基于 QoS 多目标优化的组合服务执行路径的选择

李 滢[1,2],童维勤[1],亢朝峰[3],支小莉[1]

(1. 上海大学计算机科学与技术学院,上海 200072)
(2. 南京晓庄学院数学与信息技术学院,江苏 南京 211171)
(3. 中兴通讯,江苏 南京 200012)

[摘要] 针对目前启发式算法求解组合服务执行路径中存在的不足,将组合服务执行路径上节点服务的 QoS 多目标优化问题转化为蚂蚁从巢穴到食物之间的最短路径选择问题,提出基于改进的蚁群算法——蚁群系统作为优化工具,并提出了用于屏蔽执行路径结构的虚拟抽象服务的概念,设计了算法中的局部与全局更新规则,将多目标优化的 QoS 约束参数融入 dij 参数的定义中,最后通过仿真实验验证了算法的可行性与优越性.

[关键词] QoS,多目标优化,Pareto 解集,蚁群系统,虚拟抽象服务
[中图分类号] TP311 [文献标志码] A [文章编号] 1001-4616(2012)03-0125-09

The QoS-Based Multi-Objective Optimization Algorithm of the Composite Service Execution Path

Li Ying[1,2], Tong Weiqing[1], Kang Chaofeng[3], Zhi Xiaoli[1]

(1. School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)
(2. School of Mathematics & Information Technology, Nanjing Xiaozhuang University, Nanjing 211171, China)
(3. ZTE Corporation, Nanjing 200012, China)

Abstract: In this paper, after discussing the limitations of heuristic algorithm using for composite service execution path optimization, an optimization algorithm of Ant Colony System based on Ant Colony Algorithm is proposed. The algorithm regards the optimization problem as the shortest path solving problem of ant foraging from nest to food. In order to overcome the lack of Ant Colony System only using to solved sequential organization path, a concept of virtual abstract service was put forward innovatively to shield the different path structures, then local and global updating rules had been redesigned and the constrained QoS parameters were also into the definition of parameter. Finally, through simulation and comparing with basic Ant Colony Algorithm experimental verified the usability and superiority of the algorithm.

Key words: QoS, multi-objective optimization, Pareto set, Ant Colony System, virtual abstract service

组合服务执行路径是由可执行具体服务按照一定的功能次序以及满足约定的服务 QoS 约束链接起来的,因此组合服务的执行路径直接决定了服务组合的可用性,对于服务组合的整体评价的意义重大. 已有报道,服务模板的构建已经完成了组合服务执行路径的功能次序的组织[1],因此本文重点考虑组合服务执行路径中涉及到的具体服务在满足约定的 QoS 基础上的优化问题. Web 服务的多个 QoS 优化目标之间通常是相互制约、相互矛盾的,是一个典型的多目标优化问题. 基于服务 QoS 的组合服务执行路径的多目标优化,实际上就是基于某个服务模板在众多功能相同、服务质量不同的具体服务中选出满足服务 QoS 约束的最优组合.

当前较流行的组合服务执行路径的多目标优化算法主要是采用启发式算法[2-5,8],如遗传算法、粒子群优化算法、蚁群算法等. 遗传算法存在收敛较慢且最优解质量不高的问题;粒子群优化算法在实际应用中的可靠性还有待提高;蚁群算法的产生背景决定了它适合于求解离散空间中的组合优化问题[13]. 最近,

收稿日期: 2012-07-27.
基金项目: 国家自然科学基金(60573109)、上海市科技创新行动计划项目(11511500200).
通讯联系人: 李 滢,博士研究生,研究方向: 面向服务的计算. E-mail bxyzly@163.com

基于蚁群算法对服务组合进行的研究是一个研究热点,我们以前的研究采用了改进后的蚁群系统算法,但在路径 d_{ij} 的设计上显示并没有实现多目标优化. Peng 等[5] 采用了蚁周系统模型,它是基本蚁群算法的3种常用模型中的1种,因此决定了该算法不可避免地存在不易收敛的缺点、找寻最优解成功率低的问题. 此外,一些研究和本文的研究目标一致,但仅仅重点考虑了服务的选择优化问题,而没有兼顾到服务的执行路径[10,11,13]. 因此本文采用改进的基本蚁群算法,即蚁群系统(Ant Colony System),来实现组合服务执行路径的多目标优化,即 MOACS 算法.

MOACS 算法基本思路是: 首先构建组合服务路径选择问题的数学模型; 其次提出了虚拟抽象服务的概念,淡化了路径结构带来的影响,满足了蚁群系统算法自身的应用特点; 接着定义了算法实现的相关规则与关键参数; 最后根据设计的实验数据表执行算法并获取 Pareto 最优解集,即组合服务的执行路径的 Pareto 解.

