变量
        If t∈T_q Then t_Set=t_Set\{t}
      Next
    End If
    Layer=M_stru.Depth
    t=M_stru.Transition
    If t≠Null Then Push t Into σ
    If t≠Null∧t∈T_q Then t_Set=t_Set∪{t}
    If σ(t_Set)≥ConQoS Then Goto NL '进入下一个循环
    M=M_stru.State '为一个临时状态变量
    设H为状态M下可实施变迁集合
    If H≠∅ Then
      Q≠∅  'Q为一个临时状态结构元素集合
      For Each t∈H
        SameState=False '重复状态标志
        M[t>M' 't实施产生状态M'
        For i=1 to |Q|
          If Q(i).State=M' Then '变迁产生相同状态
            SameState=True
            If Q(Q(i).Transition)>Q(t) Then
              Q(i).Transition=t '选择QoS最小的变迁
            End If
            Exit For
          End If
        Next
        If SameState=False Then
          M_stru.State=M'
          M_stru.Transition=t
          M_stru.Depth=Layer+1
          Q=Q∪{M_stru}
        End If
      Next
      For Each q∈Q Push q Into S
    Else
      If δ(t_Set)<ConQoS Then
        OpCon=t_Set 'OpCon代表最优配置的变迁集合
        ConQoS=δ(t_Set)
      End If
    End If
  NL: Loop
  Return OpCon
End
```

图 6 最优服务配置选择算法

t_{Set} 构成了一种服务配置方式. 当搜索过程中发现当前路径的服务质量低于搜索到的最优配置服务质量时,停止沿当前节点搜索,转而由新的分支节点继续

搜索.

SCS-Net 的可达状态图并不一定是一个纯粹的树型结构. 虽然不存在循环状态结构，即不存在回边，但有可能存在横跨边. 横跨边可使当前状态节点转移到一个已访问过的非祖先节点，由此造成需要重复搜索部分已搜索过的路径以记录一个完整的服务配置路径信息. 因此，在最坏情况下，也就是在所有可能的服务配置都被完全搜索到的情况下，如果不存在横跨边，算法复杂度为 $O(\eta)$，η 为 SCS-Net 状态数量；如果每个搜索路径都遭遇横跨边，且重复搜索包括了大多数状态节点时，算法复杂度接近但不会大于 $O(n^2)$. 在横跨边引起的重复搜索量并不大的情况下，算法复杂度接近 $O(n)$. 由以上分析可知，本文算法的执行效率很高.

4　配置策略性能对比实验

通过实验对比了以下 3 种配置策略对系统性能的影响：①静态配置，即以固定组件服务完成服务配置；②随机配置，即随机选择服务配置路径，在组件服务调用超时的情况下随机改选另一路径；③动态配置，即本文算法的配置方式. 可在考虑完整服务配置路径 QoS 的基础上，进行最优服务配置选择. 策略①和策略②不需要监视组件服务 QoS；策略②和策略③需要服务配置计划的支持；策略②无 QoS 敏感性，但具有一定的容错性.

仿真实验以图 3 所示模型为基础，以响应时间作为 QoS 实验指标. 假设组件服务响应时间的变化符合正态分布，(μ,σ) 对应数学期望和标准差. $t_1 \sim t_{16}$ 的响应时间分布设为 (5,1)，(3,1)，(7,2)，(3,1)，(4,1)，(4,1)，(16,8)，(18,7)，(21,4)，(25,6)，(23,5)，(26,5)，(22,4)，(17,3)，(15,3)，(16,3). 实验中，静态配置选择数学期望值最小的配置方式：$\{t_1,\cdots,t_6,t_7,t_9,t_{13},t_{15}\}$. 首先，利用 Matlab 软件中的 randn() 函数对 $t_1 \sim t_{16}$ 生成 100 组共 100×16 个正态分布的仿真 QoS 数值，每一组数据代表了某一时刻服务社区中各组件服务的响应时间，数据以文本文件方式存储以供实验程序调用. 然后，针对每一组组件服务 QoS 数值，运行相应算法获得随机配置和动态配置方式下的系统配置. 最后，分别计算静态、随机和动态 3 种服务配置 QoS，得到 100 组共 100×3 个数值，每一组数据代表了某一时刻系统在不同配置方式下的 QoS.