1　组合服务执行路径的多目标优化问题的描述

1.1　优化问题的相关数学描述

如图1所示,在 Web 服务组合中,一个完整的 Web 组合服务是由多个 Web 抽象服务(WS_i)组成,每个 Web 抽象服务又对应多个具体服务(ws_{ij}). 这些具体服务来自于不同的提供者,它们具有相同的调用接口和功能,但具有不同 QoS 值. 因此基于 QoS 角度,每个组合服务模板(WT)又包含多个执行路径,如何找到具有最佳 QoS 组合的执行路径 WE_i 是本文研究的重点.

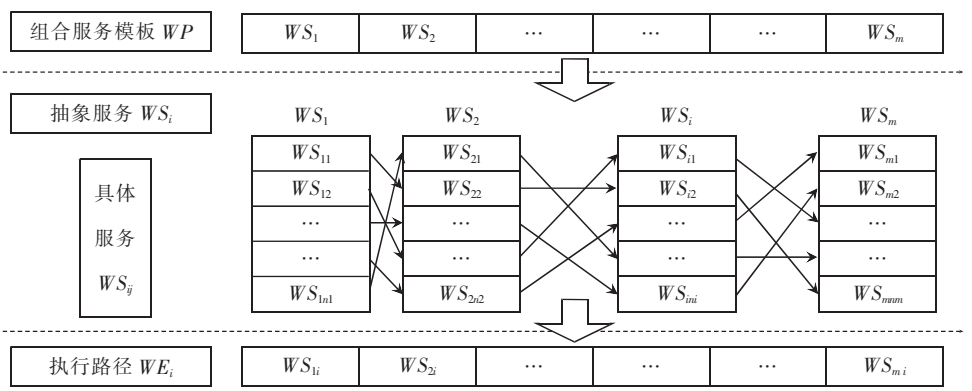


图1　服务组合结构示意图

Fig.1　Architecture of service composition

首先,作如下定义:

定义1　抽象服务(WS),是对一组功能相同、接口相似的具体服务的抽象,可以用一个三元组 WS_i = (Name,Input,Output) 表示,其中 Name 是服务功能描述的描述,Input 是抽象服务的输入参数集合,Output 是抽象服务的输出参数集合.

定义2　具体服务(ws),是指可以通过 UDDI 发现、并获得的可执行服务. 具体服务可以用一个四元组 ws_i = (Name,Input,Output,QoS) 来表示,其中 Name 是具体服务功能描述的描述,Input 是具体服务的输入参数集合,Output 是具体服务的输出参数集合,QoS 是具体服务的服务质量. 服务的 QoS 属性包括服务执行时间和成本、精度、安全、授权、认证、(事务的)完整性、可靠性、可伸缩性和可用性等. 本文选取服务执行时间、执行费用、信誉等级和可靠性4个指标作为 QoS 选择的依据,用四元组 QoS = (T,C,RP,RL) 来表示,其中 T 是服务执行时间(Time),C 是服务执行费用(Cost),RP 是服务的信誉等级(Reputation),RL 是服务的可靠性(Reliability).

定义3　组合服务模板(WP),是为了满足用户的某个功能需求,按照顺序选取的一组抽象服务组合,表示为 $WP = \{WS_1, WS_2, \cdots, WS_m\}$,其中 m 为服务模板中包含的抽象服务数目. 执行路径是针对服务模板中的每个抽象服务分别选择出对应的具体服务,这些具体服务组成一条执行路径,表示为 $WC = \{ws_1, ws_2, \cdots, ws_m\}$,其中 m 为执行路径中具体服务的数目. 对于 WC 中的服务 ws_i 归属于 WP 中的抽象服务 WS_i,

即 $ws_i \in WS_i$,则称 WC 是 WT 的一条执行路径,表示为 $WC \in WP$.

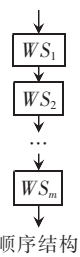
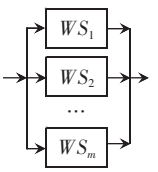
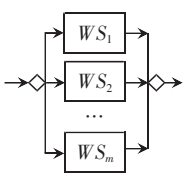
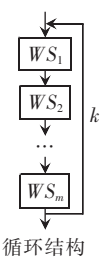
定义 4　组合服务的 QoS 是指完成某个服务模板的服务质量[7],它跟该组合方案中执行路径上的具体服务的 QoS 有着密切的关系,表示为

$$F(WP) = (f_t(WP), f_c(WP), f_{rp}(WP), f_{rl}(WP)). \tag{1}$$

组合服务执行路径的结构指的就是节点服务之间的结构关系,主要有 4 种结构: 顺序结构、并行结构、选择结构、循环结构. 对应这 4 种不同的结构,组合服务的 QoS 计算方法如表 1 所示.

表 1　执行路径结构以及对应的 QoS 计算公式

Table 1　List of execution path and QoS computing formula

路径结构	QoS 计算公式	路径结构	QoS 计算公式
 顺序结构	$F(WP) = \begin{cases} f_t(WP) = \sum_{i=1}^{m} f_t(ws_i) \\ f_c(WP) = \sum_{i=1}^{m} f_c(ws_i) \\ f_{rp}(WP) = \sum_{i=1}^{m} \frac{f_{rp}(ws_i)}{m} \\ f_{rl}(WP) = \prod_{i=i}^{m} f_{rl}(ws_i) \end{cases}$	 并行结构	$F(WP) = \begin{cases} f_t(WP) = \max(f_t(ws_i)), i = 1,\cdots,m \\ f_c(WP) = \sum_{i=1}^{m} f_c(ws_i) \\ f_{rp}(WP) = \sum_{i=1}^{m} \frac{f_{rp}(ws_i)}{m} \\ f_{rl}(WP) = \min(f_t(ws_i)), i = 1,\cdots,m \end{cases}$
 选择结构	$F(WP) = \begin{cases} f_t(WP) = \sum_{i=1}^{m} s_i f_t(ws_i) \\ f_c(WP) = \sum_{i=1}^{m} s_i f_c(ws_i) \\ f_{rp}(WP) = \sum_{i=1}^{m} \frac{s_i f_{rp}(ws_i)}{m} \\ f_{rl}(WP) = \prod_{i=i}^{m} s_i f_{rl}(ws_i) \end{cases}$	 循环结构	$F(WP) = \begin{cases} f_t(WP) = k\sum_{i=1}^{m} f_t(ws_i) \\ f_c(WP) = k\sum_{i=1}^{m} f_c(ws_i) \\ f_{rp}(WP) = \sum_{i=1}^{m} \frac{f_{rp}(ws_i)}{m} \\ f_{rl}(WP) = \prod_{i=i}^{m} f_{rl}(ws_i) \end{cases}$
式中 s_i 服务被选中的概率		k 为循环次数	

定义 5　前向关系,指某个组合模板 $WP = \{WS_1, WS_2, \cdots, WS_m\}$ 中抽象服务 WS_i 将要调用的抽象服务为 WS_j,则称抽象服务 WS_j 为抽象服务 WS_i 的前向服务,表示为 $WS_i \rightarrow WS_j$. 组合模板中的抽象服务的前向关系集合组成组合方案的前向关系矩阵,表示为: $R(WP) = \{r_{ij}\}$,其中:

$$r_{ij} = \begin{cases} 1, \text{当 } WS_i \rightarrow WS_j; \\ 0. \end{cases} \tag{2}$$

定义 6　虚拟抽象服务是指由几个有前向关系的抽象服务组合成一个新的抽象服务,记为 $WS_i^I = (\text{Name}, \text{Input}, \text{Output})$,其中 Name 是可以取服务名称的并集,Input、Output 是跟虚抽象服务中的抽象服务相关. 提出虚抽象服务的概念,主要将不同控制结构的服务组合问题转化为基于递归的顺序结构服务组合问题,使优化问题更加适用于 MOACS 算法求解.

如图 2 所示,经过转换后,变为顺序结构的组合模型,在转换后的模型中,WS^I 和 WS 具有相同的属性,因此可以将虚抽象服务直接看作抽象服务. 如果没有特别指明,本文后续部分的抽象服务包括虚抽象服务.