为了比较 3 种服务配置对服务请求的影响，需设立 2 个指标：①服务配置 QoS 阈值，该指标反映了用户请求组合服务时希望获得的最低服务质量，当某一时刻服务配置 QoS 超过阈值时，系统不能满足用户的 QoS 需求；②服务请求满足率，该指标反映了在接受一定数量服务请求时，系统能够满足用户最低服务质量要求的比率，即最低服务质量要求被满足的服务请求数量除以服务请求总数.

假设服务配置 QoS 阈值为 100，所得实验结果如图 7 所示. 可以看出：动态配置具有最好的服务请求满足率，其均值为 89.36%，标准差为 5.95%；静态配置下的服务请求满足率很低，其均值为 46.61%，标准差为 10.27%；随机配置下的服务请求满足率最差，其均值为 31.41%，标准差为 11.17%.

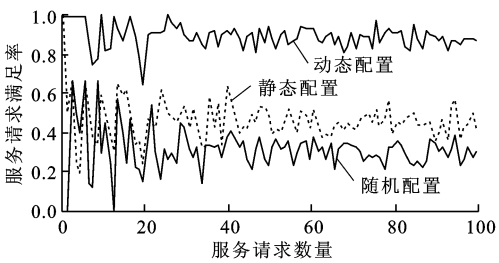


图 7　不同配置下的服务请求满足率

图 8 所示的实验结果反映了不同服务配置 QoS 阈值下 3 种配置方式的服务请求平均满足率的变化. 随着阈值的增加，也就是随着用户对系统 QoS 的要求越来越宽松，3 种配置方式的服务请求平均满足率均单调递增，最后都能够 100%满足服务请求. 但是，从 3 种配置曲线可以明显看出，在任意阈值下，动态配置的服务请求平均满足率均大于或等于其他 2 种配置方式，说明它能最好地满足用户对系统的 QoS 需求.

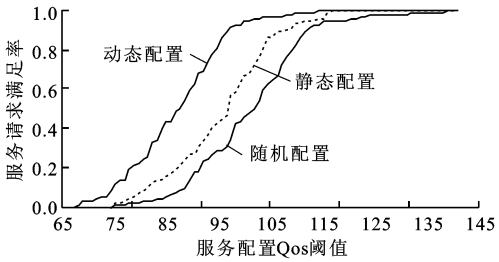


图 8　不同配置 QoS 阈值对服务请求满足率的影响

5　结论与展望

静态配置的 Web 服务组合系统难以适应组件服务 QoS 动态变化的情况，系统缺乏容错性，也不

能够反映系统随可用组件服务数量增长而不断演进的固有特征. 本文通过服务配置计划扩展 Petri 网建模、配置计划验证、最优服务配置选择这 3 个环节,实现了 QoS 敏感的服务组合动态配置,有效地解决了以上问题. 通过不同配置策略下的性能对比实验,说明了动态配置在实际应用中的优势,验证了所提算法的正确性和实用性. 未来工作将围绕服务配置计划的自动生成与更新进一步开展深入的研究.

参考文献:

[1] HWANG S, LIM E, LEE C, et al. Dynamic Web service selection for reliable Web service composition [J]. IEEE Trans on Services Computing, 2008, 1(2): 104-116.

[2] XIONG P, FAN Y, ZHOU M. QoS-aware Web service configuration [J]. IEEE Trans on Systems, Man, and Cybernetics, 2008, 38(4): 888-895.

[3] DMYTRO Z. A Petri net-based approach for automated goal-driven Web service composition [J]. Simulation, 2007, 83(1): 33-63.

[4] KOBTI Z, ZHIYANG W. An adaptive approach for QoS-aware Web service composition using cultural algorithms [C]// The 20th Australian Joint Conference on Artificial Intelligence. Berlin, Germany: Springer, 2007: 140-149.

[5] LIU Y, TRUONG S, CHEN S, et al. Composing adaptive Web services on COTS middleware [C]// 2008 IEEE International Conference on Web Services. Los Alamitos, CA, USA: IEEE Computer Society, 2008: 377-384.

(编辑 苗凌)