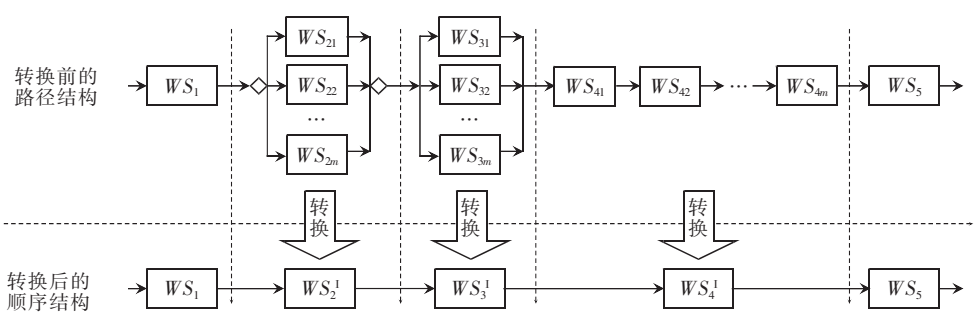


图 2　虚拟抽象服务概念图

Fig.2　Virtual abstraot service diagram

1.2 多目标优化的概念

在定义4中,某一执行路径中多个节组成服务的QoS优化目标之间通常是相互制约的,它的最优解不是一个确定的解,而是一组均衡解,这组解被称为Pareto最优解集. 如何在最优解集中求出一组Pareto最优解供决策者选择是本文的研究目标.

定义7 根据多目标问题定义,基于QoS的多目标Web服务组合问题的数学模型可以定义为:

$$\min(y = f(WE) = (f_t(WE), f_c(WE))). \tag{3}$$

$$\text{subject to} \begin{cases} f_{rp}(WE) \geqslant RP \\ f_{rl}(WE) \geqslant RL \end{cases} \tag{4}$$

where $WE = (WE_1, WE_2, \cdots, WE_{nm}) \in D$.

其中WE组成$m * n$维决策(解)空间,$f(WE)$为目标函数,y组成2维目标空间,式(3)、(4)为约束条件,D为解空间名称.

定义8 Pareto支配关系,对于决策变量u、v满足:

$$\forall f_t(u) \leqslant f_t(v), f_c(u) \leqslant f_c(v)$$
$$\wedge \exists f_t(u) < f_t(v), f_c(u) < f_c(v)$$

则称$f(u)$支配$f(v)$,记为$f(u) > f(v)$.

定义9 Pareto最优解,对于决策变量$x \in D$,若不存在$x^* \in D$,使得$x^* < x$,则称x是D上的Pareto最优解.

定义10 Pareto最优解集(非劣解集),非劣解集P是由解空间D中没有被其他解支配的解构成的集合,即$P = \{x \in D \mid \neg \exists x^* \in D, x^* < x\}$.

基于QoS的多目标服务组合问题,就是针对服务组合模板$WP = \{WS_1, WS_2, \cdots, WS_m\}$,如何在服务执行路径$WC = \{ws_1, ws_2, \cdots, ws_m\}$中找出满足定义9的Pareto最优解集. 求解Pareto最优解集的算法有很多,本文采用精英保留机制中的外部存储器思想. 设置一个外部存储器保存每一代群体中的优良个体,即非支配解,并在算法运行结束后,将外部存储器中的解作为对问题最优解集的近似.

2 *MOACS*算法的设计

2.1 关于蚁群系统

蚁群算法本身所具有的特性,使得该算法具有很强的发现较好解的能力. 但是基本蚁群算法也存在容易出现停滞现象、收敛缓慢等不足. 为了克服这些缺陷、提高算法性能,本文采用蚁群系统作为求解工具. 蚁群系统(Ant Colony System,ACS)是由Dorigo和Gambardella在1996年提出的[9],它在蚂蚁系统的基础上主要做了3个方面的改进[10]:

1) 合理利用关于问题的先验知识. 其状态转移规则为更合理地利用新路径和利用关于问题的先验知识提供了方法. 与蚂蚁系统不同,蚁群系统没有采用仅仅依赖于概率进行路径选取的随机比例规则,而是使用了基于不同决策规则的伪随机比例规则,因此可以同时实现合理利用问题的先验知识以及有倾向性的路径探索.

2) 提高了算法的搜索效率. 改变全局更新规则对系统中所有蚂蚁都更新的方法,每次循环后只对最优蚂蚁所走的路径进行信息素的增强,其他路径由于挥发机制信息素逐渐减少,这就增大了最优路径和最差路径在信息素上的差异,从而使得搜索行为能够很快地集中到最优路径附近,从而提高了算法的搜索效率.

3) 求解过程中应用局部信息素更新规则. 蚁群系统的信息素更新规则与以往只在每次循环后对所有路径进行一次全局更新不同,它是在蚂蚁构造路径的同时就进行局部信息素的更新,然后在每次循环后再对路径进行一次全局更新.

根据前面服务组合问题的定义,我们可以把起点抽象服务WS_1看作是人工蚁群的巢穴,把终点抽象服务WS_m看作为要寻找的食物,将具体服务ws_{ij}看作路径上的节点,则此服务组合问题就可以转化为人工蚁群从巢穴到食物间路径的寻优问题.

2.2　初始化禁忌表

定义一个类 R_WS,用来表示服务前进表中的某个节点. iRelation 表示前向关系,指针 pR 不为 NULL 表示,该节点的抽象服务为虚抽象服务,该指针指向虚服务包含的抽象服务.

```
struct{
    int iRelation;
    R_WS * pR;
} R_WS
```

按照 $R(WP)=\{r_{ij}\}$ 生成的前向关系矩阵来生成服务前进表数组 allowed. 在该数组中,每个元素均用类 R_WS 表示. 只有存在前向关系的抽象服务,路径才可达,iRelation 记为 1,否则记为 0. 通过递归完成 allowed 的构建. 初始化禁忌表 tabu ‖ ‖ = allowed ‖ ‖.

2.3　状态转移公式

在 MOACS 算法中,蚂蚁采用伪随机比例规则选择下一个节点,即 t 时刻位于具体服务 ws_i 的第 k 只蚂蚁将以概率 q_0 转移到具体服务 ws_j,其中服务 ws_j 使 $(\tau_{ij}(t)^{\alpha}*\eta_{ij}^{\beta})$ 最大,这是根据先验知识选择最好的. 否则蚂蚁将以概率 $(1-q_0)$、按照式(9)选择下一个具体服务. 这意味着转移概率按照概率选择移动的目标节点服务,增强了搜索的多样性,以避免算法过早陷入搜索停滞.

$$P_{ij}(t)=\begin{cases}\max(\tau_{ij}(t)^{\alpha}*\eta_{ij}^{\beta}), & \text{若 } q\leqslant q_0,\ j\in \text{allowed},\text{且 } i \text{ 到 } j \text{ 存在直接路径;}\\ \dfrac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t)}{\sum\limits_{s\in \text{allowed}}\tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)}, & \text{若 } q\geqslant q_0,\ j\in \text{allowed},\text{且 } i \text{ 到 } j \text{ 存在直接路径;}\\ 0, & \text{否则.}\end{cases} \tag{5}$$

其中 $\tau_{ij}(t)$ 表示 t 时刻具体服务 ws_i 和具体服务 ws_j 间的信息素,$\eta_{ij}(t)$ 表示具体服务 ws_i 和具体服务 ws_j 间的能见度,α 是信息素因子,β 是启发式因子,q 是在 [0,1] 均匀分布的随机数,q_0 是一个参数$(0\leqslant q_0\leqslant 1)$,它的大小决定了利用先验知识与探索新路径之间的相对重要性.

2.4　局部更新规则

信息素的局部更新规则是指在寻找一个最优组合服务路径过程中,在所有的蚂蚁都完成一次转移后进行,使用下式对信息素更新:

$$\tau_{ij}(t+1)=(1-\rho)*\tau_{ij}(t)+\rho*\tau_{ij}(0). \tag{6}$$

其中,ρ 为信息挥发系数,$0<\rho<1$. 局部更新规则的应用使得相应的信息素轨迹量逐渐减少,可以有效地避免算法过早地收敛于同一路径. 上式中信息素初始值定义为:

$$\tau_0=(nL_{nn})^{-1}=n*\frac{1}{0.5\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}d_{ij}}=\frac{2n}{\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}d_{ij}}. \tag{7}$$

上式中 n 是具体服务的个数,L_{nn} 是由最近的邻域启发产生的一个路径长度,d_{ij} 的计算方法可由下式获得:

$$d_{ij}=\begin{cases}f_t(ws_j)+f_c(ws_j), & \text{若 } r_{ij}=1,\\ f_{rp}(ws_j)>RP, \quad f_{rl}(ws_j)>RL; & \\ \infty, & \text{否则.}\end{cases} \tag{8}$$

上式中,i、j 代表 2 个抽象服务中的具体服务编号,RP 表示信誉等级的最小值,RL 表示可靠性等级的最小值.

2.5　全局更新规则

在 MOACS 中,全局更新不再用于所有的蚂蚁,而只是对每次循环中最优的蚂蚁使用. 信息素更新规则可以用下式来表示:

$$\tau_{ij}(t+1)=(1-\rho)*\tau_{ij}(t)+\rho*\Delta\tau_{ij}^{gb}(t). \tag{9}$$

其中 $\Delta\tau_{ij}^{gb}(t)=\dfrac{1}{L_{gb}}$,$L_{gb}$ 是从寻找活动开始到当前为止全局最优路径长度. 在上式中规定只有那些属于全局最优路径边上的信息素才会得到增强. 式中 L_{gb} 可用以下公式求得:

$$L_{gb} = 0.5\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}. \quad (10)$$

2.6 *Pareto* 最优解集构造

建立外部存储器 P,用来存储最优解,设置解集大小为 M. 被初始化为空集,当一只蚂蚁找到一条执行路径 wp_i 后,将 wp_i 放入外部存储器 P 中. 如果 P 中最优解数目小于 M,直接将 wp_i 最为 Pareto 最优解放入 P;如果 P 中最优解数目大于、等于 M,如果存在 $wp_j \in P$,使得 $wp_i \prec wp_j$,则将 P 中 wp_j 删除,将 wp_i 加入 Pareto 最优解 P 中,否则丢弃 wp_i.

2.7 算法的实现

算法实现的具体步骤:

```
Begin
  初始化外部存储器 P ← ∅;
  设置 ACS 相关参数: α、β、q0、ρ、Na(蚂蚁数目)、N_Smax(最大循环次数),N_S ← 0;
  生成算法关键数据: allowed、τij(t)、ηij(t);
  while (N_S < N_Smax) do
  将 Na 只蚂蚁随机地放置于起始服务上;
  初始化信息素初值 τij(0)、Δτij(t);
  for k = 1 to m do// 蚂蚁 k ← k + 1 开始寻找路径
    获取禁忌表;
    while(一条服务路径未寻找完成) do
    根据状态转移规则选定下一个具体服务;
    移动到下一个服务节点;
    信息素局部更新;
  End while
  Call calculate Sum Of Distance() 计算本次服务执行的 QoS,并保存到 Lk,k = {1,2,3,…,m};
      清空禁忌表;
  End for
  获取最佳路径 wpi: Lgb = minLk;
  信息素全局更新;
  If (P < P_MAX) do
    将本次最优执行路径 wpi 放入外部存储器 P 中;
  Else
      While(wpj ∈ P) do
      If(wpi ≺ wpj) do
        将 P 中 wpj 删除;
       将 wpi 加入 Pareto 最优解 P 中;
     End if
   End while
  End if
 End while
End.
```

3 实验与结果分析

3.1 实验环境与结果

为了验证算法的有效性,我们进行了仿真实验. 实验环境为:主机配置 Pentium(R) Dual-Core CPU E5300 @ 2.6 GHz 2.59 GHz,2 G 内存,操作系统为 Windows XP,算法采用 VC ++ 语言,使用 Microsoft

Visual C ++ 2008 Express Edition 开发. 鉴于目前没有相关的标准平台和标准测试数据集，实验数据(服务组合的任务数、Web 服务及其 QoS 属性) 均采用模拟的方式生成.

1) 服务组合的仿真模型构造

图 3 中每个方框表示 1 个抽象服务,每个抽象服务包含 10 个具体服务,方框中的数字即为具体服务的 ID. 该模型由 10 个抽象服务{SW_1,SW_2,…,SW_{10}} 组成,服务模型中包含顺序结构: $SW_2 \rightarrow SW_3$、$SW_4 \rightarrow SW_5$、$SW_6 \rightarrow SW_8$; 选择结构: $SW_1 \rightarrow SW_2$、SW_4、SW_6; 并行结构: SW_7、SW_9. 循环结构在实际的服务组合中应用比较少,本模型没有包含. 在该模型中起始服务为 $SW1$,包含 10 个具体服务,服务 ID 为 0 ~ 9. 结束服务为 SW_{10},也包含 10 个具体服务. 通过 MOACS 算法,要找到从 SW_1 到 SW_{10} 中的以具体服务为节点的执行路径的 Pareto 最优解集.

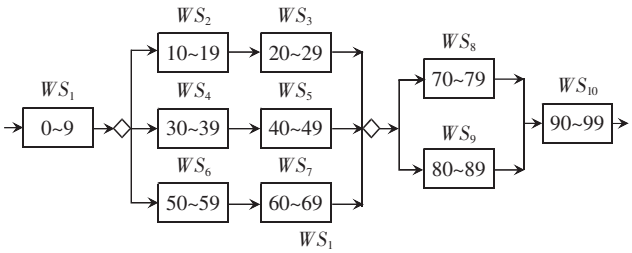


图 3　仿真服务组合模型结构

Fig.3　Test structure of simulation services composition model

2) 生成仿真数据

采用下面算法随机生成 100 个具体服务的 QoS 参数:

```
for( int i = 0; i < 100; i ++)
{
    wsQos [i] [0] = random( 1000) ;
    wsQos [i] [1] = random( 200) ;
    wsQos [i] [2] = random( 10) ;
    wsQos [i] [3] = random( 50) ;
}
```

其中 wsQos [100] [4] 为存储具体服务 QoS 的数组,执行时间是 0 ~ 1 000 的一个随机整数,执行费用是 0 ~ 200 的一个随机整数,信誉等级是 0 ~ 10 的一个随机整数,可靠性是 0 ~ 50 的一个随机整数. 表 2 列出了随机算法生成的前 2 个抽象服务(ID 为 0 ~ 19) 的具体服务的 QoS.

表 2　部分具体服务仿真数据表

Table 2　Part simulation QoS data of specific service

抽象服务名称	具体服务	执行时间 /ms	执行费用(标准单位)	信誉等级	可靠性	抽象服务名称	具体服务	执行时间 /ms	执行费用(标准单位)	信誉等级	可靠性
WS1	ws0	295	162	3	12	WS2	ws10	667	99	5	44
	ws1	41	67	4	0		ws11	703	11	2	33
	ws2	169	124	8	8		ws12	673	64	1	11
	ws3	962	64	5	45		ws13	253	68	7	44
	ws4	281	27	1	41		ws14	662	157	7	9
	ws5	995	142	7	36		ws15	723	141	9	28
	ws6	391	4	2	3		ws16	316	35	0	42
	ws7	292	182	1	16		ws17	288	106	0	42
	ws8	718	95	7	26		ws18	264	48	6	5
	ws9	771	138	9	12		ws19	890	129	0	0

3) MOACS 算法主要参数设置参见表 3.

4) 仿真结果

通过采用前面的服务组合模型及相关参数设置,运行 MOACS 算法,结果如图 4 所示.

通过实验仿真,最终找到了 Pareto 最优解集,从单个目标来看,这些 QoS 解有可能不是最优的,但同时从服务执行时间、执行费用、信誉等级、可靠性 4 个维度来看,这些解是非劣的,而且解的分布是均匀的,算法是有效可行的. 用户可以根据实际的需要选择最合适的 Pareto 解,其他没有被选用的 Pareto 解可以作为备选路径,在组合服务流程执行过程发生意外时启用.

表3　MOACS参数设置表

Table 3　MOACS parameter setting

参数名称	数值
信息素因子 α	2
启发因子 β	5
q_0	0.1
信息挥发系数 ρ	0.1
蚂蚁数目 N_a	10
最大循环次数 $N_{S_{\max}}$	2 000
信誉等级的最小值 RP	5
可靠性等级的最小值 RL	10

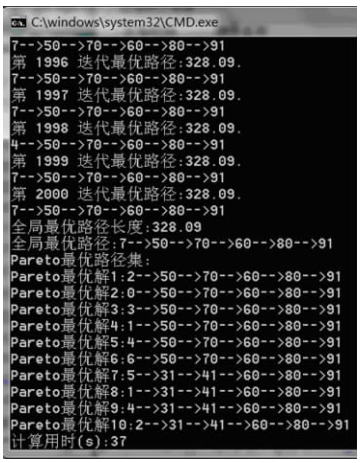


图4　算法运行结果截屏

Fig.4　The operation results of MOACS

3.2　与MOACA算法的比较

MOACA是以基本蚁群算法为工具的多目标优化算法，本文用MOACA作为比较对象，得出如图6的仿真结果.

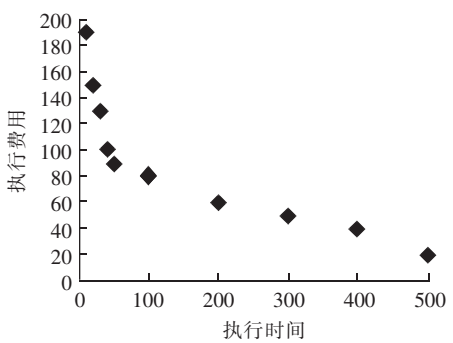


图5　MOACS算法的Pareto解集

Fig.5　Pareto solution set of MOACS

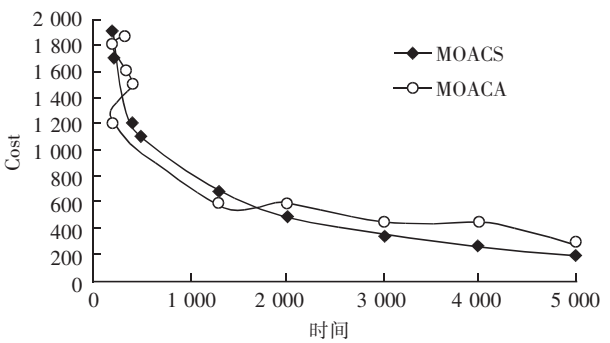


图6　MOACS与MOACA的Pareto解集对比图

Fig.6　Pareto solution set based on MOACS comparing with MOACA

由算法所得的Pareto分布图可以看出，和MOACA算法相比，本文算法所得结果能够很好地逼近Pareto前沿，而且得到的解分布更加均匀. 并且和MOACA算法来求解多目标QoS服务组合问题所得的结果相比，在分布区间上可以获得相对较多的解. 在求解过程中，由于使用外部存储器存储Pareto最优解和全局更新规则对系统中所有蚂蚁都更新的方法，在迭代较少次数的情况下就能够得到理想解，搜索盲目性减少，效率明显提高. 同时克服了MOACA基本蚁群算法存在的容易出现停滞现象、收敛缓慢的不足.

4　结束语

本文设计并实现了用于组合服务执行路径多目标优化的MOACS算法，克服了基本蚁群算法搜索效率低的不足，通过改进设定局部与全局更新规则，提高了Patreto解的质量. 该算法有以下特点:

1) 提出了一种新的生成Pareto非支配集的方法，该方法由于采用了相对较优的解组成，提高了算法的收敛速度.

2) 相比传统的MOACA算法，采用了多目标蚁群系统算法(MOACS)来求解多目标的有效解，与蚂蚁系统相比，蚁群系统在处理优化问题上有了很大的改进，它能更好地避免优化算法出现过早停滞的现象.

3) 在MOACS算法中，提出了前向关系矩阵来表示服务间的选择方向: 优化了转移概率、局部更新规则、全局更新规则的算法.

4) 针对服务组合中，存在的不同控制结构的问题，本文提出了虚抽象服务的概念，将不同控制结构的服务组合转化为基于递归的顺序结构服务组合问题. 在MOACS算法中，采用二维链来表示这种关系.

本文下一步的研究工作主要基于 2 个方面: 1) 在参数选择对优化结果的影响及结果是否分配均匀、逼近 Pareto 前沿方面进行进一步的论证; 2) 将服务 QoS 的动态性参数融入到算法中来,实现实时优化.

[参考文献]

[1] Li Y, Hu Y, Tong W Q,et al. The service template composition method based on separation of concerns [C] // Proc of the 2010 IEEE International Conference on Progress in Informatics and Computing. Shanghai, 2010: 1 057-1 059.

[2] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm [C] // Giannakoglou K, Tsahalis D T, Périaux J, et al. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. Berlin: Springer-Verlag, 2002: 95-100.

[3] Coello C A, Pulido G T, Lechuga M S. Handing multiple objectives with particle swarm optimization [J]. IEEE Transaction on Evolutionary Computations, 2004,8(3): 256-279.

[4] Wang Y, Dai G P, Hou Y R. Dynamic methods of Trust-Aware composite service selection [J]. Chinese Journal of Computers,2009,32(8): 1 668-1 675.

[5] Peng X M,He Y X,Zhu B J. Application of Ant Colony Algorithm in Web services composition [J]. Computing Engineering, 2009, 35(10): 182-188.

[6] Li S Y. Ant Colony Algorithms With Applications [M]. Harbin: Harbin Institute of Technology Press, 2004.

[7] Nahrstedt K, Wichadakul D, Xu D. Distributed QoS compilation and runtime instantiation [C] // Proceedings of the 8th IEEE/IFIP International Workshop on Quality of Service. Pittsburgh, 2000: 198-207.

[8] Martin M, Chopard B, Albuquerque P. Formation of an ant cemetery: Swarm intelligence of statistical accident [J]. Future Generation Computer Systems, 2002, 18: 893-901.

[9] Duan H B. Ant Colony Algorithms: Theory and Applications [M]. Beijing: Science Press, 2005.

[10] Zeng L Z, Benatallah B. QoS-aware middleware for Web services composition [J]. IEEE Transactions on Software Engineering, 2004, 30(5): 311-327.

[11] Liu S L, Liu Y X, Zhang F, et al. A dynamic Web services selection algorithm with QoS global optimal in Web services composition [J]. Journal of Software, 2007, 18(3): 646-656.

[12] 刘波,杨路明,雷刚跃. 融合粒子群与蚁群算法优化 XML 群体智能搜索 [J]. 计算机研究与发展,2008, 45(8): 1 371-1 378.

[13] Dai Y, Yang L, Zhang B, et al. QoS for composite Web services and optimizing [J]. Chinese Journal of Computers,2006, 29(7): 1 167-1 178.

[责任编辑: 黄　敏]